CONSIDERING AUTOCORRELATION IN PREDICTIVE MODELS

Daniela Stojanova

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia, December 2012

Evaluation Board:
Prof. Dr. Marko Bohanec, Chairman, Jožef Stefan Institute, Ljubljana, Slovenia
Assoc. Dr. Janez Demšar, Member, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia
Asst. Prof. Michelangelo Ceci, Member, Università degli Studi di Bari “Aldo Moro”, Bari, Italy

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL
Ljubljana, Slovenia

Daniela Stojanova

CONSIDERING AUTOCORRELATION IN PREDICTIVE MODELS

Doctoral Dissertation

UPOŠTEVANJE AVTOKORELACIJE V NAPOVEDNIH MODELIH

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

Ljubljana, Slovenia, December 2012

Contents

To my family
Na mojata familija

Abstract

Most machine learning, data mining and statistical methods rely on the assumption that the analyzed data are independent and identically distributed (i.i.d.). More specifically, the individual examples included in the training data are assumed to be drawn independently from each other from the same probability distribution. However, cases where this assumption is violated can be easily found: For example, species are distributed non-randomly across a wide range of spatial scales. The i.i.d. assumption is often violated because of the phenomenon of autocorrelation.

The cross-correlation of an attribute with itself is typically referred to as autocorrelation: This is the most general definition found in the literature. Specifically, in statistics, temporal autocorrelation is defined as the cross-correlation between the attribute of a process at different points in time. In time-series analysis, temporal autocorrelation is defined as the correlation among time-stamped values due to their relative proximity in time. In spatial analysis, spatial autocorrelation has been defined as the correlation among data values, which is strictly due to the relative location proximity of the objects that the data refer to. It is justified by Tobler's first law of geography according to which "everything is related to everything else, but near things are more related than distant things". In network studies, autocorrelation is defined by the homophily principle as the tendency of nodes with similar values to be linked with each other.

In this dissertation, we first give a clear and general definition of the autocorrelation phenomenon, which includes spatial and network autocorrelation for continuous and discrete responses. We then present a broad overview of the existing autocorrelation measures for the different types of autocorrelation and data analysis methods that consider them. Focusing on spatial and network autocorrelation, we propose three algorithms that handle non-stationary autocorrelation within the framework of predictive clustering, which deals with the tasks of classification, regression and structured output prediction. These algorithms and their empirical evaluation are the major contributions of this thesis.

We first propose a data mining method called SCLUS that explicitly considers spatial autocorrelation when learning predictive clustering models. The method is based on the concept of predictive clustering trees (PCTs), according to which hierarchies of clusters of similar data are identified and a predictive model is associated to each cluster. In particular, our approach is able to learn predictive models for both a continuous response (regression task) and a discrete response (classification task). It properly deals with autocorrelation in data and provides a multi-level insight into the spatial autocorrelation phenomenon. The predictive models adapt to the local properties of the data, providing at the same time spatially smoothed predictions. We evaluate our approach on several real world problems of spatial regression and spatial classification.

The problem of "network inference" is known to be a challenging task. In this dissertation, we propose a data mining method called NCLUS that explicitly considers autocorrelation when building predictive models from network data. The algorithm is based on the concept of PCTs that can be used for clustering, prediction and multi-target prediction, including multi-target regression and multi-target classification. We evaluate our approach on several real world problems of network regression, coming from the areas of social and spatial networks. Empirical results show that our algorithm performs better

than PCTs learned by completely disregarding network information, CLUS* which is tailored for spatial data, but does not take autocorrelation into account, and a variety of other existing approaches.

We also propose a data mining method called NHMC for (Network) Hierarchical Multi-label Classification. This has been motivated by the recent development of several machine learning algorithms for gene function prediction that work under the assumption that instances may belong to multiple classes and that classes are organized into a hierarchy. Besides relationships among classes, it is also possible to identify relationships among examples. Although such relationships have been identified and extensively studied in the literature, in particular as defined by protein-to-protein interaction (PPI) networks, they have not received much attention in hierarchical and multi-class gene function prediction. Their use introduces the autocorrelation phenomenon and violates the i.i.d. assumption adopted by most machine learning algorithms. Besides improving the predictive capabilities of learned models, NHMC is helpful in obtaining predictions consistent with the network structure and consistently combining two information sources (hierarchical collections of functional class definitions and PPI networks). We compare different PPI networks (DIP, VM and MIPS for yeast data) and their influence on the predictive capability of the models. Empirical evidence shows that explicitly taking network autocorrelation into account can increase the predictive capability of the models, especially when the PPI networks are dense. NHMC outperforms CLUS-HMC (that disregards the network) for GO annotations, since these are more coherent with the PPI networks.

Povzetek

Večina metod za podatkovno rudarjenje, strojno učenje in statistično analizo podatkov temelji na predpostavki, da so podatki neodvisni in enako porazdeljeni (ang. independent and identically distributed – i.i.d.). To pomeni, da morajo biti učni primeri med seboj neodvisni ter imeti enako verjetnostno porazdelitev. Vendar so primeri, ko podatki niso i.i.d., v praksi zelo pogosti. Tako so na primer živalske vrste porazdeljene po prostoru nenaključno. Predpostavka i.i.d. je pogosto kršena zaradi avtokorelacije.

Najbolj splošna definicija avtokorelacije je, da je to prečna korelacija atributa samega s seboj. V statistiki je časovna avtokorelacija definirana kot prečna korelacija med atributom procesa ob različnem času. Pri analizi časovnih vrst je časovna avtokorelacija definirana kot korelacija med časovno odvisnimi vrednostmi zaradi njihove relativne časovne bližine. V prostorski analizi je prostorska avtokorelacija definirana kot korelacija med podatkovnimi vrednostmi, ki je nastala samo zaradi relativne bližine objektov, na katero se nanašajo podatki. Definicija temelji na prvem Toblerjevem zakonu o geografiji, po katerem "je vse povezano z vsem, vendar so bližje stvari bolj povezane kot oddaljene stvari." Pri analizi omrežij je avtokorelacija definirana s pomočjo načela homofilnosti, ki pravi, da vozlišča s podobnimi vrednostmi težijo k medsebojni povezanosti.

V disertaciji najprej podamo jasno in splošno definicijo avtokorelacije, ki vključuje prostorsko in omrežno avtokorelacijo za zvezne in diskretne spremenljivke. Nato predstavimo obširen pregled obstoječih mer za avtokorelacijo skupaj z metodami za analizo podatkov, ki jih uporabljajo. Osredotočimo se na prostorsko in omrežno avtokorelacijo in predlagamo tri algoritme, ki upoštevajo spremenljivo avtokorelacijo v okviru napovednega razvrščanja. Na ta način lahko obravnavamo klasifikacijske in regresijske naloge ter napovedovanje strukturiranih spremenljivk. Ti trije algoritmi in njihovo empirično vrednotenje so glavni prispevek disertacije.

Najprej predlagamo metodo podatkovnega rudarjenja SCLUS, ki izrecno upošteva prostorsko avtokorelacijo pri učenju modelov za napovedno razvrščanje. Metoda temelji na gradnji odločitvenih dreves za napovedno razvrščanje (DNR), pri kateri podatke razvrstimo v hierarhično strukturo s skupinami med seboj podobnih podatkov ter vsaki skupini predružimo napovedni model. Naša metoda omogoča učenje napovednih modelov za zvezne in diskretne ciljne spremenljivke (klasifikacija in regresija). Metoda pravilno upošteva avtokorelacijo v podatkih in omogoča večnivojski vpogled v pojav prostorske avtokorelacije. Napovedni modeli se prilagajajo lokalnim lastnostim podatkov in hkrati zagotavljajo gladko spreminjanje napovedi v prostoru. Naš pristop ovrednotimo na več različnih realnih problemih prostorske regresije in klasifikacije.

Problem "omrežnega sklepanja" je znan kot zahtevna naloga. V disertaciji predlagamo algoritem podatkovnega rudarjenja z imenom NCLUS, ki izrecno upošteva avtokorelacijo pri gradnji napovednih modelov na podatkih o omrežjih. Algoritem temelji na konceptu dreves za napovedno razvrščanje, ki jih je mogoče uporabiti za razvrščanje, regresijo in klasifikacijo preprostih ali strukturiranih spremenljivk. Naš pristop ovrednotimo na več različnih realnih problemih s področja socialnih in prostorskih omrežij. Empirični rezultati kažejo, da naš algoritem deluje bolje kot navadna drevesa za napovedno razvrščanje, zgrajena brez upoštevanja informacij o omrežjih, bolje kot metoda CLUS*, ki je prilagojena za analizo prostorskih podatkov, a ne upošteva avtokorelacije, in bolje od drugih obstoječih pristopov.

Predlagamo tudi metodo podatkovnega rudarjenja NHMC za hierarhično večznačkovno klasifikacijo. Motivacija za ta pristop je bil nedavni razvoj različnih algoritmov strojnega učenja za napovedovanje funkcij genov, ki delujejo pod predpostavko, da lahko primeri sodijo v več razredov, ti razredi pa so organizirani v hierarhijo. Poleg odvisnosti med razredi, je mogoče določiti tudi odvisnosti med primeri. Čeprav so te povezave identificirane in obširno raziskane v literaturi, še posebej v primeru omrežij interakcij med proteini (IMP), pa še vedno niso dovolj upoštevane v okviru hierarhične večznačkovne klasifikacije funkcij genov. Njihova uporaba uvaja avtokorelacijo in krši predpostavko neodvisnosti med primeri, na kateri temelji večina algoritmov strojnega učenja. Poleg izboljšane napovedne točnosti naučenih modelov, nam NHMC omogoča napovedi, ki so skladne s strukturo omrežja in konsistentno upoštevajo dva različna vira informacij (hierarhične zbirke funkcijskih razredov in omrežij IMP). Primerjali smo tri različna omrežja IMP (DIP, VM in MIPS pri kvasovkah) in njihovo napovedno točnost. Empirični rezultati kažejo, da upoštevanje omrežne avtokorelacije izboljša napovedno točnost modelov, še posebej v primeru, ko so omrežja IMP gosta. Metoda NHMC dosega boljše rezultate kot metoda CLUS-HMC (ki ne upošteva omrežja) za oznake GO (Gene Ontology), ker so te bolj usklajene z omrežji IMP.

Abbreviations

AUPRC	=	Area Under the Precision-Recall Curve
$\overline{AUPRC}$	=	Average AUPRC
AU$\overline{PRC}$	=	Area Under the average Precision-Recall Curve
ARIMA	=	AutoRegressive Integrated Moving Average
ARMA	=	AutoRegressive Moving Average
AR	=	AutoRegressive Model
CAR	=	Conditional AutoRegressive Model
CCF	=	Cross-Correlation Function
CC	=	Correlation Coefficient
CLUS	=	Software for Predictive Clustering, learns PCTs
CLUS-HMC	=	CLUS for Hierarchical Multi-label Classification
DAG	=	Directed Acyclic Graph
DM	=	Data Mining
GIS	=	Geographic Information System
GO	=	Gene Ontology
GWR	=	Geographically Weighted Regression
HMC	=	Hierarchical Multi-label Classification
ILP	=	Inductive Logic Programming
ITL	=	Iterative Transductive Regression
LiDAR	=	Light Detection and Ranging
MA	=	Moving Average Model
ML	=	Machine Learning
MRDM	=	Multi-Relational Data Mining
MTCT	=	Multi-Target Classification Tree
MTDT	=	Multi-Target Decision Tree
MT	=	Model Trees
NCLUS	=	Network CLUS
NHMC	=	Network CLUS for Hierarchical Multi-label Classification
PCT	=	Predictive Clustering Tree
PPI	=	Protein to Protein Interaction
PR	=	Precision-Recall
RMSE	=	Root Mean Squared Error
RRMSE	=	Relative Root Mean Squared Error
RS	=	Remote Sensing
RT	=	Regression Trees
SAR	=	Spatial AutoRegressive Model
SCLUS	=	Spatial CLUS
SPOT	=	Système Pour L'observation de la Terre
STARIMA	=	Spatio-Temporal AutoRegressive Integrated Moving Average
STCT	=	Single-Target Classification Tree

1 Introduction

In this introductory chapter, we first place the dissertation within the broader context of its research area. We then motivate the research performed within the scope of the dissertation. The major contributions of the thesis to science are described next. We conclude this chapter by giving an outline of the structure of the remainder of the thesis.

1.1 Outline

The research presented in this dissertation is placed in the area of artificial intelligence (Russell and Norvig, 2003), and more specifically in the area of machine learning. Machine learning is concerned with the design and the development of algorithms that allow computers to evolve behaviors based on empirical data, i.e., it studies computer programs that automatically improve with experience (Mitchell, 1997). A major focus of machine learning research is to extract information from data automatically by computational and statistical methods and make intelligent decisions based on the data. However, the difficulty lies in the fact that the set of all possible behaviors, given all possible inputs, is too large to be covered by the set of observed examples.

In general, there are two types of learning: inductive and deductive. Inductive machine learning (Bratko, 2000) is a very significant field of research in machine learning, where new knowledge is extracted out of data that describes experience and is given in the form of learning examples (instances). In contrast, deductive learning (Langley, 1996) explains a given set of rules by using specific information from the data.

Depending on the feedback the learner gets during the learning process, learning can be classified as supervised or unsupervised. Supervised learning is a machine learning technique for learning a function from a set of data. Supervised inductive machine learning, also called predictive modeling, assumes that each learning example includes some target property, and the goal is to learn a model that accurately predicts this property. On the other hand, unsupervised inductive machine learning, also called descriptive modeling, assumes no such target property to be predicted. Examples of machine learning methods for predictive modeling include decision trees, decision rules and support vector machines. In contrast, examples of machine learning methods for descriptive modeling include clustering, association rule modeling and principal-component analysis (Bishop, 2007).

In general, predictive and descriptive modeling are considered as different machine learning tasks and are usually treated separately. However, predictive modeling can be seen as a special case of clustering (Blockeel, 1998). In this case, the goal of predictive modeling is to identify clusters that are compact in the target space (i.e., group the instances with similar values of the target variable). The goal of descriptive modeling, on the other hand, is to identify clusters compact in the descriptive space (i.e., group the instances with similar values of the descriptive variables).

Predictive modeling methods are used for predicting an output (i.e., target property or target attribute) for an example. Typically, the output can be either a discrete variable (classification) or a continuous variable (regression). However, there are many real-life problems, such as text categorization, gene function prediction, image annotation, etc., where the input and/or the output are structured. Beside the typical classification and regression task, we also consider the latter, namely, predictive modeling tasks with structured outputs.

Predictive clustering (Blockeel, 1998) combines elements from both prediction and clustering. As in clustering, clusters of examples that are similar to each other are identified, but a predictive model is associated to each cluster. New instances are assigned to clusters based on cluster descriptions. The associated predictive models provide predictions for the target property. The benefit of using predictive clustering methods, as in conceptual clustering (Michalski and Stepp, 2003), is that besides the clusters themselves, they also provide symbolic descriptions of the constructed clusters. However, in contrast to conceptual clustering, predictive clustering is a form of supervised learning.

Predictive clustering trees (PCTs) are tree structured models that generalize decision trees. Key properties of PCTs are that *i)* they can be used to predict many or all attributes of an example at once (multi-target), *ii)* they can be applied to a wide range of prediction tasks (classification and regression) and *iii)* they can work with examples represented by means of a complex representation (Džeroski et al, 2007), which is achieved by plugging in a suitable distance metric for the task at hand. PCTs were first implemented in the context of First-Order logical decision trees, in the system TILDE (Blockeel, 1998), where relational descriptions of the examples are used. The most known implementation of PCTs, however, is the one that uses attribute-value descriptions of the examples and is implemented in the predictive clustering framework of the CLUS system (Blockeel and Struyf, 2002). The CLUS system is available for download at `http://sourceforge.net/projects/clus/`.

Here, we extend the predictive clustering framework to work in the context of autocorrelated data. For such data the independence assumption which typically underlies machine learning methods and multivariate statistics, is no longer valid. Namely, the autocorrelation phenomenon directly violates the assumption that the data instances are drawn independent from each other from an identical distribution (i.i.d.). At the same time, it offers the unique opportunity to improve the performance of predictive models which would take it into account.

Autocorrelation is very common in nature and has been investigated in different fields, from statistics and time-series analysis, to signal-processing and music recordings. Here we acknowledge the existence of four different types of autocorrelation: spatial, temporal, spatio-temporal and network (relational) autocorrelation, describing the existing autocorrelation measures and the data analysis methods that consider them. However, in the development of the proposed algorithms, we focus on spatial autocorrelation and network autocorrelation. In addition, we also deal with the complex case of predicting structured targets (outputs), where network autocorrelation is considered.

In the PCT framework (Blockeel, 1998), a tree is viewed as a hierarchy of clusters: the top-node contains all the data, which is recursively partitioned into smaller clusters while moving down the tree. This structure allows us to estimate and exploit the effect of autocorrelation in different ways at different nodes of the tree. In this way, we are able to deal with non-stationarity autocorrelation, i.e., autocorrelation which may change its effects over space/networks structure.

PCTs are learned by extending the heuristics functions used in tree induction to include the spatial/network autocorrelation. In this way, we obtain predictive models that are able to deal with autocorrelated data. More specifically, beside maximizing the variance reduction which minimizes the intra-cluster distance in the class labels associated to examples, we also maximize cluster homogeneity in terms of autocorrelation at the same time doing the evaluation of candidate splits for adding a new node to the tree. This results in improved predictive performance of the obtained models and in smother predictions.

A diverse set of methods that deal with this kind of data, in several fields of research, already exists in the literature. However, most of them either deal with specific case studies or assume a specific experimental setting. In the next section, we describe the existing methods and motivate our work.

1.2 Motivation

The assumption that data examples are independent from each other and are drawn from the same probability distribution, i.e., that the examples are independent and identically distributed (i.i.d.), is common to most of the statistical and machine learning methods. This assumption is important in the classical form of the central limit theorem, which states that the probability distribution of the sum (or average) of i.i.d. variables with finite variance approaches a normal distribution. While this assumption tends to simplify the underlying mathematics of many statistical and machine learning methods, it may not be realistic in practical applications of these methods.

In many real-world problems, data are characterized by a form of autocorrelation, where the value of a variable for a given example depends on the values of the same variable in related examples. This is the case for the spatial proximity relation encounter in spatial data, where data measured at nearby locations often (but not always) influence each other. For example, species richness at a given site is likely to be similar to that of a site nearby, but very much different from sites far away. This is due to the fact that the environment is more similar within a shorter distance and the above phenomenon is referred to as spatial autocorrelation (Tobler, 1970).

The case of the temporal proximity relation encounter in time-series data is similar: data measured at a given time point are not completely independent of the past values. This phenomenon is referred to as temporal autocorrelation (Epperson, 2000). For example, weather conditions are highly autocorrelated within one year due to seasonality. A weaker correlation exists between weather variables in consecutive years.

A similar phenomenon also occurs in network data, where the values of a variable at a certain node often (but not always) depend on the values of the variables at the nodes connected to the given node: This phenomenon is referred to as network homophily (Neville et al, 2004). Recently, networks have become ubiquitous in several social, economical and scientific fields ranging from the Internet to social sciences, biology, epidemiology, geography, finance, and many others. Researchers in these fields have demonstrated that systems of different nature can be represented as networks (Newman and Watts, 2006). For instance, the Web can be considered as a network of web-pages, connected with each other by edges representing various explicit relations, such as hyperlinks. Social networks can be seen as groups of members that can be connected by friendship relations or can follow other members because they are interested in similar topics of interests. Gene networks can provide insight about genes and their possible relations of co-regulation based on the similarities of their expression levels. Finally, in epidemiology, networks can represent the spread of diseases and infections.

Moreover, in many real-life problems of predictive modeling, not only the data are not independent and identically distributed (i.i.d.), but the output (i.e., the target property) is structured, meaning that there can be dependencies between classes (e.g., classes are organized into a tree-shaped hierarchy or a directed acyclic graph). These types of problems occur in domains such as the life sciences (predicting gene function), ecology (analysis of remotely sensed data, habitat modeling), multimedia (annotation and retrieval of images and videos), and the semantic web (categorization and analysis of text and web content). The amount of data in these areas is increasing rapidly.

A variety of methods, specialized in predicting a given type of structured output (e.g., a hierarchy of classes (Silla and Freitas, 2011)), have been proposed (Bakır et al, 2007). However, many of them are computationally demanding and not suited for dealing with large datasets and especially with large outputs spaces. The predictive clustering framework offers a unifying approach for dealing with different types of structured outputs and the algorithms developed in this framework construct the predictive models very efficiently. Moreover, PCTs can be easily interpreted by a domain expert, thus supporting the process of knowledge extraction.

Since the presence of autocorrelation introduces a violation of the i.i.d. assumption, the work on this analysis of such data needs to take this into account.

Such work either removes the autocorrelation dependencies during pre-processing and then use traditional algorithms (e.g., (Hardisty and Klippel, 2010; Huang et al, 2004)) or modifies the classical machine learning, data mining and statistical methods in order to consider the autocorrelation (e.g., (Bel et al, 2009; Rinzivillo and Turini, 2004, 2007)). There are also approaches which use a relational setting (e.g., (Ceci and Appice, 2006; Malerba et al, 2005)), where the autocorrelation is usually incorporated through the data structure or defined implicitly through relationships among the data and other data properties.

However, one limitation of most of the approaches that take autocorrelation into account is that they assume that autocorrelation dependencies are constant (i.e., do not change) throughout the space/network (Angin and Neville, 2008). This means that possible significant variability in autocorrelation dependencies in different points of the space/network cannot be represented and modeled. Such variability could result from a different underlying latent structure of the space/network that varies among its parts in terms of properties of nodes or associations between them. For example, different research communities may have different levels of cohesiveness and thus cite papers on other topics with varying degrees. As pointed out by Angin and Neville (2008), when autocorrelation varies significantly throughout a network, it may be more accurate to model the dependencies locally rather than globally.

In the dissertation, we extend the predictive clustering framework in the context of PCTs that are able to deal with data (spatial and network) that do not follow the i.i.d. assumption. The distinctive characteristic of the proposed approach is that it explicitly considers the non-stationary (spatial and network) autocorrelation when building the predictive models. Such a method not only extends the applicability of the predictive clustering approach, but also exploits the autocorrelation phenomenon and uses it to make better predictions and better models.

In traditional PCTs (Blockeel, 1998), the tree construction is performed by maximizing variance reduction. This heuristic guarantees, in principle, accurate models since it reduces the error on the training set. However, it neglects the possible presence of autocorrelation in the training data. To address this issue, we propose to simultaneously maximize autocorrelation for spatial/network domains. In this way, we exploit the spatial/network structure of the data in the PCT induction phase and obtain predictive models that naturally deal with the phenomenon of autocorrelation.

The consideration of autocorrelation in clustering has already been investigated in the literature, both for spatial clustering (Glotsos et al, 2004) and network clustering (Jahani and Bagherpour, 2011). Motivated by the demonstrated benefits of considering autocorrelation, we exploit some characteristics of autocorrelated data to improve the quality of PCTs. The consideration of autocorrelation in clustering offers several advantages, since it allows us to:

- determine the strength of the spatial/network arrangement on the variables in the model;
- evaluate stationarity and heterogeneity of the autocorrelation phenomenon across space;
- identify the possible role of the spatial/network arrangement/distance decay on the predictions associated with each of the nodes of the tree;
- focus on the spatial/network "neighborhood" to better understand the effects that it can have on other neighborhoods and vice versa.

These advantages of considering spatial autocorrelation in clustering, identified by (Arthur, 2008), fit well into the case of PCTs. Moreover, as recognized by (Griffith, 2003), autocorrelation implicitly defines a zoning of a (spatial) phenomenon: Taking this into account reduces the effect of autocorrelation on prediction errors. Therefore, we propose to perform clustering by maximizing both variance reduction and cluster homogeneity (in terms of autocorrelation) at the same time, during the phase of adding a new node to the predictive clustering tree.

The network (spatial and relational) setting that we address in this work is based on the use of both the descriptive information (attributes) and the network structure during training, whereas we only use the descriptive information in the testing phase and disregard the network structure. More specifically, in the training phase, we assume that all examples are labeled and that the given network is complete. In the testing phase, all testing examples are unlabeled and the network is not given. A key property of our approach is that the existence of the network is not obligatory in the testing phase, where we only need the descriptive information. This can be very beneficial when predictions need to be made for those examples for which connections to others examples are not known or need to be confirmed. The more common setting where a network with some nodes labeled and some nodes unlabeled is given, can be easily mapped to our setting. We can use the nodes with labels and the projection of the network on these nodes for training and only the unlabeled nodes without network information in the testing phase.

This network setting is very different from the existing approaches to network classification and regression where the descriptive information is typically in a tight connection to the network structure. The connections (edges in the network) between the data in the training/testing set are predefined for a particular instance and are used to generate the descriptive information associated to the nodes of the network (see, for example, (Steinhaeuser et al, 2011)). Therefore, in order to predict the value of the response variable(s), besides the descriptive information, one needs the connections (edges in the network) to related/similar entities. This is very different from what is typically done in network analysis as well. Indeed, the general focus there is on exploring the structure of a network by calculating its properties (e.g. the degrees of the nodes, the connectedness within the network, scalability, robustness, etc.). The network properties are then fitted into an already existing mathematical (theoretical) network (graph) model (Steinhaeuser et al, 2011).

From the predictive perspective, according to the tests in the tree, it is possible to associate an observation (a test node of a network) to a cluster. The predictive model associated to the cluster can then be used to predict its response value (or response values, in the case of multi-target tasks). From the descriptive perspective, the tree models obtained by the proposed algorithm allow us to obtain a hierarchical view of the network, where clusters can be employed to design a federation of hierarchically arranged networks.

A hierarchial view of the network can be useful, for instance, in wireless sensor networks, where a hierarchical structure is one of the possible ways to reduce the communication cost between the nodes (Li et al, 2007). Moreover, it is possible to browse the generated clusters at different levels of the hierarchy, where each cluster can naturally consider different effects of the autocorrelation phenomenon on different portions of the network: at higher levels of the tree, clusters will be able to consider autocorrelation phenomenons that are spread all over the network, while at lower levels of the tree, clusters will reasonably consider local effects of autocorrelation. This gives us a way to consider non-stationary autocorrelation.

1.3 Contributions

The research presented in this dissertation extends the PCT framework towards learning from autocorrelated data. We address important aspects of the problem of learning predictive models in the case when the examples in the data are not i.i.d, such as the definition of autocorrelation measures for a variety of learning tasks that we consider, the definition of autocorrelation-based heuristics, the development of algorithms that use such heuristics for learning predictive models, as well as their experimental evaluation.

In our broad overview, we consider four different types of autocorrelation: spatial, temporal, spatio-temporal and network (relational) autocorrelation, we survey the existing autocorrelation measures and methods that consider them. However, in the development of the proposed algorithms, we focus only on spatial and network autocorrelation.

The corresponding findings of our research are published in several conference and journal publications in the areas of machine learning and data mining, ecological modeling, ecological informatics and bioinformatics. The complete list of related publications is given in Appendix A. In the following, we summarize the main contributions of the work.

- The major contributions of this dissertation are three extensions of the predictive clustering approach for handling non-stationary (spatial and network) autocorrelated data for different predictive modeling tasks. These include:
  - **SCLUS** (Spatial Predictive Clustering System) (Stojanova et al, 2011) (chapter 6), that explicitly considers spatial autocorrelation in regression (and classification),
  - **NCLUS** (Network Predictive Clustering System) (Stojanova et al, 2011, 2012) (chapter 7), that explicitly considers network autocorrelation in regression (and classification), and
  - **NHMC** (Network Hierarchical Multi-label Classification) (Stojanova et al, 2012) (chapter 8), that explicitly considers network autocorrelation in hierarchical multi-label classification.
- The algorithms are heuristic: we define new heuristic functions that take into account both the variance of the target variables and its spatial/network autocorrelation. Different combinations of these two components enable us to investigate their influence in the heuristic function and on the final predictions.
- We perform extensive empirical evaluation of the newly developed methods on single target classification and regression problems, as well as multi-target classification and regression problems.
  - We compare the performance of the proposed predictive models for classification and regression tasks, when predicting single and multiple targets simultaneously, to current state-of-the-art methods (chapters 6, 7, 8). Our approaches compare very well to mainstream methods that do not consider autocorrelation, as well as to well-known methods that consider autocorrelation. Furthermore, our approach can more successfully remove the autocorrelation of the errors of the obtained models. Finally, the obtained predictions are more coherent in space (or in the network context).
  - We also apply the proposed predictive models to real-word problems, such as the prediction of outcrossing rates from genetically modified crops to conventional crops in ecology (Stojanova et al, 2012) (chapter 6), prediction of the number of views of online lectures (Stojanova et al, 2011, 2012) (chapter 7) and protein function prediction in functional genomics (Stojanova et al, 2012) (chapter 8).

1.4 Organization of the dissertation

This introductory chapter presents the general perspective and context of the dissertation. It also specifies the motivation for performing the research and lists the main original scientific contributions. The rest of the dissertation is organized as follows.

In Chapter 2, first we give a broad overview of the field of predictive modeling and present the most important predictive modeling tasks. Next, we explain the relational aspects that we consider along with the relations themselves stressing their origin and use within our experimental settings. Finally, we define the different forms of autocorrelation that we consider and discuss them along two dimension:

the first one considering the type of targets in predictive modeling and the second one focusing on the type of relations considered in the predictive model: spatial, temporal, spatio-temporal and network autocorrelation.

In Chapter 3, we present the existing measures of autocorrelation. In particular, we divide the measures according to the different forms of autocorrelation that we consider: spatial, temporal, spatio-temporal and network autocorrelation, as well as accordingly to the predictive (classification and regression) task that they are defined for. Finally, we introduce new measures of autocorrelation for classification and regression that we have defined by adapting the existing ones, in order to deal with the autocorrelation phenomenon in the experimental settings that we use.

In Chapter 4, we present an overview of relevant related work and their main characteristics focussing on predictive modeling methods listed in the literature that consider different forms of autocorrelation. The way that these works take autocorrelation into account is particularly emphasized. The relevant methods are organized according to the different forms of autocorrelation that they consider, as well as accordingly to the predictive (classification and regression) task that they concern.

In Chapter 5, we give an overview of the predictive clustering trees framework focussing on the different predictive modeling tasks that it can handle, from standard classification and regression tasks to multi-target classification and regression, as well as hierarchial multi-label classification, as a special case of predictive modeling with structured outputs. The extensions that we propose in the following chapters are situated in this framework and inherit its characteristics.

Chapter 6 describes the proposed approach for building predictive models from spatially autocorrelated data, which is one of the main contributions of this dissertation. In particular, we focus on the single and multi-target classification and regression tasks. First, we present the experimental questions that we address, the real-life spatial data, the evaluation measures and the parameter instantiations for the learning methods. Next, we stress the importance of the selection of the bandwidth parameter and analyze the time complexity of the proposed approach. Finally, we present and discuss the results for each considered task separately, in terms of their accuracy, as well as in terms of the properties of the predictive models by analyzing the model sizes, the autocorrelation of the errors of the predictive models and their learning times.

Chapter 7 describes the proposed approach for learning predictive models from network autocorrelated data, which is another main contribution of this dissertation. Regression inference in network data is a challenging task and the proposed algorithm deals both with single and multi-target regression tasks. First, we present the experimental questions that we address, the real-life network data, the evaluation measures and the parameter instantiations for the learning methods. Next, we present and discuss the obtained results.

Chapter 8 describes the proposed approach for learning predictive models from network autocorrelated data, with the complex case of the Hierarchial Multi-Label Classification (HMC) task. Focusing on functional genomics data, we learn to predict the (hierarchically organized) protein functional classes, considering the autocorrelation that comes from the protein-to-protein interaction (PPI) networks. We evaluate the performance of the proposed algorithm and compare its predictive performance to already existing methods, using different yeast data and different yeast PPI networks. The development of this approach is the final main contribution of this dissertation.

Finally, chapter 9 presents the conclusions drawn from the presented research, including the development of the different algorithms and their experimental evaluation. It first presents a summary of the dissertation and its original contributions, and then outlines the possible directions for further work.

2 Definition of the Problem

The work presented in this dissertation concerns with the problem of learning predictive clustering trees that are able to deal with the global and local effects of the autocorrelation phenomenon. In this chapter, we first define the most important predictive modeling tasks that we consider. Next, we explain the relational aspects taken into account within the defined predictive modeling tasks. We focus on the different types of relations that we consider and describe their origin. Moreover, we explain their use and importance within our experimental setting. Finally, we introduce the concept of autocorrelation. In particular, we define the different forms of autocorrelation that we consider and discuss them along two orthogonal dimensions: the first one considering the type of targets in predictive modeling and the second one focusing on the type of relations considered in the predictive model.

2.1 Learning Predictive Models from Examples

Predictive analysis is the area of data mining concerned with forecasting the output (i.e., target attribute) for an example. Predictive modeling is a process used in predictive analysis to create a mathematical model of future behavior. Typically, the output can be either a discrete variable (classification) or a continuous variable (regression). However, there are many real-life domains, such as text categorization, gene networks, image annotation, etc., where the input and/or the output can be structured.

A predictive model consists of a number of predictors (i.e., descriptive attributes), which are independent variables that are likely to influence future behavior or results. In marketing, for example, a customer's gender, age, and purchase history might predict the likelihood of a future sale.

In predictive modeling, data is collected for the relevant descriptive attributes, a mathematical model is formulated, (for some type of models attributes are generated) and the model is validated (or revised) as the additional data becomes available. The model may employ a simple linear equation or a complex neural network, or a decision tree.

In the model building (training) process a predictive algorithm is constructed based on the values of the descriptive attributes for each example in the training data, i.e., *training set*. The model can then be applied to a different (testing) data, i.e., *test set* in which the target values are unknown. The test set is usually independent of the training set, but that follows the same probability distribution.

Predictive modeling usually underlays on the assumption that data (sequences or any other collection of random variables) is independent and identically distributed (i.i.d.) (Clauset, 2001). It implies that an element in the sequence is independent of the random variables that came before it. By doing so, it tends to simplify the underlying mathematics of many statistical methods. This assumption is different from a Markov Sequence (Papoulis, 1984) where the probability distribution for the n-th random variable is a function of the $n-1$ random variable (for a First Order Markov Sequence).

The assumption is important in the classical form of the central limit theorem (Rice, 2001), which states that the probability distribution of the sum (or average) of i.i.d. variables with finite variance approaches a normal distribution. However, in practical applications of statistical modeling this assumption may or may not be realistic.

Motivated by the violations of the i.i.d. assumption in many real-world cases, we consider the predictive modeling task without this assumption. The task of predictive modeling that we consider can be

formalized as follows.
Given:

- A descriptive space X that consists of tuples of values of primitive data types (boolean, discrete or continuous) spanned by m independent (or predictor) variables X_j, i.e., $\mathbf{X} = \{X_1, X_2, \ldots X_m\}$,
- A target space Y which is a tuple of several variables (discrete or continuous) or a structured object (e.g., a class hierarchy), spanned by T dependent (or target), i.e., $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_T\}$, variables Y_j,
- A context space D of dimensional variables (e.g., spatial coordinates) that typically consists of tuples $\mathbf{D} = \{D_1, D_2, \ldots D_r\}$ on which a distance $d(\cdot,\cdot)$ is defined,
- A set E of training examples, (x_i, y_i) with $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$,
- a quality criterion q defined on a predictive model and a set of examples, which rewards models with high predictive accuracy and low complexity.

Find: a predictive model (i.e., a function) $f : X \rightarrow Y$, such that f maximizes q.

This formulation differs from the classical formulation of the predictive modeling by the context space D, that is the result of the violation of the i.i.d. assumption. The context space D serves as a background knowledge and introduces information, related to the target space, in form of relations between the training examples. D is not directly included in the mapping $X \rightarrow Y$, enabling the use of propositional data mining setup (one table representation of the data). Moreover, they can be cases when this context space is not defined with dimensional variables, but only using a distance $d(\cdot,\cdot)$ over the context space. This is discussed in more details in the next section.

Also, note that the function f will be represented with decision trees, i.e., predictive clustering trees (PCTs). Furthermore, beside the typical classification and regression task, we are also concerned with the predictive modeling tasks where the outputs are structured.

2.1.1 Classification

Classification task is the task of learning a function that maps (classifies) a dependent variable into one of several predefined classes (Bratko, 2000). This means that the goal is to learn a model that accurately predicts an independent discrete variable. Examples include detecting spam email messages based upon the messages header and content, categorizing cells as malignant or benign based upon the results of MRI scans, etc.

A classification task begins with a training set E with descriptive (boolean, discrete or continuous) attributes X and discrete target variable Y. In the training process, a classification algorithm classifies the examples to previously given classes based on the values of the descriptive attributes for each example in the training set, i.e., maps the examples according to a function $f : X \rightarrow Y$. The model than can be applied to different test sets, in which the target values are unknown.

Table 2.1 shows an example of dataset with several continuous descriptive attributes, two dimensional attributes and a discrete target. The descriptive attributes describe the environmental conditions of the study area, the dimensional attributes are the spatial coordinates, while the target is binary (0,1) with 1 representing presence of contamination at sample points and 0 representing absence of contamination at sample point.

A classification model can serve as an explanatory tool to distinguish among examples of different classes or to predict the class label of unknown data. For example, it would be useful for biologists to have a model that summarizes the most important characteristics of the data and explains what features

Table 2.1: *An example of dataset with one target, multiple continuous attributes and one discrete attribute.* The target is the contamination (outcrossing) at sample points that comes from the surrounding genetically modified fields, the dimensional attributes are the spatial coordinates, while the descriptive attributes describe the environmental conditions of the study area.

Target	Dimensional variables		Attributes							
Contamination	X	Y	NbGM	NonGMarea	GMarea	AvgDist	AvgVisA	AvgWEdge	MaxVisA	MaxWEdge
0	502100	4655645	2	1.131	14032.315	86.901	3.518	726.013	89.123	18147.61
1	501532	4655630	2	0.337	31086.363	39.029	5.731	1041.469	156.045	28895.724
1	501567	4655640	2	0.4	26220.011	57.713	5.538	1011.021	156.045	29051.768
0	501555	4655680	2	0.637	16458.855	74.621	5.538	1014.568	156.045	29207.813

define the different species. Moreover, it would be useful for them to have a model that can forecast the type of new species based on already known features.

A learning algorithm is employed to identify a model that best fits the relationship between the attribute set and the class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of the data it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capabilities, i.e., models that accurately predict the class labels of previously unknown data.

The most common learning algorithms used in the classification task include decision tree classifiers, rule-based classifiers, neural networks, support vector machines and naive Bayes classifiers.

Classification modeling has many applications in text classification, finance, biomedical and environmental modeling.

2.1.2 Regression

Regression task is the task of learning a function which maps a dependent variable to a real-valued prediction variable (independent continuous variable) (Witten and Frank, 2005). This is different from the task of classification and can be treated as a special case of classification when the target variable is numeric. Examples include predicting the value of a house based on location, number of rooms, lot size, and other factors; predicting the ages of customers as a function of various demographic characteristics and shopping patterns; predicting the mortgage rates etc. Moreover, profit, sales, mortgage rates, house values, square footage, temperature, or distance could all be predicted using regression techniques.

A regression task begins with a training set E with descriptive (boolean, discrete or continuous) attributes X and continuous target variable Y. In the model building (training) process, a regression algorithm estimates the value of the target as a function of the predictors for each case in the training data, i.e., $f : X \rightarrow Y$. These relationships between predictors and target are summarized in a model, which can then be applied to different test sets in which the target values are unknown.

Figure 2.2 shows an example of dataset with several continuous descriptive attributes, two dimensional attributes and a continuous target. The descriptive attributes describe the environmental conditions of the study area, the dimensional attributes are the spatial coordinates, while the target represents the measurements of pollen dispersal (crossover) rates.

Regression models are tested by computing various statistics that measure the difference between the predicted values and the expected values.

Regression modeling has many applications in trend analysis, business planning, marketing, financial forecasting, time series prediction, biomedical and drug response modeling, and environmental modeling.

Table 2.2: *An example of dataset with multiple continuous attributes and one target.* The descriptive attributes describe the environmental conditions of the study area, the dimensional attributes present the spatial coordinates, while the target is pollen dispersal coming from fertile male populations of GM crops (Stojanova et al, 2012).

Dimensional variables		Attributes				Target
X	Y	Angle	CenterDistance	MinDistance	VisualAngle	MF
3	3	-0.75	57.78	50.34	0.24	0.00
3	6	-2.55	55.31	48.27	0.25	0.00
3	9	0.75	57.78	12.20	0.70	0.17
3	12	0.99	57.78	14.76	0.63	0.44
6	3	0.54	57.21	14.34	0.64	0.00
6	6	2.33	21.08	50.79	0.72	0.31
6	9	2.11	17.49	10.77	0.76	0.55
6	12	1.99	16.15	10.55	0.78	0.00

2.1.3 Multi-Target Classification and Regression

The task of Multi-Target learning refers to the case when learning two or more, discrete or continuous, target variables at the same time (Struyf and Džeroski, 2006). In the case where there are two or more discrete target variables, the task is called *Multi-Target Classification*, whereas in the case where there are two or more continuous target variables, the task is called *Multi-Target Regression*. In contrast to classification and regression where the output is a single scalar value, in this case the output is a vector containing two or more classes depending on the number of target variables.

Examples of Multi-Target learning include predicting two target variables, such as predicting the male and female population in an environmental modeling; predicting the genetically modified and non-genetically modified crops in ecological modeling; categorizing the malignant and benign cells based upon the results of MRI scans, etc. An example of Multi-Target learning of more than two target variables is predicting canopy cover and forest stand properties (vegetation height, percentage of vegetation cover, percentage of vegetation, vertical vegetation profiles at 99, 95, 75, 50, 25, 10, 5 percentiles of vegetation height, vertical vegetation profiles at maximum height) from satellite images in forestry. There are 11 target variables in this example.

A Multi-Target classification (regression) learning task begins with a training set E with descriptive (boolean, discrete/continuous attributes X and discrete (continuous) target variables Y. In the model building (training) process, a function of the predictors for each case in the training data, i.e., a function $f : X \rightarrow Y$ is mapped. These relationships between predictors and targets are summarized in a model, which can then be applied to a different test sets in which the target values are unknown.

Table 2.3 shows an example of dataset with several continuous descriptive attributes and two continuous targets. The descriptive attributes describe the environmental conditions of the study area, the dimensional attributes are the spatial coordinates, while the targets are measurements of pollen dispersal (crossover) rates from two lines of plants (fertile and sterile male populations of GM crops) (Stojanova et al, 2012).

The advantages of such learning (over learning a separate model for each target variable) are that: *i)* a multi-target model is learned instead of two or more separate models for each target variable, *ii)* such a multi-target model explicates dependencies between the different target variables, *iii)* the learning of such a model saves time and resources *iv)* the size of such a multi-target model is smaller than the total size of the individual models for all target variables, and *v)* smaller models are usually more comprehensive and easier to use in practice.

Multi-target models however do not always lead to more accurate predictions. For a given target variable, the variable's single-target model may be more accurate than the multi-target model.

Table 2.3: *An example of dataset with multiple continuous attributes and two targets.* The descriptive attributes describe the environmental conditions of the study area, the dimensional attributes are the spatial coordinates, while the targets are measurements of pollen dispersal (crossover) rates from two lines of plants (fertile and sterile male populations of GM crops) (Stojanova et al, 2012).

Dimensional variables		Attributes				Targets	
X	Y	Angle	CenterDistance	MinDistance	VisualAngle	MF	MS
3	3	-0.75	57.78	50.34	0.24	0.00	0.00
3	6	-2.55	55.31	48.27	0.25	0.00	0.00
3	9	0.75	57.78	12.20	0.70	0.17	0.63
3	12	0.99	57.78	14.76	0.63	0.44	2.56
6	3	0.54	57.21	14.34	0.64	0.00	0.46
6	6	2.33	21.08	50.79	0.72	0.31	0.33
6	9	2.11	17.49	10.77	0.76	0.55	4.64
6	12	1.99	16.15	10.55	0.78	0.00	0.27

2.1.4 Hierarchial Multi-Label Classification

In many real-life problems of predictive modeling, the input/output are not discrete or continuous variables and therefore cannot be handled by the classical classification and regression tasks. In such cases, we deal with complex variables and treat them as objects. Often, the output is structured, i.e., there can exist dependencies between classes or some internal relations between the classes (e.g., classes are organized into a tree-shaped hierarchy or a directed acyclic graph). These types of problems are motivated by an increasing number of new applications including semantic annotation of images and video(news clips, movies clips), functional genomics (gene and protein function), music categorization into emotions, text classification (news articles, web pages, patents, emails, bookmarks, ...), directed marketing and others.

The task of *Hierarchial Multi-Label Classification* (HMC) (Silla and Freitas, 2011) is concerned with learning models for predicting structured outputs such that there can exist dependencies between classes or some internal relations between the classes (e.g., classes are organized into a tree-shaped hierarchy or a directed acyclic graph). It takes as input a tuple of attribute values and produces as output a structured object (hierarchy of classes).

We formally define the task of hierarchical multi-label classification as follows:
Given:

- A description space X that consists of tuples of values of primitive data types (boolean, discrete or continuous), i.e., $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, ..., x_{i_{Des}})$, where Des is the size of the tuple (or number of descriptive variables),

- a target space S, defined with a class hierarchy $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order (structured as a rooted tree) representing the superclass relationship ($\forall\ c_1, c_2 \in C : c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2),

- A context space D of dimensional variables (e.g., spatial coordinates) that typically consists of tuples $\mathbf{D} = \{D_1, D_2, \ldots D_r\}$ on which a distance $d(\cdot,\cdot)$ is defined,

- a set E, where each example is a pair of a tuple and a set from the descriptive and target space respectively, and each set satisfies the hierarchy constraint, i.e., $E = \{(X_i, S_i) || X_i \in X, S_i \subseteq C, c \in S_i \Rightarrow \forall c' \leq_h c : c' \in S_i, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = ||E||$), and

- a quality criterion q, which rewards models with high predictive accuracy and low complexity.

Find: a function $f : D \times X \longmapsto 2^C$ (where 2^C is the power set of C) such that f maximizes q and $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$, i.e., predictions made by the model satisfy the *hierarchy constraint*.

The context space D serves as a background knowledge and is a result of the violation of the i.i.d. assumption, also for this specific task.

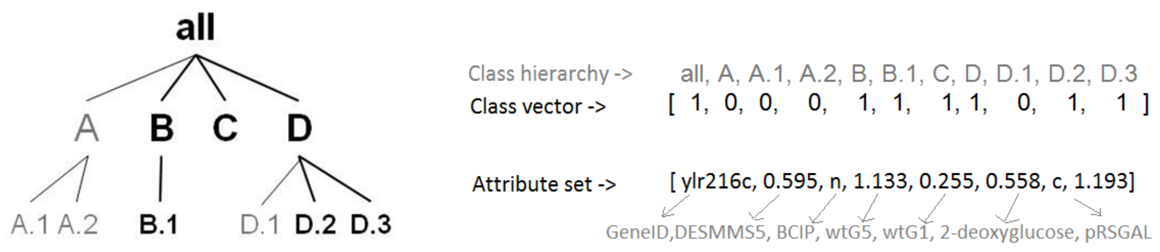

all
A
B
C
D
A.1 A.2
B.1
D.1 D.2 D.3
Class hierarchy -> all, A, A.1, A.2, B, B.1, C, D, D.1, D.2, D.3
Class vector -> [1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1]
Attribute set -> [ylr216c, 0.595, n, 1.133, 0.255, 0.558, c, 1.193]
GeneID,DESMMS5, BCIP, wtG5, wtG1, 2-deoxyglucose, pRSGAL

Figure 2.1: *An example of dataset for hierarchical multi-label classification of functional genomics data.* The descriptive attributes present different aspects of the genes in the yeast genome, while the targets are protein functional classes organized hierarchically. (Stojanova et al, 2012).

Figure 2.1 gives an example of hierarchical multi-label classification. In particular, it presents an example dataset for hierarchical multi-label classification of functional genomics data. On the left part of the figure the targets, i.e., protein functional classes are presented in form of a tree (hierarchy structure). The hierarchy has three levels of classes. On the right part of the figure these same targets are presented as a binary vector. We can see that the targets that are given in bold in the hierarchy on the left are the ones that have values of one in the binary vector representation on the right, i.e., the one that actually appear in the given example. On the right part of the figure we also present the descriptive (discrete and continuous) attributes that describe different aspects of the genes in the yeast genome (Stojanova et al, 2012).

In contrast to classification and regression, where the output is a single scalar value, in this case the output is a data structure, such as a tuple or a Directed Acyclic Graph (DAG). This is also in contrast to the multi-target classification where each example is associated with a single label from a finite set of disjoint labels, as well as to multi-label classification where the labels are not assumed to be mutually exclusive: multiple labels may be associated with a single example, i.e., each example can be a member of more than one class.

These types of problems occur in domains such as life sciences (gene function prediction, drug discovery), ecology (analysis of remotely sensed data, habitat modeling), multimedia (annotation and retrieval of images and videos) and the semantic web (categorization and analysis of text and web). Having in mind the needs of the application domains and the increasing quantities of structured data, the task of "mining complex knowledge from complex data" was listed as one of the most challenging problems in data mining (Yang and Wu, 2006).

2.2 Relations Between Examples

When learning predictive models the i.i.d. assumption is violated in many real-world cases. Therefore, as we mentioned in the previous section, we consider the predictive modeling tasks without this assumption. Having defined a context space D that serves as a background knowledge and introduces additional information, related to the target space, and especially the distance $d(\cdot,\cdot)$ on which it is defined, we can say that data is linked through some kind of relations. Moreover, they can be cases when this context space is not explicitly defined using dimensional variables, but only using a distance $d(\cdot,\cdot)$ over the context space.

The relations that we consider are relations between the examples. They can be inherited from the data itself or artificially defined by the user, in order to substitute the missing natural ones or to enable some learning tasks to be accordingly addressed. We argue the use of these relations and emphasize the advantages of the used experimental settings which serve as a basis of our work.

In the following Section, first we focus on the different types of relations that we consider. Then, we describe their origin. Finally, we explain their use within our experimental settings.

2.2.1 Types of Relations

Common machine learning algorithms assume that the data is stored in a single table where each example is represented by a fixed number of descriptive attributes. These are called attribute-value or propositional techniques (as the patterns found can be expressed in propositional logic). Propositional machine learning techniques (such as the classification or regression decision trees discussed in the previous section) are popular mainly because they are efficient, easy to use and are widely accessible. However, in practice, the single table assumption turns out to be a limiting factor for many machine learning tasks that involve data that comes from different data sources.

Traditional classification techniques would ignore the possible relations among examples, since the main assumption on which rely most of the data mining, machine learning and statistical methods is the data independence assumption. According to this, the examples included in the learning process must be independent from each other and identically distributed (i.i.d.). However, examples of non i.i.d. data can be found everywhere: nature is not independent; species are distributed non-randomly across a wide range of spatial scales, etc. The consideration of such relations among example is related to phenomenon of autocorrelation.

The autocorrelation phenomenon is a direct violation of the assumption that data are independently and identically distributed (i.i.d.). At the same time, it offers a unique opportunity to improve the performance of predictive models on non i.i.d. data, as inferences about one entity can be used to improve inferences about related entities. We discuss this phenomenon in more details in the next Section 2.3.

The work presented in this dissertation differs from the propositional machine learning methods underlying the i.i.d. assumption. It introduces and considers autocorrelation that comes from the existence of relations among examples i.e., examples are related between each other. As mentioned in Section 2.1, this upgrades the setup of classical predictive modeling tasks it considers by introducing a context space D that embraces these relations.

Such relations have already been considered in collective classification (Sen et al, 2008). It exploits dependencies between examples by considering not only the correlations between the labels of the examples and the observed attributes of such examples or the examples in the neighborhood of a particular object, but also the correlations between labels of interconnected (or in a more general case we can say that there exists a reciprocal relation between the examples) examples labels of the examples in the neighborhood. In general, one of the major advantages of collective inference lies in its powerful ability to learn various kinds of dependency structures (e.g., different degrees of correlation (Jensen et al, 2004)).

Alternatively, even greater (and usually more reliable) improvement in classification accuracy can occur when the target values (class labels) of the linked examples (e.g., pages) are used instead to derive relevant relational features (Jensen et al, 2004).

Examples of collective classification can be found in the webpage classification problem where webpages are interconnected with hyperlinks and the task is to assign each webpage with a label that best indicates its topic, and it is common to assume that the labels on interconnected webpages are correlated. Such interconnections occur naturally in data from a variety of applications such as bibliographic data, email networks and social networks.

The relations (dependencies) between examples are the most interesting and challenging problems in Multi-Relational Data Mining (MRDM) (Domingos, 2003). For example, molecules are not independent in the cell; rather, they participate in a complex chains of reactions whose outcomes we are interested in. Likewise, Webpages are best classified by taking into account the topics of pages in their graph neighborhood, and customers' buying decisions are often best predicted by taking into account the influence of their friends and acquaintances.

In contrast, although most of the Inductive Logic Programming (ILP) research has focused on problems where individual examples have relational structure, examples are still independent from each other (Domingos, 2003). The data consists of a collection of facts in first-order logic. Each fact represents a part, and individuals can be reconstructed by piecing together these facts. For instance, an example might be a molecule, with the bonds between atoms as the relational structure, and the task being to predict whether the molecule is a carcinogen.

Inspired by all of these, we position the research presented in this dissertation in between the approaches mentioned above. In particular, we use traditional single table representation of the data, and at the same time we consider the relations as in collective classification. We deal with the correlations between labels of interconnected examples, labels of the examples in the neighborhood, known as autocorrelation (Cressie, 1993). However, we do not use these interconnections for generation of features, but we consider them as a background knowledge related to the target variables. By doing so, we ensure that our models are general and do not depend on a specific application; moreover they are reusable and can be applied for similar tasks within the same application domain.

2.2.2 The Origin of Relations

Next, we give a more detailed view of the foundations of our research by describing the nature and importance of the considered relations. We distinguish between relationships between examples that are inherited from the data itself (explicit relations) and relationships that can be defined artificially in order to substitute the missing natural ones or to enable some learning tasks to be accordingly addressed (implicit relations) and discuss them separately.

Explicit Relations

The nature of the relations (dependencies) between examples that may be considered in a learning task is versatile. Relationships are usually inherited by the problem concerned in the learning task, i.e., they are explicitly given with the problem. In this case they have a natural interpretation related to the modeling problem. Examples of such relations can be found in biological data, where the relations can mean a connection/relation between different organisms; the relations among co-authors and papers in bibliographic data; the popular “friend” relations in social networks, etc.

In practice, these relations may be discussed together and viewed in the form of a network, where the entities are represented by nodes which may be connected with (related to) each other by edges. Each entity is called a node of the network. A number (which is usually taken to be positive) called “weight” is associated with each edge. In a general formulation, a network can be represented as a (weighted and directed) graph, i.e., a set of nodes and a ternary relation which represents both the edges between nodes and the weight associated to each edge.

For instance, the Web can be considered as a network of web-pages, which may be connected with each other by edges representing various explicit relations such as hyperlinks. Social networks can be seen as groups of members that can be connected by friendship relations or can follow other members because they are interested in similar topics of interests. Metabolic networks can provide an insight about genes and their possible relations of co-regulation based on similarities in their expressions level. Finally, in epidemiology networks can represent the spread of diseases and infections.

The weights represent the strength of the relation between the nodes of the network. For example, in hypermedia data, it could be the number of hyperlinks from one page to another; in epidemiology it can be the speed of spread of a disease or infection over a particular geographical region in some time interval or, on Twitter, the fact that a user follows another user (Kwak et al, 2010) (keeping in mind that a user is not obligated to reciprocate followers by following them). In the latter case the relationship is binary ("is following" or not) and expresses a lower amount of information then the former quantitative one which expresses an existence of a relation and gives information about the strength of the relation. These examples, indeed, are related to the case of modeling asymmetric relationships.

In the case of modeling symmetric relationships, the weights can represent the frequency with which a pair of lectures been viewed together or the distance between cities in the same country. The weights can also be given by a binary value indicating whether two proteins interact each other within a biological network, or on Facebook, the "friend" relationship, where everyone who you have claimed as a friend has also claimed you as a friend. This is a two-way reciprocal relationship and it is twice (in both directions) encountered in the further analysis.

Implicit Relations

When the relations (dependencies) between examples come along with the problem concerned in the learning task, we deal with them as they are defined. However, sometimes it is the case that the relationships are not immediately available from the data. In such cases, we need to define them within the context of the learning task that we deal with. These relations are then called *implicit relations*.

There is more than one unique way to define implicit relations. We have the ability to choose the type of the relations that we plan to use and determine their strength, according to our needs and interest. For that purpose, the weights representing the strength of the relations are often computed based on symmetric and nonnegative similarity measures between examples.

Different similarity measures have been defined in almost every scientific field, however one can generally distinguish among distance-based, feature-based and probabilistic similarity measures. The main difference between them is that the perception used in the first two measures is deterministic, whereas for the latter one, it varies over time (probabilistic). We will focus on distance-based similarity (dissimilarity) measures, as a quantitative degree of how far apart two examples are.

Most common distance measures for continuous data are Euclidean distance, Manhattan distance, Chebychev distance, block distance and Minkowski distance, whereas for discrete data are chi-square or phi-square distance (Bishop, 2007). Most popular distance measures for binary data include Euclidean distance, squared Euclidean distance, size difference, pattern difference, variance or shape.

The notion of distance for discrete data is not as straightforward as for continuous data. The key characteristic of discrete data is that the different values that a discrete variable takes are not inherently ordered (Boriah et al, 2008). Thus, it is not possible to directly compare two different discrete values. The simplest way to find similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical.

Overall, the concept of Euclidean distance has prevailed in different fields and has been used as a universal distance measure for all types of data or patterns (Bishop, 2007). The Euclidean distance between examples p and q is defined as the length of the line segment connecting them ($\overline{pq}$). In Cartesian coordinates, if $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ are two points in Euclidean n-space, then the distance from p to q, or from q to p is defined as:

$$d(p,q) = d(q,p) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{2.1}$$

Moreover, when it is necessary to balance out the contributions of each element, a more suitable choice is the weighted Euclidean distance, defined as:

$$d(p,q) = d(q,p) = \sqrt{\sum_{j}^{N} w_j (p_j - q_j)^2} \tag{2.2}$$

where $w_j = 1/s_j^2$ is the inverse of the j-th variance. We think of w_j as a weight attached to the j-th variable: in other words, we compute the usual squared differences between the variables on their original scales, as in the Euclidean distance, but then multiply these squared differences by their corresponding weights.

In addition, Manhattan distance can be used as an alternative to the Euclidean distance (Deza, 2009). Manhattan distance is the distance between two points measured along axes at right angles. It is calculated by summing the (absolute) differences between point coordinates. In a plane with p at (x_1, y_1) and q at (x_2, y_2), it is:

$$d_m(p,q) = d_m(q,p) = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid \tag{2.3}$$

This is easily generalized to higher dimensions. Manhattan distance is often used in integrated circuits, where wires only run parallel to the X or Y axis.

Aside from this, new distance measures (e.g., (Džeroski et al, 2007)) have been recently exploited for a variety of applications encountered in many different fields such as anthropology, biology, chemistry, computer science, ecology, information theory, geology, mathematics, physics, psychology, statistics, etc. In addition, there have been considerable efforts within different fields in finding the appropriate measures among such a plethora of choices because it is of fundamental importance for the analyzed problem.

We consider the examples as nodes in a network. The similarity measures between nodes can be based on the similarity of the node labels or on the distance between examples in the case of spatial, temporal and spatio-temporal data. The network G is represented by an adjacency matrix $\mathbf{W}$ which is sometimes also called the connection matrix. The matrix has entries $w_{ij} > 0$ if there is an edge connecting i to j, and $w_{ij} = 0$ otherwise. We impose $w_{ii} = 0$ and we define the degree of a node u_i as $Deg(u_i) = \sum_j w_{ij}$.

When dealing with network data, the relations are already inherited from the data and we use them as such within the learning task. Often these relations are symmetric and binary indicating that a connection/relation between two nodes exists. Otherwise in the case of spatial, temporal and spatio-temporal data we use a contingency matrix.

A special type of the adjacency matrix $\mathbf{W}$ is the spatial connectivity (contiguity) matrix, typically specifying the connections between locations that are physically adjacent. Specifically, it contains weights based on the spatial proximity of the location of an example i to the examples' locations around i. The basic idea is that the examples close to a specific object have more influence in the estimation of its target value than examples farther away.

The simplest choice to solve this problem is to use a Binary similarity measure:

$$w_{ij} = \begin{cases} 1 & \text{if } d_{lj} < b \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where b is referred to as the bandwidth and d_{ij} is the Euclidean/Manhattan spatial distance between examples o_i and o_j. However, it does not reflect the actual geographical processes very well because it suffers from the problem of discontinuity.

A popular choice to solve this problem is to use the Gaussian-like similarity measure, defined as:

$$w_{ij} = \begin{cases} e^{-\frac{d_{lj}^2}{b^2}} & \text{if } d_{lj} < b \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where b is referred to as the bandwidth and d_{ij} is the Euclidean/Manhattan spatial distance between examples o_i and o_j. If o_i and o_j are at the same location, $w_{ij} = 1$. The weighting of other data will decrease according to a Gaussian-like curve, as the distance between o_i and o_j increases. If o_i and o_j are farther away from each other than the bandwidth b, then they are not considered in the analysis. We refer to this weighting function as "Gaussian".

As an alternative, it is possible to use a discrete weighting function (see Equation 2.6) and a bi-square density function (see Equation 2.7), defined as:

$$w_{ij} = \begin{cases} 1 - \frac{d_{lj}}{b} & \text{if } d_{lj} < b \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$w_{ij} = \begin{cases} (1 - \frac{d_{lj}^2}{b^2})^2 & \text{if } d_{lj} < b \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

which we refer to as "Euclidean" and "Modified", respectively.

Since we already discussed the distances (e.g., Equation 2.1 and Equation 2.3) and the weights (e.g., Equation 2.5, Equation 2.6 and Equation 2.7), we need to focus on the selection of the optimal bandwidth. This problem is tackled in Section 6.2.3 and Section 7.2.3, for the spatial and network data, accordingly.

We would like to note here that there is no clear-cut agreement on the choice of a proper weighting function (or weighting matrix). According to Dubin (1998), there is little agreement regarding the best form for the connectivity matrix and the above described forms which are commonly used. In fact, all of these forms depend on the scale of the data in some way. Moreover, the specification also involves the bandwidth b parameter, which is typically chosen apriori while the weights are generally treated as exogenous. In contrast, Anselin and Bera (1998) argue that the spatial weight matrix should be constructed by a spatial interaction theory, such as the concept of accessibility, instead of simple physical feature of spatial units.

2.2.3 The Use of Relations

In the following subsection, we argue the experimental settings and discuss them along the setting on which existing related approaches are based. We compare these settings and stress their differences.

i) Autocorrelation over the target space

Many machine learning algorithms and statistical methods that deal with the autocorrelation phenomenon take into account the autocorrelation of the input space (descriptive attributes) (see for example (Appice et al, 2009; Malerba et al, 2005)). This is very intuitive, especially in spatial regression studies, where it is common to resample the study area until the input variables no longer exhibit statistically significant spatial autocorrelation.

In order to explicitly take autocorrelation into account, we need to define the spatial/network dimension of the data. For this purpose, in addition to the descriptive space and the target space, it is necessary to add information on the spatial/network structure of the data in order to be able to capture the spatial/network arrangement of the examples (e.g., the coordinates of the spatial examples involved in the analysis or the pairwise distances between them).

A naïve solution would consider both the descriptive and autocorrelation attributes together as input of a learning process. This has already been done in a number of studies (e.g., (Appice et al, 2009)) in different domains. However, this solution would lead to models that would be difficult to apply in the same domain, but in different spatial/network contexts.

Following (Ester et al, 1997), we do not consider spatial/network information in together with descriptive one in the learned models. This limitation of the search space allows us to have more general models, at the price of possible loss in predictive power of the induced models.

In contrast to these studies, we are interested in accounting for autocorrelation related to the target (output) space, when predicting one or more (discrete and continuous, as well as structured) target variables, at the same time.

ii) Autocorrelation as a background knowledge

Autocorrelation can be considered with the learning process in many different ways. Various approaches, such as collective classification and typical network analysis (already mentioned in Section 2.2.1), consider different types of relations to exploit the phenomenon of autocorrelation within the learning process: from the synthesis of in-network features to the propagation of a response across the network.

In most of the collective classification studies (Gallagher et al, 2008), the connections/relations (edges in the network) between the data in the training/testing set are predefined for a particular instance and are used to generate the descriptive information associated to the nodes of the network. In this way, in-network features are created. However, such features can be a limiting factor in the predictive modeling process that can lead the models to lose their generality and possible general applicability.

In typical network studies, the general focus is on exploring the structure of a network by calculating its properties (e.g. the degrees of the nodes, the connectedness within the network, scalability, robustness, etc.). The network properties are then fitted into an already existing mathematical network model or a theoretical graph model (Steinhaeuser et al, 2011). Also in this case the created in-network features are in a tight, inseparable relation to the data which is a limitation toward the generality of the models.

Another limitation of most of the models is that they only consider the cases where training and testing data (nodes) belong to the same network. This means that the prediction phase requires complete knowledge of the network arrangement (e.g., connections to other nodes of the network) of any unlabeled node to be predicted.

In contrast to these studies, in this dissertation, the connections are not in a tight inseparable relation to the data. In fact they relate to the target space and not to the descriptive attributes. Moreover, different types of relations (explicit and implicit) can be used with the same data, as a tool to access the quality of the relational data.

The network setting that we address in this work is based on the use of both the descriptive information (attributes) and the network structure during training whereas, on the use of the descriptive information in the testing phase where we disregard the network structure.

More specifically, in the training phase we assume that all examples are labeled and that the given network is complete. In the testing phase all testing examples are unlabeled and the network is not given. Because of this setting, a key property of the proposed solution is that the existence of the network is not obligatory in the testing phase, where we only need the descriptive information. This can be very beneficial especially in cases where the prediction needs to be made for those examples for which connections to other examples are not known or need to be confirmed.

The setting where a network with some nodes labeled and some nodes unlabeled (Appice et al, 2009) is given, can be mapped to our setting. In fact, we can always use the nodes with labels and the projection of the network on these nodes for training and only the unlabeled nodes without network information in the testing phase.

The existence of the relations is not obligatory for in the testing set. This leads to the creation of general models.

iii) Non-stationary autocorrelation

A limitation of most of the models that represent and reason for autocorrelation is that the methods assume that autocorrelation dependencies are stationary (i.e., do not change) throughout the considered context space (time, space or network) (Angin and Neville, 2008). This means that possible significant variabilities in autocorrelation dependencies throughout the space/network cannot be represented and modeled. However, the variabilities could be caused by a different underlying latent structure of the network (or generally in any dimension) that varies among its portions in terms of properties of nodes or associations between them. For example, different research communities may have different levels of cohesiveness and thus cite papers on other topics with varying degrees. As pointed out by Angin and Neville (2008), when autocorrelation varies significantly throughout a network, it may be more accurate to model the dependencies locally rather than globally.

To overcome this issue, in this work, we develop an approach for modeling non-stationary autocorrelation data, where the autocorrelation is related to the target (output) space.

Since we consider the PCT framework, the tree models obtained by the proposed algorithm allow us to obtain a hierarchical view of the network, where clusters can be employed to design a federation of hierarchically arranged networks. This can turn to be useful, for instance, in wireless sensor networks, where a hierarchical structure is one of the possible ways to reduce the communication cost between the nodes (Li et al, 2007).

Moreover, it is possible to browse the generated clusters at different levels of the hierarchy, where each cluster can naturally consider different effects of the autocorrelation phenomenon on different portions of the network: at higher levels of the tree, clusters will be able to consider autocorrelation phenomenons that are spread all over the network, while at lower levels of the tree, clusters will reasonably consider local effects of autocorrelation.

This gives us the way to consider non-stationary autocorrelation.

2.3 Autocorrelation

The autocorrelation phenomenon takes the central place of discussion in this dissertation. To stress its importance and the role it plays in the predictive modeling process, it is discussed along two orthogonal dimension: the first one considers the type of targets in predictive modeling, whereas the second one focuses on the type of relations considered in the predictive model. First, we discuss the autocorrelation phenomenon in terms of the types of the targets in predictive modeling. Next, we present different types of the autocorrelation phenomenon and describe their specific properties. In particular, we focus on temporal autocorrelation, spatial autocorrelation, spatio-temporal autocorrelation and network autocorrelation.

2.3.1 Type of Targets in Predictive Modeling

In machine learning and data mining, different predictive modeling tasks are recognized according to the type of the response target variables considered in predictive modeling process (an overview of these tasks is presented in Section 2.2.2). Moreover, the type of the targets in the predictive models determines the relations that can exist among the instances included in the predictive modeling process. As already discussed in Section 2.2.2, the distance measures that express the relations are related to the data, i.e., type of the targets in the predictive models.

Besides the classical Euclidean distance, we use a weighted Euclidean distance that reflects the similarity within a hierarchy of classes. In a HMC context, the similarity of class labels at higher levels of the hierarchy is more important than the similarity of class labels at lower levels. This weighted Euclidean distance used in this case is given with Equation 8.1 in Section 8.2.

In particular, the classical Euclidean distance is used for the definition of almost all measures of autocorrelation and a weighted Euclidean distance is used for the definition of the measures of autocorrelation in a hierarchial multi-label classification (HMC). The measures of autocorrelation are defined in Chapter 3.

The choice of the distance measure is an important factor that influences the definition of the measures of autocorrelation, not only in the HMC context, but in all predictive modeling tasks. Euclidean distance is not always the best choice of a suitable distance measure, especially when the targets are structured like time-series data where the Euclidean distance is highly sensible to small distortions in the time axis. However, there exists no autocorrelation measure that does not depend on the Euclidean or Manhattan distance. This opens a possibility of future generalizations of the autocorrelation measure that will make them independent of the distance measure and will make them applicable for different types of targets.

2.3.2 Type of Relations

Autocorrelation occurs in many fields, such as statistics (e.g., correlation between the values of variables that are based on associated aspects), spatial and time-series analysis (e.g., terrain elevations, soil types, and surface air temperatures measures at close locations) and networks (e.g., current financial transactions are related to previous similar transaction), sociology (e.g., social relations affect social influence), web mining (e.g., hyperlinked web pages typically share the same topic) and bioinformatics (e.g., proteins located in the same place in a cell are more likely to share the same function than randomly selected proteins). In these fields, different definitions of autocorrelation are in use depending on the field of study which is being considered and not all of them are equivalent.

The cross-correlation of an attribute with itself (Cressie, 1993) is typically referred to as autocorrelation and this is the most general definition found in the literature. Specifically, in statistics, autocorrelation is generically defined as the cross-correlation between the attribute of a process at different points in time (Epperson, 2000). In time-series analysis, temporal autocorrelation is defined as the correlation among timestamped values due to their relative proximity in time (Epperson, 2000).

In spatial analysis, spatial autocorrelation has been defined as the correlation among data values, which is strictly due to the relative location proximity of the examples that the data refer to. It is justified by the Tobler's (Legendre, 1993) first law of geography according to which "everything is related to everything else, but near things are more related than distant things". In network studies the autocorrelation is defined by the homophily's principle as the tendency of nodes with similar values to be linked with each other (McPherson et al, 2001).

In general, the complexity of the different definitions of autocorrelation depends on the number of dimensions that they consider. For example, temporal autocorrelation is the simplest form of autocorrelation over one dimension, i.e., time. This is also often used in statistics, signal processing and time series analysis. Spatial and network autocorrelation are more complex forms of autocorrelation, since they consider two, three or more dimensions, i.e., are multi-dimensional functions.

Historically, the concept of autocorrelation was first established in stochastic processes, as the correlation between paired values of a random or stochastic process $x(n)$, indexed with $n = 0, \pm 1, \pm 2, ...$, taken at usually constant intervals that indicates the degree of periodicity of the function. A stochastic process Mitsa (2010) is a collection of random variables defined on a given probability space, indexed by the time variable t, where t varies over an index set T. Moran (1950) was the first to define this concept for one and two dimensional processes.

The simplest case is that of a stationary stochastic process in one (time) dimension. It is a stochastic process whose joint probability distribution does not change when shifted in (time) dimension. Consequently, parameters such as the mean

$$\mu(n) = E[x(n)] \tag{2.8}$$

and the variance

$$\sigma_x(n)^2 = E[(x(n) - \mu(n))^2] \tag{2.9}$$

also do not change over this dimension. A stationary stochastic process is fully described by the joint probability density function of (the usually normally distributed) observations. In this context, one can always estimate the autocovariance function that indicates how fast a signal can change in (time) dimension.

The *autocovariance function* of x_n is the covariance between two observations x_n and x_{n+k} of a such process, defined as:

$$r(k) = cov(x_n, x_{n+k}) = E[(x_n - \mu)(x_{n+k} - \mu)] \tag{2.10}$$

It measures the covariance between pairs of variables at a distance or time lag k, for all different values of k, and can be normalized to give the *autocorrelation function* $\rho(k)$:

$$\rho(k) = \frac{r(k)}{\sigma_x^2} = \frac{E[(x_n - \mu)(x_{n+k} - \mu)]}{\sigma_x^2} \tag{2.11}$$

The value for the autocorrelation function at lag 0 is 1. This property also follows from the definition of stationarity where the correlation should be only a function of the time lag between two observations; the lags $-k$ and k are equal in that respect. Thus, the autocorrelation function is symmetrical ($\rho(-k) = \rho(k)$) about the origin where it attains its maximum value of one ($\|\rho(k)\| \leq 1$).

Equation 2.11 gives the autocorrelation function $\rho(k)$ for continuous processes (Moran, 1950). However, Moran (1950) defined it also for discrete processes by discretizing the continuous process, i.e., by considering the values of $x(n)$ only at discrete values of $k(= 0, \pm 1, \pm 2, ...)$. Such a process would have an autocorrelation function $\rho(k)$:

$$\rho(k) = \frac{r(\|k\|)}{\sigma_x^2}, k(= 0, \pm 1, \pm 2, ...) \tag{2.12}$$

The autocorrelation function $\rho(k)$ is very similar to the normalized covariance between two random variables X and Y ($cov(X,Y) = E[(X - \mu_x)(Y - \mu_y)]$), i.e., the *correlation coefficient* which is defined as the ratio of the covariance of the variables X and Y and the product of their standard deviations:

$$CC_{x,y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_x)(Y - \mu_y)]}{E[(X - \mu_x)]E[(Y - \mu_y)]} \tag{2.13}$$

where E is the expected value operator and μ_x is the mean of the variable X.

When the autocorrelation function $\rho(k)$ is normalized by subtracting the mean and dividing by the variance, it is referred to as the *autocorrelation coefficient*, given as:

$$AC(k) = \frac{cov(X_n, X_{n+k})}{\sigma_X^2} = \frac{E[(X_n - \mu_X)(X_{n+k} - \mu_X)]}{E[(X_n - \mu_X)]} \tag{2.14}$$

In the case of stationary stochastic processes in two dimensions, we take two parameters t and u to correspond to these dimensions and obtain two dimensions:

$$\rho(p,q) = cor\{x(t,u), x(t+p, u+q)\} = E[x(p^2 + q^2)^{1/2}] \tag{2.15}$$

Generalizing the Moran's definition of autocorrelation (Moran, 1950) of a random variable x_n to match n dimensions, we deal with a n-dimensional vector and the autocorrelation function is expressed as the covariance between the values $x_d = (x_{1,d}, ..., x_{n,d})$ and $x_{d'} = (x_{1,d'}, ..., x_{n,d'})$, where $\tau = d' - d$ is the total shift (distance or difference) between the two vectors x_d and $x_{d'}$ along each of the n dimensions:

$$\rho(\tau) = \frac{E[(X_d - \mu_X)(X_{d'} - \mu_X)]}{E[(X_d - \mu_X)]} \tag{2.16}$$

If the process is stationary, the denominator can be seen as the standard deviation of x, which is assumed to be constant at all times. For processes that are not stationary, the above presented autocorrelation functions will also depend on n. This means that for a one dimensional continuous non-stationary process, the autocorrelation function at lag τ is defined as:

$$\rho(\tau) = \int_{-\infty}^{+\infty} x(n)x^*(n-\tau)dn \tag{2.17}$$

For one dimensional discrete non-stationary processes, the autocorrelation function at lag τ is defined as:

$$\rho(\tau) = \sum_n x_n x_{n-\tau} \tag{2.18}$$

Multi-dimensional autocorrelation functions are defined similarly. For three dimensional discrete non-stationary process, the autocorrelation function is defined as:

$$\rho(\tau, k, l) = \sum_{n,q,r} x_{n,q,r} x_{n-\tau,q-k,r-l} \tag{2.19}$$

The most relevant properties of the autocorrelation (Moran, 1950) include:

- A fundamental property of the autocorrelation is symmetry.
- The autocorrelation of a periodic function is itself periodic, with the same period.
- The autocorrelation of the sum of two completely uncorrelated functions (the cross-correlation is zero for all) is the sum of the autocorrelations functions.
- Since autocorrelation is a specific type of cross-correlation, it maintains all the properties of cross-correlation, such as sampling.

In the remainder of this Section, we proceed with the presentation of the different types of autocorrelation (temporal, spatial, spatio-temporal and network autocorrelation) and the description of their specific properties.

Temporal Autocorrelation

Before we introduce this special type of autocorrelation, let us define some basis concepts. A *time lag* is the time period between two observations. For example, in discrete time we have a lag of 1 between Y_t and Y_{t-1}, and a lag of 2 between Y_t and Y_{t-2}. Observations can also be lagged forward, for example we have a lag of 1 between Y_t and Y_{t+1}.

A sequence of data points, measured at successive time instants spaced at uniform time intervals, is known as a *time series*. Time series appear naturally in a variety of different domains, from engineering to scientific research, finance and medicine. Time series data have a natural temporal ordering. Examples of time series are the daily closing value of the Dow Jones index or the annual flow volume of the Nile River at Aswan.

Quite often, successive observations of time series are correlated. This is mostly a positive correlation, in the sense that large values follow large values and small values follow small values. This

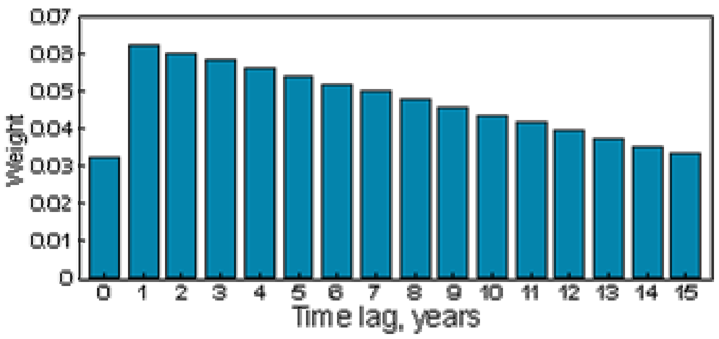


Figure 2.2: *Temporal autocorrelation.* An example of the effect of the temporal autocorrelation. Figure taken from (Sharov, 1997).

phenomenon is known as *temporal autocorrelation*. It is defined as the correlation among data values, which is strictly due to their relative time proximity (Mitsa, 2010).

As already mentioned in Section 2.3.2, the stationary temporal autocorrelation can be expressed as the covariance between two observations x_n and x_{n+k} at time lag k, and defined as:

$$Temporal\_AC(k) = \frac{cov(x_n, x_{n+k})}{\sigma_x^2} = \frac{E[(x_n - \mu_x)(x_{n+k} - \mu_x)]}{E[(x_n - \mu_x)]} \tag{2.20}$$

If the process is stationary, the denominator can be seen as the standard deviation of x, which is assumed to be constant at all time points.

An example of temporal autocorrelated variable is presented in Figure 2.2. The figure presents the periodicity in food weight supply in a fifteen-year period of time and is taken from (Sharov, 1997). The values of the weight supply follow some trend in the data and these values are more similar on shorter time interval than on longer intervals.

Temporal autocorrelation is very common in ecological and environmental, as well as financial studies. For example, the likelihood of tomorrow being rainy is greater if today is rainy than if today is dry. Moreover, weather conditions are highly autocorrelated within one year due to seasonality. A weaker correlation exists between weather variables in consecutive years. Similar example is the annual food supply for a supermarket or the daily financial markets exchange value of the most important financial indexes.

Temporal autocorrelation refers to the correlation of a time series with its own past and future values and it is also sometimes called "lagged correlation" or "serial correlation". In a time series, positive autocorrelation might show up as unusually long runs, or stretches, of several consecutive observations above or below the mean, whereas negative autocorrelation might show up as an unusually low incidence of such runs.

Positive temporal autocorrelation might be considered a specific form of "persistence", a tendency for a system to remain in the same state from one observation to the next. For example, the likelihood of tomorrow being rainy is greater if today is rainy than if today is dry. In contrast, negative autocorrelation is characterized by a tendency for positive trends to follow negative ones, and vice versa.

Temporal autocorrelation complicates the application of statistical tests by reducing the number of independent observations. It can also complicate the identification of significant covariance or correlation between time series (e.g., precipitation with a tree-ring series).

Frequent causes of temporal autocorrelation (Fortin and Dale, 2005) are misidentifications of the following types:

- Omitted variables
- Structural breaks in time-series when we see an unexpected shift
- Lags (yesterday impacts today)
- Over or under differencing the data when transforming parts of a time-series into stationary time-series

Temporal autocorrelation is the simplest form of autocorrelation over one dimension and one direction. However, one complication arises when observations in time are not uniformly spaced. This happens mainly when data were not collected at uniform intervals through time, or a data generating process does not produce values that are equally spaced through time. In this case, usually one assumes that observations in time are piece-wise (or locally) stationary.

Spatial Autocorrelation

Spatial autocorrelation is a very general statistical property of ecological variables observed across geographic space. *Spatial autocorrelation* is defined as a property of random variables taking values, at pairs of locations a certain distance apart, that are more similar (positive autocorrelation) or less similar (negative autocorrelation) than expected for pairs of observations at randomly selected locations (Legendre, 1993).

As already mentioned in Section 2.3.2, spatial autocorrelation is actually correlation among values of a variable strictly due to their relative location proximity. As a location can be defined with two/three spatial coordinates, spatial autocorrelation can occur in two/three dimensions.

For a continuous variable X, measured at locations at a distance (Δ_a, Δ_b), the spatial autocorrelation can be expressed as:

$$Spatial\_AC(\Delta_a, \Delta_b) = \frac{E[(X_{a,b} - \mu_x)(X_{a+\Delta_a, b+\Delta_b} - \mu_x)]}{E[(X_{a,b} - \mu_x)]} \tag{2.21}$$

where $d = \sqrt{(\Delta_a)^2 + (\Delta_b)^2}$ is the Euclidean distance between the values of the variable X at the two locations, E is the expected value operator and μ_x is the mean of the variable X.

Positive spatial autocorrelation means that geographically nearby values of a variable tend to be similar: high values tend to be located near high values, medium values near medium values, and low values near low values. Consequently, positive spatial autocorrelation has all similar values appearing together. In contrast to positive spatial autocorrelation, negative spatial autocorrelation has not so similar values appearing in close association. Finally, Zero spatial autocorrelation means geographically random phenomena and chaotic distribution of geographical data. Informally, spatial positive (negative) autocorrelation occurs when the values of a given property are highly uniform (different) among similar spatial examples in the neighborhood.

In geography, spatial autocorrelation is justified by Tobler's (Tobler, 1970) first law of geography, according to which "everything is related to everything else, but near things are more related than distant things". This means that by picturing the spatial variation of some observed variables in a map, we may observe regions where the distribution of values is smoothly continuous with some boundaries possibly marked by sharp discontinuities.

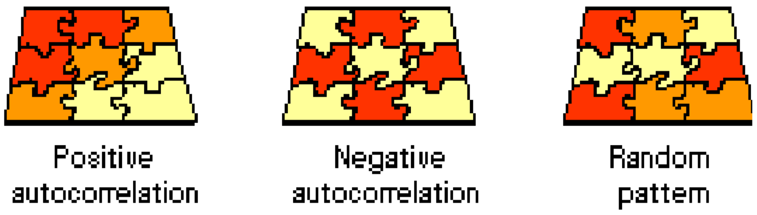


Figure 2.3: *Spatial autocorrelation.* An example of the effect of the spatial autocorrelation on an observed variable in a map. Positive autocorrelation, negative autocorrelation and a random pattern. The different colors indicate different values of the observed variable.

An example of the effect of the spatial autocorrelation on an observed variable in a map is presented in Figure 2.3. The Figure shows three different cases in terms of the spatial autocorrelation phenomenon: positive autocorrelation, negative autocorrelation and a random pattern. The different colors indicate different values of the observed variable. In the case of positive spatial autocorrelation, nearby values of the variable presented in the map have the same color, i.e., geographically nearby values of a variable tend to be similar, whereas in the case of negative spatial autocorrelation, nearby values of the variable presented in the map have not so similar colors, i.e., geographically nearby values of a variable tend to be not so similar. Finally, in the case of zero spatial autocorrelation, the distribution of different colors on the map is random, i.e., the Figure 2.3 shows a geographically random phenomena.

Demographic and socio-economic characteristics like population density and house price are good examples of variables exhibiting positive spatial autocorrelation. Neighborhoods tend to be clusters of households with similar preferences. Families tend to organize themselves in a way that concentrates similar household attributes on a map-creating positive spatial autocorrelation amongst many variables-with government policies and activities, such as city planning and zoning, reinforcing such patterns. Frequently, we observe positive spatial autocorrelation at smaller distances, and negative spatial autocorrelation at greater distances.

Commonly, we tend to model positive spatially autocorrelated data because of the way this phenomena are geographically organized. However, the phenomenon of spatial autocorrelation occurs not only in geographical data, but also in data which are not geo-referenced, although a spatial dimension still exists. For instance, in bioinformatics, proteins located in the same part of a cell are more likely to share the same function than randomly selected proteins. Another example are sentences in text: sentences that are in close proximity are generally more semantically related than distant ones. Moreover, in general, nature is autocorrelated. For example, species are distributed non-randomly across a wide range of spatial scales. In this case, the occurrences of species in multiple cells or plots generate the autocorrelation in geographical data and this is not related to data quality problems.

Therefore, there are two primary reasons of autocorrelation (Dubin, 1998):

- it indexes the nature and degree to which conventional statistical inferences are compromised when non-zero a spatial autocorrelation is overlooked.
- the measurement of spatial autocorrelation describes the overall pattern across a geographical landscape, supporting spatial prediction and allowing detection of striking deviations.

The causes of spatial autocorrelation depend on the specific domain we are dealing with. For instance, in ecological and environmental modeling, where data are typically geo-referenced, four factors are particularly common (Legendre, 1993; Legendre et al, 2002):

- Biological processes of speciation, extinction, dispersal or species interactions are typically distance-related;

- Non-linear relationships may exist between species and environments, but these relationships may be incorrectly modeled as linear;
- Classical statistical modeling may fail in the identification of the relationships between different kinds of data without taking into account their spatial arrangement (Besag, 1974);
- The spatial resolution of data should be taken into account: Coarser grains lead to spatial smoothing of data.

Spatial autocorrelation has many interpretations. We present some of them (Dubin, 1998):

- As a nuisance parameter, inserted in a model specification because its presence is necessary for a good description but it is not the topic of interest.
- As a self-correlation, arising from the geographical context within which attribute values occur.
- As a map pattern, viewed in terms of trends, gradients or mosaics across a map.
- As a diagnostic tool for proper sampling design, model misspecification, nonconstant variance or outliers.
- As a redundant (duplicate) information in geographical data, connected to the missing values estimation, as well as to notation of effective sample size and degrees of freedom.
- As a missing variables indicator/surrogate, popular in spatial econometrics (Anselin, 1988).
- As an outcome of areal unit demarcation in statistical analysis.

Consequently, when analyzing spatial data, it is important to check for autocorrelation. If there is no evidence of spatial autocorrelation, then proceeding with a standard approach is acceptable. However, if there is evidence of spatial autocorrelation, then one of the underlying assumptions of your analysis may be violated and your results may not be valid.

Spatial autocorrelation is more complicated than temporal autocorrelation as it can occur in any direction. Moreover, the phenomenon of existing spatial and temporal autocorrelation at the same time has even higher complexity.

Spatio-Temporal Autocorrelation

In nature, it is a common case that data is not only affected by the phenomenon of spatial autocorrelation, but also by the phenomenon of temporal autocorrelation at the same time. In that case, we consider **spatio-temporal autocorrelation**, as a special case of autocorrelation.

Spatio-temporal autocorrelation is a property of a random variable taking values, at pairs location a certain distance apart in space and time, that a more similar or less similar than expected for pairs of observations at random selected locations and times. It is the actual correlation among values of a variable strictly due to their relative location proximity and time proximity.

For a continuous variable X, measured at locations at a distance (Δ_a, Δ_b), the spatio-temporal autocorrelation can be expressed as:

$$SpatioTemporal\_AC(\Delta_a, \Delta_b, \tau) = \frac{E[(X_{a,b,k} - \mu_x)(X_{a+\Delta_a, b+\Delta_b, k+\tau} - \mu_x)]}{E[(X_{a,b,k} - \mu_x)]} \tag{2.22}$$

where $d = \sqrt{(\Delta_a)^2 + (\Delta_b)^2}$ is the spatial distance between the values of the variable X at the two locations, τ is the temporal lag, E is the expected value operator and μ_x is the mean of the variable X.

The simplest case of spatio-temporal autocorrelation is the short-range positive spatio-temporal autocorrelation where samples are more similar when they are closer together in space or in time. Animals close in space are more likely to suffer mortality close in time due to some disease that is spreading. More complexity can be found in cases involving cyclic behavior, such as diurnal migration and vertical migration of zooplankton, where autocorrelation will be positive at short space and time lags, then negative over short space and longer time lags and then positive again over even longer time lags. For example, the weather conditions in one year time period within a geographical region that consists of many subregions are the same time highly (temporally) autocorrelated due to the seasonality and highly (spatially) autocorrelated due to their spatial vicinity.

For example, the slowly evolving and moving low pressure systems in the atmosphere might impart persistence to daily rainfall. Or the slow drainage of groundwater reserves might impart correlation to successive annual flows of a river. Or stored photosynthates might impart correlation to successive annual values of tree-ring indices.

In ecology, the concept of spatio-temporal pattern usually describes that certain vegetation types tended to occur close together in space and time. One factor that contributes to this phenomenon is the clonal nature of some plants. Clonal growth forms are often described based on the spatial pattern of the ramets, which is related to patterns of establishment in space and time.

Ecologists are interested in a variety of spatio-temporal association patterns involving sequences of events abstracted from the measurement values of ecological variables at various spatial locations. The most common types of patterns (Fortin and Dale, 2005) are:

- Intra-zone non-sequential patterns - relationships among events in the same region or zone, ignoring the temporal aspects of the data.
- Inter-zone non-sequential pattern - relationships among events happening in different regions or zones, ignoring temporal aspects of the data.
- Intra-zone sequential pattern - temporal relationships among events occurring within the same region or zone.
- Inter-zone sequential pattern - temporal relationships among events occurring at different spatial locations.

An example of the effect of the spatio-temporal autocorrelation on an observed variable at two different times is presented in Figure 2.4. The Figure represents the US county votes cast in the 1980 and 1984 presidential elections (Pace and Barry, 1997). The values of the votes cast change thought the United States counties. The different colors indicate different values of the observed variable. The values are more similar on shorter spatial distances than on longer distances. Moreover, the values are more or less similar at the same locations, in the left and right image, representing the map of the same variable in two consequent election years 1980 and 1984.

Spatio-Temporal autocorrelation has higher complexity than spatial and temporal autocorrelation, as it can occur in both spatial and temporal dimensions and it can have any direction (therefore, its specific definition can be obtained from the general autocorrelation definition, given with the Equation 2.16, for that particular multi-dimensional case). Because of this and also because of its application specific treatment, it has been generally addressed as a special case of spatial or temporal autocorrelation and dealt within the spatial or temporal context exclusively.

Network Autocorrelation

Networks have become ubiquitous in several social, economical and scientific fields, ranging from the Internet to social sciences, biology, epidemiology, geography, finance, and many others. Indeed, re-

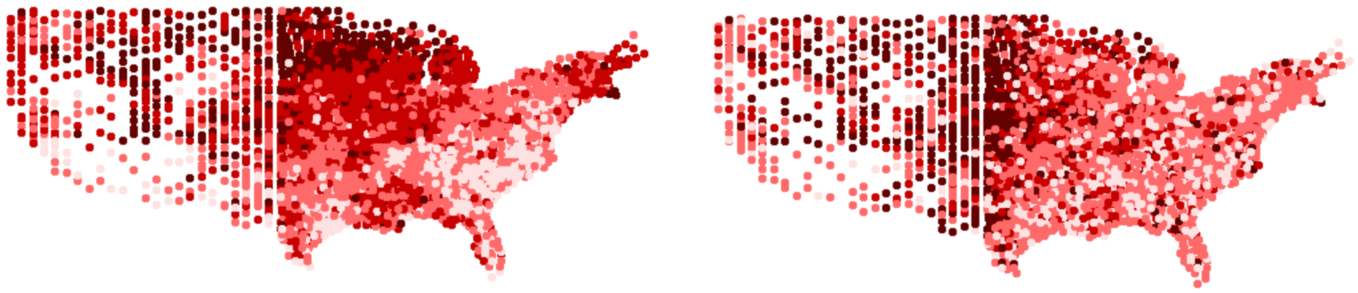


Figure 2.4: *Spatio-temporal autocorrelation.* An example of the effect of the spatio-temporal autocorrelation on an observed variable in a map. The different colors indicate different values of the observed variable. The different maps represent the observed variable in 1980 and 1984.

searchers in these fields have proven that systems of different nature can be represented as networks (Newman and Watts, 2006). For instance, the Web can be considered as a network of web-pages, which may be connected with each other by edges representing various explicit relations, such as hyperlinks. Social networks can be seen as groups of members that can be connected by friendship relations or can follow other members because they are interested in similar topics of interests. Metabolic networks can provide insight about genes and their possible relations of co-regulation based on similarities in their expressions level. Finally, in epidemiology, networks can represent the spread of diseases and infections.

Regardless of where we encounter them, networks consist of entities (nodes), which may be connected to each other by edges. The nodes in a networks are generally of the same type and the edges between nodes express various explicit relations. Information on the nodes is provided as a set of properties (attributes) whose values are associated to each node in the network. The edges reflect the relation or dependence between the properties of the nodes.

Network (relational) autocorrelation is defined as the property that a value observed at a node depends on the values observed at neighboring nodes in the network (Doreian, 1990). It can be seen as a special case of spatial (multi-dimensional and multi-directional) autocorrelation, but with different measures of relationship applied. In social analysis, it is recognized by the homophily's principle, that is the tendency of nodes with similar values to be linked to each other (McPherson et al, 2001).

Figure 2.5 shows an example of a data network where different colors represent different node labels. We can observe that there is a higher tendency of nodes having the same color to be connected to each other, than the tendency of nodes having different colors to be connected to each other.

Formally, a network is a set of entities connected by edges. Each entity is called a *node* of the network. A number (which is usually taken to be positive) called *weight* is associated with each edge. In a general formulation, a network can be represented as a (weighted and directed) graph, i.e., a set of nodes and a ternary relation which represents both the edges between nodes and the weights associated to those edges. The network is represented by an adjacency matrix $\mathbf{W}$, whose entries are positive ($w_{ij} > 0$) if there is an edge connecting i to j, and equal to 0 ($w_{ij} = 0$), otherwise. In practice, when the original data come in the form of a network, the weight w_{ij} represents the strength of the connection from one node to another.

Generalizing the Moran's definition of spatial autocorrelation (Moran, 1950) of a random variable x_n to match n dimensions, we deal with a n-dimensional vector and the autocorrelation function is expressed as the covariance between the values $x_d = (x_{1,d}, \ldots, x_{n,d})$ and $x_{d'} = (x_{1,d'}, \ldots, x_{n,d'})$, where $\tau = d' - d$ is the

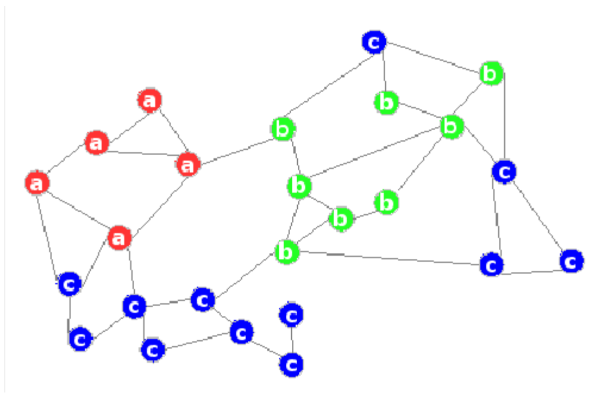


Figure 2.5: *Network autocorrelation.* Different nodes are given in different colors depending on the nodes' labels.

total shift between the two vectors x_d and $x_{d'}$ along the network:

$$Network\_AC(\tau) = \frac{E[(X_d - \mu_X)(X_{d'} - \mu_X)]}{E[(X_d - \mu_X)]} \tag{2.23}$$

In social networks, homophily is defined as the tendency of individuals to associate and bond with similar others (*friendship*). Actually, homophily shows that people's social networks are homogeneous with regard to many sociodemographic, behavioral, and intra-personal characteristics. Homophily effects among *friends* have demonstrated their importance in marketing (Chuhay, 2010). Moreover, homophily is the hidden assumption of recommender systems, although it veers away from how people are socially connected to how they are measurably similar to each another.

In the context of *Twitter*, as special type of a social network, homophily implies that a *twitterer* follows a *friend* because he/she is interested in some topics the *friend* is publishing and the *friend* follows back the *twitterer* if he/she shares similar topics of interest. This is due to the fact that "a contact between similar people occurs at a higher rate than among dissimilar people" (Weng et al, 2010). Recently, (Kwak et al, 2010) investigated homophily in two contexts: geographic location and popularity. They considered the time zone of a user as an approximate indicator for the location of the user and the number of followers as a measure for user's popularity. Among reciprocated users they observed some level of homophily.

Frequent causes of network (relational) autocorrelation (Gujarati, 2002) are misidentifications of the following types:

- Omitted variables
- Fitting an inappropriate functional form, which can cause rise of autocorrelation in the errors of the model
- Cobweb phenomenon (it takes time to adjust to a change in policy)
- Manipulation of data (creating subsets over time resulting in a systematic pattern)
- Data transformations

2.4 Summary

In this chapter we defined the problem that we consider, i.e., learning when examples are related among each other.

First, in Section 2.1 we defined the predictive modeling tasks that we deal with. In contrast to classical predictive modeling tasks that underlie on the i.i.d assumption, we take into account data where examples are related among each other. Beside the typical single and multi-target classification and regression task, we are also concerned with the predictive modeling tasks where the outputs are structured.

In addition, in Section 2.2 we explained the relational aspects taken into account within the defined predictive modeling tasks. In Section 2.2.1 we focused on the different types (explicit and implicit) of relations that we consider. Moreover, in Section 2.2.2 we described their origin, in Section 2.2.3 we explained their use and importance within our experimental setting. Furthermore, we compared our experimental setting to other existing ones.

Finally, in Section 2.3 we introduced the concept of autocorrelation. In particular, we presented the definitions of autocorrelation, from the most simple one regarding the autocorrelation of a continuous /discrete variable over one dimension to the case of two and more dimensions. Moreover, we provided a generic multi-dimensional definition of autocorrelation that can be used to define different forms of autocorrelation in deferent areas. Furthermore, we defined the different forms of autocorrelation that we consider and discussed them along two orthogonal dimension.

In Section 2.3.1 we presented the forms of autocorrelation according to the type (discrete, continuous and structured) of targets in predictive modeling, whereas in Section 2.3.2 we focused on the forms of autocorrelation according to the type (temporal, spatial, spatio-temporal and network autocorrelation) of relations that have been considered in the predictive model.

3 Existing Autocorrelation Measures

In this chapter, we give an overview of the measures of autocorrelation. We consider the existing measures of autocorrelation, however we also define some new ones.

The measures of autocorrelation are divided according to the different forms of autocorrelation that we consider: temporal, spatial, spatio-temporal and network autocorrelation. The overview of the measures starts from the measures of temporal autocorrelation, as the simplest form of autocorrelation. Then, we present the measures of spatial and spatio-temporal autocorrelation. Next, we present the measures of network (relational) autocorrelation

Orthogonally, we distinguish between measures of autocorrelation for the classification and regression task. In addition, we define new measures of autocorrelation for the HMC tasks.

3.1 Measures of Temporal Autocorrelation

Computing the autocorrelation for a time series can be used to test whether the time series is random (the autocorrelation between successive observations is very small) or to find the most important lags (usually the highest autocorrelation is noted for the first lag). We present the most common measures of temporal autocorrelation, defined in Section 2.3 as the simplest form of autocorrelation. We also present some statistical tests for the presence of temporal autocorrelation in the residuals of the predictive models. In addition, we distinguish between measures of autocorrelation for the classification and regression task.

3.1.1 Measures for Regression

For the regression task, a measure of temporal autocorrelation and several statistical tests (including the Durbin-Watson statistic, the Q-Statistic and the Breusch-Godfrey LM test) for its presence in the residuals of the predictive models are used.

Moran's *I*

According to Moran (1950), temporal autocorrelation for time lag m is defined as:

$$AC_m = \frac{\sum_i^{N-m} w_{ij}(Y_i - \overline{Y})(Y_{i+m} - \overline{Y})}{\sum_i^N (Y_i - \overline{Y})^2} \tag{3.1}$$

where Y_1, Y_2,..., Y_N is a time series of observations, w_{ij} is a weight describing the temporal relationship of the observations i and j (already discussed in Section 2.2.2), $\overline{Y}$ is the mean of the observations, and m is the lag considered.

The values of the Moran's I measure (AC_m) for temporal autocorrelation are in the interval [-1,1], where -1 means high negative temporal autocorrelation, 0 means no temporal autocorrelation (random distribution of the data) and 1 means high positive temporal autocorrelation. The autocorrelation in this case is perceived over one dimension. It can also be seen as a simplified version of the formula of the Global Moran's I (see Equation 3.4) used in spatial statistics.

The Durbin-Watson statistic

The autocorrelation between the points in a time-series of observations occurs when the residual error terms of the same variable at different times are correlated. The Durbin-Watson statistic (Durbin and Watson, 1950) is used to detect the presence of autocorrelation in the residuals (prediction errors) from a regression analysis of time-series data and is defined as:

$$DW_Y = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T}(e_t)^2} \tag{3.2}$$

where $e_t = \hat{Y}_t - Y_t$ is the residual associated with the observation Y_t at time t and T is the number of temporal units considered. The values of *DW* typically lie in the interval [0,4]. The value of *DW* is close to 2 if the errors are uncorrelated, less than 2 if the errors are positively correlated, and larger than 2 if the errors are negatively correlated.

This autocorrelation of the residuals may not be a very good estimation of the autocorrelation of the true errors, especially if there are few observations and the independent variables have certain patterns. Positive autocorrelation of the errors generally tends to overestimate the error variance, so confidence intervals are too narrow and true null hypotheses are rejected with a higher probability than the stated significance level. On the other hand, negative autocorrelation of the errors generally tends to make the estimate of the error variance too large, so confidence intervals are too wide and the power of significance tests is reduced.

There are several main limitations of the *DW* statistics as a test of detecting the presence of temporal autocorrelation. First, the distribution of the *DW* statistics under the null hypothesis depends on the data. The usual approach to handling this problems is to place bounds on the critical region, creating a region where results are inconclusive. Second, if there are lagged dependent variables on the right-hand side of the regression, the *DW* test is no longer valid. Third, one may only test the null hypothesis of no temporal autocorrelation against the alternative hypothesis of first-order (at time lag one) temporal autocorrelation. Overall, the major limitation of this test is that there are zones of indecision in which it is not possible to decide whether (first order) temporal correlation exists.

The *Q-Statistic* and *Breusch-Godfrey LM test* overcome the limitations of the *DW* statistic and are preferred in most applications. They are described in the following subsections.

The Q-Statistic

The Ljung-Box Q-Statistic (Box et al, 1994) is also a statistical test for temporal autocorrelation in the errors of a regression model. The Ljung-Box Q-Statistic assesses the null hypothesis that a series of residuals exhibits no autocorrelation for a fixed number of lags L, against the alternative that some autocorrelation coefficient $\rho(k)$, $k = 1,...,L$, is nonzero. The test statistic is

$$Q - Statistic = T(T+2)\sum_{k=1}^{L}\left(\frac{\rho(k)^2}{T-k}\right) \tag{3.3}$$

where T is the sample size, L is the number of autocorrelation lags, and $\rho(k)$ is the sample autocorrelation at lag k. Under the null hypothesis, the asymptotic distribution of Q-Statistic is chi-square with L degrees of freedom.

The range of input lags affects the power of the test. If L is too small, the test will not detect high-order autocorrelation; if it is too large, the test will lose power when a significant correlation at one lag is washed out by insignificant correlations at other lags. The default value of $L = min[20, T-1]$ for the input lags is suggested by Box et al (1994). The degrees of freedom are reduced by the number of estimated coefficients excluding constants, when the vector of residuals is obtained by fitting a model to the data.

Besides the use of the Q-Statistic as a statistical test for temporal autocorrelation in the errors in a regression model, it can also be used to identify conditional heteroscedasticity (ARCH effects) by testing squared residuals.

The BG/LM test

The Breusch-Godfrey (*BG*) or Lagrange Multiplier (*LM*) Test (Godfrey, 1978) is a statistical test for temporal autocorrelation of the errors of a regression model. The test is useful because it accounts for *i)* lagged dependent variables, *ii)* higher order autoregressive processes, as well as single or higher order moving average processes. The basic idea is to make use of the residuals from the model being considered in a regression analysis. A test statistic is derived from these. The null hypothesis is that there is no temporal autocorrelation of any order up to order p.

The BG/LM test actually tests the presence of temporal autocorrelation that has not been included in a proposed regression model structure. If such autocorrelation is present, this would mean that incorrect conclusions would be drawn from other tests or that sub-optimal estimates of model parameters are obtained if it is not taken into account. This type of structure is common in econometric models.

The test is more general than the Durbin-Watson statistic (Durbin and Watson, 1950), which is only valid for non-stochastic regressors and for testing the possibility of a first-order autoregressive model for the regression errors. The test has none of these restrictions, and is statistically more powerful than *DW* statistic (Godfrey, 1978). However, a drawback of the *BG* test is that the number of lags cannot be determined apriori.

3.1.2 Measures for Classification

According to Moran (1950), temporal autocorrelation for time lag m can also be defined for a discrete variable Y. In this case, a discretization is needed. One possible way is to transform the discrete variable Y into q $0/1$ binary variables $Y_1, Y_2, \ldots, Y_q$ and calculate the temporal autocorrelation for time lag m as an average of q binary variables.

3.2 Measures of Spatial Autocorrelation

Several spatial autocorrelation statistics are the most commonly encountered measures of autocorrelation in the literature. In general, one can distinguish between global and local measures of spatial autocorrelation. Global measures of spatial autocorrelation estimate the overall degree of spatial association across the whole data set (spatial region or area), whereas the local ones estimate the local degree of spatial association in the data. We describe the most relevant spatial autocorrelation measures for both continuous and discrete variables.

3.2.1 Measures for Regression

Most measures for spatial autocorrelation are defined for a continuous variable. We give an overview of the existing spatial autocorrelation measures for continuous variables, i.e., for the regression task, including both global and local measures of spatial autocorrelation.

Global Moran's *I*

In spatial data analysis, one of the oldest measures of spatial autocorrelation is Global Moran's *I* (Moran, 1950). This autocorrelation measure is the most common measure of spatial autocorrelation and is still a de facto standard for determining spatial autocorrelation. It is usually applied to continuous variables that have a spatial dimension (usually defined by spatial coordinates of points or objects or distances between them) associated to them.

The Global Moran's *I* (Moran, 1950) is defined as:

$$I_Y = \frac{N \sum_i \sum_j w_{ij}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_i \sum_j w_{ij} \sum_i (Y_i - \overline{Y})^2} \tag{3.4}$$

where N is the number of spatial examples indexed by i and j; Y_i and Y_j are the values of the variable Y for the examples o_i and o_j, respectively; Y is the variable of interest; $\overline{Y}$ is the overall mean of Y; and $w_{ij}, i, j = 1, \ldots, N$ are the values of a $N \times N$ matrix of spatial weights, i.e., $W = \sum_{i,j} w_{ij}$. The matrix of spatial weights W can be defined with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 as already discussed in Section 2.2.2.

Global Moran's *I* can also be seen as a measure of the overall clustering of the data and presented as:

$$I_Y = \frac{N \sum_i \sum_j w_{ij} Z_i Z_j}{S_0 \sum_i Z_i^2} \tag{3.5}$$

where $Z_i = Y_i - \overline{Y}$ is the deviation of the variable Y with respect to the mean and $S_0 = \sum_i \sum_j w_{ij}$ is the sum of spatial weights w_{ij}.

Moran (1950) derived this measure from the Pearson's correlation coefficient, as a measure of correlation between two random variables by replacing Y's with X's in Equation 2.13. This is done by computing the numerator term only when areal units i and j are nearby, i.e., spatial weight $w_{ij} = 1$ for neighbors, and 0 otherwise, and by averaging the numerator cross-product terms over the total number of pairs denoted as being nearby. Therefore, Moran's *I* can be seen as a spatially weighted version of the Pearson's correlation coefficient.

Dubin (1998) showed that the expected values of Moran's *I* can be determined mathematically by using the eigenvalues of the spatial matrix W, that is:

$$E(I_Y) = \frac{-1}{N-1} \tag{3.6}$$

where $I_Y(C) \geq -1/(n-1)$ indicates positive autocorrelation, while $I_Y(C) \leq -1/(n-1)$ indicates negative autocorrelation. The values of $I_Y(C)$ generally range from -1 to +1 and high positive (negative) values of $I_Y(C)$ indicate strong positive (negative) autocorrelation.

Global Moran's *I* compares the value of the variable at any location with the value at all other locations (Legendre, 1993). However, in most cases, we are interested in comparing the values of a variable within a certain neighborhood. Therefore, the calculation of the Global Moran's *I* requires a spatial weights matrix W that reflects the spatial relationship between observations and can be used to define a neighborhood.

Global Geary's *C*

Another common measure or spatial autocorrelation between response values at different points, also very much exploited in spatial data analysis is Global Geary's *C* (Legendre, 1993). It uses paired comparisons (i.e., pairwise squared differences) and may be defined as:

$$C_Y = \frac{(N-1)\sum_i \sum_j w_{ij}(Y_i - Y_j)^2}{2W\sum_i (Y_i - \overline{Y})^2} \tag{3.7}$$

where N is the number of spatial examples indexed by i and j; Y_i and Y_j are the values of the variable Y for the examples o_i and o_j, respectively; Y is the variable of interest; $\overline{Y}$ is the overall mean of Y; and $w_{ij}, i, j = 1, \ldots, N$ are the values of the $N \times N$ matrix of spatial weights, i.e., $W = \sum_{i,j} w_{ij}$. The matrix of spatial weights W can be defined with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7, as already discussed in Section 2.2.2.

The interpretation of the Global Geary's C is very different from the one of Global Moran's I. Global Geary's C generally varies on a scale from 0 to 2 where 0 indicates perfect positive autocorrelation (clustered pattern in the data distribution), 1 indicates no autocorrelation (random distribution of the data) and 2 indicates perfect negative autocorrelation (dispersed pattern in the data distribution). The expected value of Global Geary's C is 1.

One can easily convert the values of the Global Geary's C to the scale of the Global Moran's I values (by using $I = 1 - C/2$). However, this can only be used as an approximation since their values are not equivalent. It is possible that extreme values of Moran's I and Geary's C fall outside the ranges mentioned above (de Jong et al, 1984).

Similar to the calculation of the Global Moran's I, the calculation of Global Geary's C requires a spatial weight matrix. Moreover, for the calculation of Global Geary's C the spatial weight matrix plays a bigger role, since this measure is more sensitive to differences in small neighborhoods, whereas the Global Moran's I measure gives a more global indicator of the presence of spatial autocorrelation in the data.

While both measures reflect the spatial dependence of values, they do not provide identical information: C emphasizes the differences in values between pairs of observations, while I emphasizes the covariance between the pairs. This means that Moran's I is smoother, whereas Geary's C is more sensitive to differences in small neighborhoods.

In addition, Moran's I is preferred in most cases since Getis and Ord (1992) have shown that Moran's I is statistically and consistently more powerful than Geary's C. The algebraic relationship between the two measures is given below:

$$C_Y = \frac{(N-1)}{N}\frac{N\sum_i(\sum_j w_{ij})(Y_i - \overline{Y})^2}{W\sum_i (Y_i - \overline{Y})^2} - I_Y \tag{3.8}$$

This equation reveals that Geary's C incorporates location information, in addition to the information included in Moran's I. This additional information is the ratio of squared deviations times their number of neighbors, divided by the sample variance. If outliers are present, then this numerator can become excessively large. If an areal unit has a large number of neighbors, then this numerator can be markedly influenced by the corresponding deviation.

Getis and Ord's *G*

Getis and Ord's G (Getis and Ord, 1992) is another measure of spatial autocorrelation commonly used in spatial data analysis, defined only for positive variables Y.

Formally, Getis and Ord's G is defined as:

$$G_Y = \frac{\sum_i \sum_j w_{ij}(Y_i Y_j)}{\sum_i Y_i Y_j}, \forall j \neq i \tag{3.9}$$

where N is the number of spatial examples indexed by i and j; Y_i and Y_j are the values of the variable Y for the examples o_i and o_j, respectively; Y is the variable of interest and $w_{ij}, i, j = 1, \ldots, N$ and $w_{ij}, i, j = 1, \ldots, N$ are the values of a $N \times N$ matrix of binary and symmetric spatial weights, i.e., $W = \sum_{i,j} w_{ij}$.

The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already discussed in Section 2.2.2.

Its values are in the interval $[-\infty, +\infty]$. The higher (lower) the value, the stronger the intensity of the autocorrelation. A positive (negative) Getis and Ord's G indicates a positive (negative) autocorrelation. When Getis and Ord's G is near zero, it indicates that no apparent autocorrelation is present within the considered data.

In contrast to the previously defined distanced-based spatial autocorrelation measures, Global Moran's I and Global Geary's C, Getis and Ord's G is a spatial clustering measure. This means that it is not dependent from the distance between the examples o_i and o_j, but considers all "neighbors" of the object o_i, i.e., all object that are located at a certain spatial distance d. More precisely, Getis and Ord's G measures the association degree between the values of Y around location i and the association in the value of Y around location j.

Mantel's Γ statistic

Mantel's test (Mantel, 1967) is a versatile statistical test that has many uses. The rationale of this test is to permute the independent entities in order to obtain a distribution of the target statistic under $H0$. This distribution is then used to test the parameter of interest. Although it has been called "test", the novelty introduced by Mantel was actually the permutation procedure. This procedure has been developed to analyze the relationship between regression relationship matrices and can be employed with regression coefficients, correlation coefficient or any other statistic.

The corresponding statistics is also used as an autocorrelation measure in spatial data analysis. The basic Mantel's Γ statistic is simply the sum of the products of the corresponding elements of the matrices X and Y:

$$\Gamma_Y = \sum_{ij} X_{ij} Y_{ij} \tag{3.10}$$

where $\sum_{ij}$ is the double sum over all i and j where $i \neq j$. Because Γ can take any value depending on the exact nature of X and Y, one usually uses a normalized Mantel coefficient, calculated as the correlation between the pairwise elements of X and Y. Like any product-moment coefficient, it ranges from -1 to 1, where 1 means high positive autocorrelation and -1 high negative autocorrelation.

Primarily, it examines the relationship between two square matrices (often distance matrices) X and Y. The values within each matrix (X_{ij} or Y_{ij}) represent a relationship between examples i and j. The relationship represented by a matrix could be a geographic distance, a data distance, an angle, a binary matrix, or almost any other conceivable data. Often one matrix is a binary matrix representing a hypothesis of relationships among the points or some other relationship (e.g., X_{ij} may equal 1 if examples i and j are from the same country and 0 if they are not). By definition, the diagonals of both matrices are always filled with zeros.

One advantage of Mantel's test is that, because it proceeds from a distance (dissimilarity) matrix, it can be applied to variables of different logical type (categorical, rank, or interval-scale data).

Local Moran's *I*

In addition to the Global Moran's I, a localized version of this measure called Local Moran's I has been developed (Anselin, 1988). The Local Moran's I is used to measure autocorrelation at a certain location in space and results with a unique value for each location. Differently from the Global Moran's I, which measures the covariance between pairs of observations, the Local Moran's I measures the joint covariance of neighboring localities.

Formally, the Local Moran's I statistic for each observation i is defined as:

$$I_{Y_i} = \frac{N(Y_i - \overline{Y}) \sum_j w_{ij}(Y_j - \overline{Y})^2}{\sum_j (Y_j - \overline{Y})^2} \tag{3.11}$$

where Y_i is the value of the variable at the i-th location, N is the number of observation, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j (already discussed in Section 2.2.2) and $\overline{Y}$ is the mean of Y.

The values of the Local Moran's I are in the interval $[-\infty, +\infty]$. The higher (lower) the value, the stronger the intensity of the autocorrelation. A positive (negative) Local Moran's I indicates a positive (negative) autocorrelation.

The Local Moran's I can be seen as the local equivalent of Moran's I, i.e., the sum of all local indices is proportional to the (global) value of Moran's statistic. From the previous equations, one can easily obtain the relationship between the Global Moran's I (see Equation 3.4) and the Local Moran's I (see Equation 3.11), that is:

$$I_Y = \frac{\sum_i I_{Y_i}}{N} \tag{3.12}$$

The importance of Local Moran's I is motivated by the possible occurrence of different patterns or processes in different parts of the region and this may not follow the global patterns or processes that can be detected by the use of the global measures of autocorrelation such as Global Moran's I.

Local Geary's C

Another popular measure of local spatial autocorrelation is the **local Geary's** C, defined as a localized version of the Global Geary's C. As Local Moran's I, Local Geary's C is a local equivalent of Geary's C.

Formally, the Local Geary's C statistic for each observation i is defined as:

$$C_{Y_i} = \frac{(N-1)\sum_j w_{ij}(Y_i - Y_j)^2}{2W \sum_j (Y_j - \overline{Y})^2} \tag{3.13}$$

where Y_i is the value of the variable at the i-th location, N is the number of observation, w_{ij} is a weight indicating something about the spatial relationship of the observations i (already discussed in Section 2.2.2), $W = \sum_{i,j} w_{ij}$ is the spatial matrix and j and $\overline{Y}$ is the mean of Y.

Unlikely to Local Moran's I_i, Local Geary's C_i statistic is the weighted sum of the squared differences between location i and locations j. Analogous to Local Moran's I_i, its values are in the interval $[-\infty, +\infty]$. The high positive (negative) value indicates strong positive (negative) autocorrelation.

Differently from the Global Geary's C which measures the difference in values between the pairs of observations, the Local Geary's C measures the joint differences of neighboring localities. The relationship between the Global Geary's C (see Equation 3.7) and the Local Geary's C (see Equation 3.13) is similar to the relationship between Global Moran's I and the Local Moran's I given with Equation 3.12) mentioned before. Moreover, this property of the Local Moran and Local Geary statistic to be associated with the global measure (Moran's I and Geary's C, respectively), can be used to estimate the contribution of individual measures to the corresponding global measures.

The importance of Local Geary's C is related to the identification of so-called "hot-spots" regions where the considered phenomenon is extremely pronounced across localities, as well to the detection of spatial outliers.

Local Getis and Ord's *G*

Another measure of local spatial autocorrelation is the Local Getis and Ord's *G* (Getis and Ord, 1992), defined as the proportion of all values of the response variable accounted for by the neighbors of the observation *i*.

Formally, the Local Getis and Ord's *C* statistic for each observation *i* is defined as:

$$G_i(d) = \frac{\sum_j w_{ij} Y_j}{\sum_j Y_j} \tag{3.14}$$

where Y_i is the value of the variable at the *i*-th location, N is the number of observation, w_{ij} is a weight indicating something about the spatial relationship of the observations *i*, $W = \sum_{i,j} w_{ij}$ is the spatial matrix and j and $\overline{Y}$ is the mean of Y. The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already been discussed in Section 2.2.2.

The values of the Local Getis and Ord's G range in the interval $[-\infty, +\infty]$. G will be high (low) where high (low) values group together. The relationship between the Global Getis and Ord's G (see Equation 3.9) and the Local Getis and Ord's G (see Equation 3.14) reflects that of the Global Moran's I and Local Moran's I.

Bivariate Moran's *I*

Despite the fact that the research related to measures of spatial autocorrelation focuses mainly on the measures of spatial autocorrelation of one variable, in the literature, some measures of the spatial autocorrelation of two and more variables have been proposed.

The cross-correlation between two continuous variables y_1 and y_2 can be estimated by Bivariate Moran's I (Zhu et al, 2007) and defined as:

$$I_{y_1 y_2} = \frac{N \sum_i [(\sum_j w_{ij}(y_{1i} - \overline{y_1}))(\sum_j w_{ij}(y_{2j} - \overline{y_2}))]}{\sum_i (\sum_j w_{ij})^2 \sqrt{\sum_i (y_{1i} - \overline{y_1})^2} \sqrt{\sum_i (y_{2i} - \overline{y_2})^2}} \tag{3.15}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j (already discussed in Section 2.2.2) and $\overline{y_1}$ and $\overline{y_2}$ are the means of the variables y_1 and y_2, accordingly.

A localized version of the Bivariate Moran's I can be easily derived and it can be used as a local indicator of the spatial relationship between two variables at a certain location, which may not always refer to the same global patterns of relationship.

Bivariate Association Measure

Besides the spatial autocorrelation of a single variable, in the literature, there exists also a measure of the cross-correlation between two distinct variables which is different from the former one.

The bivariate association measure (L) (Lee, 2001) presents an integration of spatial association measure Global Moran's I and Pearson's correlation coefficient r and is intended to capture spatial association among observations in terms of their point-to-point relationships across two spatial patterns, i.e., capture the spatial co-pattering by collectively gauging point-to-point association between two variables and the topological relationship among spatial entities. The bivariate association measure between two continuous variables y_1 and y_2 is defined as:

$$L_{y_1 y_2} = \frac{N \sum_i \sum_j w_{ij}(y_{1j} - \overline{y_1})(y_{2j} - \overline{y_2})}{W \sqrt{\sum_i (y_{1i} - \overline{y_1})^2 \sum_i (y_{2i} - \overline{y_2})^2}} \tag{3.16}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j (already discussed in Section 2.2.2) and $\overline{y_1}$ and $\overline{y_2}$ are the means of the variables y_1 and y_2, accordingly.

This measure provides a complementary measure to Pearson's correlation coefficient, as it efficiently captures how much bivariate association are spatially clustered. In other words, this measure can be used to parameterize the bivariate spatial dependence. Moreover, one can easily derive a local spatial bivariate association measure that will indicate the relative contribution of an individual area into the global L, as well as capture the observation's association with its neighbors in terms of the point-to-point association between two variables. Furthermore, L can be used as a measure of spatial segregation or dissimilarity) or to specialize other multivariate statistical procedures, such as principal component analysis.

3.2.2 Measures for Classification

For the classification task, there exist a few measures of spatial autocorrelation. In dissertation, besides the existing measures of spatial autocorrelation in the literature, we redefine/adapt the spatial measures of autocorrelation used for the regression task.

Joint Count Statistics

Join count statistics (Cliff and Ord, 1981) is the simplest measure of spatial autocorrelation. It can only be used for a binary variable (1 or 0). It turns out to be useful when the spatial data comes represented as two or more dimensional spatial examples like polygons, areas and regions, but not when the spatial data is represented as one dimensional point data.

The two values of the variable are referred to as "black" (B) and "white" (W). A join links two neighboring areas. So the possible types of joins are black-black (BB), black-white (BW), and white-white (WW). Join counts are counts of the numbers of BB, BW, and WW joins in the study area and these numbers are compared to the expected numbers of BB, BW and WW joins under the null hypothesis of no spatial autocorrelation. Therefore, the calculation of the Joint Count Statistic requires a contiguity matrix for spatial examples.

The observed number of BB, BW and WW joins are defined as:

$$BB = \frac{\sum_i \sum_j w_{ij} x_i x_j}{2} \tag{3.17}$$

$$BW = \frac{\sum_i \sum_j w_{ij} (x_i - x_j)^2}{2} \tag{3.18}$$

$$WW = \frac{\sum_i \sum_j w_{ij} (1 - x_i)(1 - x_j)}{2} \tag{3.19}$$

where N is the number of observations, x_i is the binary variable, 1 for black and O for white, w_{ij} is the spatial weight, 1 if two areas are contiguous, 0 otherwise.

The join count statistics relates the number of observed connections between the zones of property "presence" and those of property "absence" with the theoretical number of connections of a random distribution. The definition of the theoretical number of connections of a random distribution is related to two factors: the spatial arrangement of features of the study area and the choice of the null hypothesis.

Global Moran's *I*

A very common global measures of spatial autocorrelation is the Global Moran's I. However, it has been defined for a continuous variable. Therefore, within the scope of this dissertation, we redefine and use it also for a discrete variable, i.e., for the classification task.

The redefinition of the Global Moran's I for a discrete valuable Y requires a transformation of the formula of the Global Moran's I for a continuous variable (Equation 3.4). In particular, we transform the discrete variable Y into q $0/1$ binary variables $Y_1, Y_2, \dots, Y_q$. Then we compute I_Y as the average of I computed for each Y_k ($k = 1, 2, \dots, q$), that is:

$$I_Y = \frac{1}{q}\sum_{k=1}^{q} \frac{N \sum_i \sum_j w_{ij}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_i \sum_j w_{ij} \sum_i (Y_i - \overline{Y})^2} \tag{3.20}$$

where N is the number of observations, $w_{ij}, i, j = 1, \dots, N$ are the values of a $N \times N$ matrix of spatial weights, i.e., $W = \sum_{i,j} w_{ij}$ and $\overline{Y}$ is the mean of each Y_k ($k = 1, 2, \dots, q$).

The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already been discussed in Section 2.2.2.

Its values generally range from -1 (indicating perfect dispersion) to +1 (perfect correlation). A zero value indicates a random spatial pattern. For more details on this measure, see its definition for the regression task in Section 3.2.1.

Global Geary's *C*

Another common global measures of spatial autocorrelation is the Global Geary's C. Identically as the Global Moran's I, it has also been defined for a continuous variable (Equation 3.7). Thus, in order to use it within the classification task, first we need to redefine it for the case of a discrete variable.

Again, the redefinition of the Global Geary's C for a discrete valuable Y requires a transformation of the formula of the Global Geary's C for a continuous variable. First, we transform the discrete variable Y into q $0/1$ binary variables $Y_1, Y_2, \dots, Y_q$. Then we compute C_Y as the average of C computed for each Y_k ($k = 1, 2, \dots, q$), that is:

$$C_Y = \frac{1}{q}\sum_{k=1}^{q} \frac{(N-1) \sum_i \sum_j w_{ij}(Y_i - Y_j)^2}{2 \sum_i \sum_j w_{ij} \sum_i (Y_i - \overline{Y})^2} \tag{3.21}$$

where N is the number of observations, $w_{ij}, i, j = 1, \dots, N$ are the values of a $N \times N$ matrix of spatial weights, i.e., $W = \sum_{i,j} w_{ij}$ and $\overline{Y}$ is the mean of each Y_k ($k = 1, 2, \dots, q$).

The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already been discussed in Section 2.2.2.

Generally, global Geary's C varies on a scale from 0 to 2 where 0 indicates perfect positive autocorrelation (clustered pattern in the data distribution), 1 indicates no autocorrelation (random distribution of the data) and 2 indicates perfect negative autocorrelation (dispersed pattern in the data distribution).

For more details on this measure, see its definition for the regression task in Section 3.2.1.

Local Moran's *I*

Besides the global measures of spatial autocorrelation that estimate the overall degree of spatial association across the whole data set (spatial region or area), local measures exist of spatial autocorrelation that estimate the local degree of spatial association in the data. The Local Moran's I, as a local version of the Global Moran's I, is a common local measure of spatial autocorrelation that is often used in spatial analysis.

As for the Global Moran's *I*, the calculation of the Local Moran's *I* for a discrete valuable *Y* requires a transformation of the formula of the Local Moran's *I* for a continuous variable. Therefore, we redefine and use it also for a discrete variable, i.e., for the classification task.

In particular, we transform the discrete variable *Y* into q $0/1$ binary variables $Y_1, Y_2, \dots, Y_q$. Then we compute I_{Y_i} as the average of *I* computed for each Y_j ($j = 1, 2, \dots, q$), that is:

$$I_{Y_i} = \frac{1}{q} \sum_{i=1}^{q} \frac{N(Y_i - \overline{Y}) \sum_j w_{ij} (Y_j - \overline{Y})^2}{\sum_j (Y_j - \overline{Y})^2} \tag{3.22}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j and $\overline{Y}$ is the mean of each Y_i ($i = 1, 2, \dots, q$).

The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already been discussed in Section 2.2.2.

The values of the Local Moran's *I* are in the interval $[-\infty, +\infty]$. The high positive (negative) value indicates strong positive (negative) autocorrelation.

For more details on this measure, see its definition for the regression task in Section 3.2.1.

Local Geary's *C*

Another well known local measures of spatial autocorrelation is the Local Geary's *C*. The same as its global version, it can account the autocorrelation of a continuous variable. Therefore, we redefine and use it also for a discrete variable, i.e., for the classification task.

Equivalently to the Global Geary's *C* for the classification task, its calculation for a discrete valuable *Y* requires a transformation of the formula of the Local Geary's *C* for a continuous variable. For this purpose, first we transform the discrete variable *Y* into q $0/1$ binary variables $Y_1, Y_2, \dots, Y_q$. Then we compute C_{Y_i} as the average of *C* computed for each Y_j ($j = 1, 2, \dots, q$), that is:

$$C_{Y_i} = \frac{1}{q} \sum_{j=1}^{q} \frac{(N-1)(Y_i - \overline{Y}) \sum_j w_{ij} (Y_i - Y_j)^2}{\sum_j (Y_j - \overline{Y})^2} \tag{3.23}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j and $\overline{Y}$ is the mean of each Y_i ($i = 1, 2, \dots, q$).

The matrix of spatial weights W can be given with Equation 2.4, Equation 2.6, Equation 2.5 or Equation 2.7 and has already been discussed in Section 2.2.2.

Correspondingly, the values of the Local Geary's *C* are in the interval $[-\infty, +\infty]$. The high positive (negative) value indicates strong positive (negative) autocorrelation.

For more details on this measure, see its definition for the regression task in Section 3.2.1.

Bivariate Moran's *I*

A lot of the measures of spatial autocorrelation focus mainly on the measures of spatial autocorrelation of one variable. However, in the literature, there have been proposed some measures of the spatial autocorrelation of two and more variables. One common measure of the spatial cross-correlation of two continuous variables is the Bivariate Moran's *I* (Zhu et al, 2007).

As already explained in the previous subsections, we need to redefine the Bivariate Moran's *I* (Zhu et al, 2007) tests for the case of two discrete variables y_1 and y_2. One way to achieve this is to transform each of the discrete variables Y into q $0/1$ binary variables $Y_1, Y_2, \ldots, Y_q$. Then we compute C_Y as the average of C computed for each Y_k ($k = 1, 2, \ldots, q$), that is:

$$I_{y_1 y_2} = \frac{1}{q} \sum_{k=1}^{q} \frac{N \sum_i [(\sum_j w_{ij}(y_{1i} - \overline{y_1}))(\sum_j w_{ij}(y_{2j} - \overline{y_2}))]}{\sum_i (\sum_j w_{ij})^2 \sqrt{\sum_i (y_{1i} - \overline{y_1})^2} \sqrt{\sum_i (y_{2i} - \overline{y_2})^2}} \tag{3.24}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j (already been discussed in Section 2.2.2) and $\overline{y_1}$ and $\overline{y_2}$ are the means of the variables y_1 and y_2, accordingly. The values of the Bivariate Moran's I are in the interval [-1,1], where -1 means high negative autocorrelation, 0 means no autocorrelation (random distribution of the data) and 1 means high positive autocorrelation.

For more details on this measure, see its definition for the regression task in Section 3.2.1.

Bivariate Association Measure

Another measure of spatial autocorrelation of two variables is the bivariate association measure (L) (Lee, 2001), defined for continuous valuables.

In order to use this measure to account for cross-correlation between two discrete variables y_1 and y_2 spatial autocorrelation measure that tests for spatial . In particular, we transform each of the discrete variables Y into q $0/1$ binary variables $Y_1, Y_2, \ldots, Y_q$. Then we compute C_Y as the average of C computed for each Y_k ($k = 1, 2, \ldots, q$), that is:

$$L_{y_1 y_2} = \frac{1}{q} \sum_{k=1}^{q} \frac{N \sum_i \sum_j w_{ij}(y_{1j} - \overline{y_1})(y_{2j} - \overline{y_2})}{W \sqrt{\sum_i (y_{1i} - \overline{y_1})^2 \sum_i (y_{2i} - \overline{y_2})^2}} \tag{3.25}$$

where N is the number of observations, w_{ij} is a weight indicating something about the spatial relationship of the observations i and j (already been discussed in Section 2.2.2) and $\overline{y_1}$ and $\overline{y_2}$ are the means of the variables y_1 and y_2, accordingly.

The values of L range in the interval [-1,1], where -1 means high negative autocorrelation, 0 means no autocorrelation and 1 means high positive autocorrelation. For more details on this measure, see its definition for the regression task in Section 3.2.1.

Spatial Diversity Coefficient

Spatial diversity coefficient is defined by Li and Claramunt (2006) as a measure of spatial diversity, i.e., as the dispersion of the entropy measure over some neighborhoods. The coefficient is integrated into a ID3 decision tree, where Entropy and Gain are redefined.

The coefficient is adapted to either discrete or continuous spaces and not limited to a bounded boundary as distances rather than neighborhoods are considered. It is expressed as ratio of the of the "intra-distance" and "extra-distance". This means that it increases when either the average distance (intra-distance d_i^{int}) between the entities belonging to a given category decreases, or the average distance (extra-distance d_i^{ext}) between the entities of a given category and the entities of all the other categories increases and vice versa. These average distances are defined as follows:

$$d_i^{int} = \frac{1}{| C_i | (| C_i | - 1)} \sum_j \sum_k dist(j,k) if \mid C_i \mid > 1; and d_i^{int} = \lambda, otherwise \tag{3.26}$$

$$d_i^{ext} = \frac{1}{| C_i || C - C_i |} \sum_j \sum_k dist(j,k) if \mid C_i \neq C; and d_i^{ext} = \beta, otherwise \tag{3.27}$$

where C is the set of spatial entities of a given dataset; C_i denotes the subset of C whose entities belong to the i-th category of the classification; d_i^{int} is the average distance between the entities of C_i; d_i^{ext} is the average distance between the entities of C_i and the entities of the other categories; $dist(j,k)$ gives the distance between the entities j and k; l is a constant taken relatively small, and b a constant taken relatively high; these constants avoid the noise $\pm$ effect of null values in the calculation of the average distances.

These average distances are integrated in such a form that exhibits an increase of spatial entropy when the intra-distance d_i^{int} increases and extra-distance d_i^{ext} decreases, and conversely. (Li and Claramunt, 2006) show that the values of the spatial diversity coefficient are given by the interval [0, 2].

The spatial entropy $Entropy_s()$ is then defined as follows:

$$Entropy_s(A) = -\sum_i \frac{d_i^{int}}{d_i^{ext}} P_i log_2(P_i) \tag{3.28}$$

where d_i^{int} is the average distance between the entities of C_i, d_i^{ext} is the average distance between the entities of C_i and P_i is the proportion of entities labeled with value i over the total number of entities.

The notion of spatial entropy provides a means for an integration of the spatial dimension within the ID3 classification algorithm. The strategy retained is to replace the conventional measure of entropy $Entropy()$ with the measure of spatial entropy $Entropy_s()$. Moreover, $Entropy_s()$ is determined by not only the richness and the evenness of the distribution as the classical entropy $Entropy()$ is, but also by the ways the spatial entities are distributed in space. This means that changes in the spatial distribution of the dataset might increase or decrease the spatial entropy even if the category constitution remains identical. Furthermore, the spatial entropy surpasses the conventional entropy in the evaluation of the diversity a given spatial system exhibits.

The information gain at each level of the decision tree is replaced with the following expression $Gain_s()$:

$$Gain_s(GA,SA) = Entropy_s(GA) - \sum_{v \in values(SA)} \frac{\mid GA_v \mid}{\mid GA \mid} Entropy_s(GA_v) \tag{3.29}$$

where $Entropy_s(GA)$ is the spatial entropy of the target attribute GA, GA_v is a subset of entities of GA where the corresponding value of SA is v for each entity and $\mid GA_v \mid$ and $\mid GA \mid$ are the cardinalities of GA_v and GA, respectively.

The main principle of the ID3 decision tree built using these new measures is still valid. At each level of such a spatial form of a decision tree, the supporting attribute that gives the maximum spatial information gain $Gain_s$ is selected as a node. This guarantees that the spatial entities of a same category are preferably aggregated. By contrast, the application of a conventional decision tree learner may lead to situations in which entities of different categories might not be spatially distinguished as clearly as those of a spatial decision tree.

Spatial Information Gain

Rinzivillo and Turini (2007) also redefined the classical information gain used in a standard decision tree induction procedure.

For a given reference object they described its neighborhood by considering the attribute of the object itself and the spatial examples related by the chosen relations. The resulting spatial transactions may be either considered like “traditional” transactions, by considering only the qualitative spatial relations, or by their spatial extension which can be exploited during the data mining process. The Spatial information gain that they define is given by:

$$Gain = -\sum_{l} \frac{mes(S \mid c_i)}{mes(S)} log_2 \frac{mes(S \mid c_i)}{mes(S)} - \sum_{j} \frac{mes(S \mid v_j)}{mes(S)} H(S \mid v_j) \tag{3.30}$$

where $mes(t)$ is the aggregated spatial measure of the spatial transaction t, S is a set of spatial transactions whose class label attribute has l distinct classes (i.e., $c_1, c_2, \ldots, c_l$) and $S \mid c_i$ is the set of transactions labeled by c_i; V is an attribute that has q distinct values (i.e., $v_1, v_2, \ldots, v_q$).

This measure is then used to compute the entropy for a spatial measurement of each example and to use the information gain based on such spatial entropy measure for the induction of spatial decision trees. In particular, the spatial entropy is computed for each weighted sum of the spatially related (e.g., overlapping) examples.

In summary, measures of spatial autocorrelation can only provide a snapshot of autocorrelation, i.e., they do not consider the temporal dimension in addition to the spatial dimension. Consequently, they do not capture dynamic autocorrelation properties. For this purpose, measures for spatio-temporal autocorrelation, as a special case of autocorrelation, are required. We give an overview of the most relevant measures of spatio-temporal autocorrelation.

3.3 Measures of Spatio-Temporal Autocorrelation

A number of measures have been defined for the case of spatio-temporal autocorrelation. Most of them are extensions of the spatial or temporal autocorrelation function and have limited use since they have been defined within some specific framework for spatio-temporal modeling.

3.3.1 Measures for Regression

For the regression task, several measure of spatio-temporal autocorrelation that can be used to examine the presence of spatio-temporal autocorrelation have been defined. These include: Space-Time Index, Space-time variograms and Space-Time Autocorrelation Function, as the most common ones.

Space-Time Index

The Space-Time Index (STI) is one of several related measures that have been proposed for assessing spatio-temporal autocorrelation ((Cliff and Ord, 1981; Griffith, 1981)). The formulation provided by Griffith combines the Moran's I with the Durbin-Watson statistic (Durbin and Watson, 1950), to yield a first-order STI:

$$STI_Y = (nT - n) \frac{\sum_{t=2}^{T} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ijt-1} (Y_{ij} - \overline{Y})(Y_{jt-1} - \overline{Y})}{\sum_{t=2}^{T} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ijt-1} \sum_{t=2}^{T} \sum_{i=1}^{n} (Y_{it} - \overline{Y})^2} \tag{3.31}$$

where T is the number of temporal units, n is the number of spatial units; w_{ijt} is a weight indicating the spatio-temporal relationship of the observations i and j at time t; Y_{it} and Y_{jt} are the values of the variable of interest Y of the observations i and j at time t; $\overline{Y}$ is the mean of Y.

The values of the STI usually vary in the range [-1, 1], where -1 means high negative spatio-temporal autocorrelation, 0 means no autocorrelation (random distribution of the data) and 1 means high positive spatio-temporal autocorrelation. The expected value of STI is similar to expected value of the Global Moran's I. A major problem with this index is the possibility of spatio-temporal "anisotropy" introduced by the fact that the units of measurement in space may not be comparable (equivalent) to the units used to measure time.

Space-Time Variograms

Space-time (semi) variograms have also been proposed (Griffith and Heuvelink, 2009), as a measure for assessing spatio-temporal autocorrelation. They present the semi-variance function $\gamma(h)$ of a spatial lag h over a time lag t. Assuming time lag of $t = 0$, the semi-variance function for a continuous variable is defined as:

$$\gamma(h) = \frac{1}{2n(h)} \sum_{i=1}^{n} [Z(X_i) - Z(X_i + h)]^2 \tag{3.32}$$

where Z is the value of the variable x at the sampling location i and $n(h)$ is the number of pairs of sampling locations located at distance h from one another.

The equation for the semi-variance function (See Equation 3.32)) is quite comparable to the one of spatial Global Geary's C (see Equation 3.7)), except that it lacks the division by the standard deviation in the denominator, which standardizes the spatial autocorrelation value. Hence, the semi-variance function is in the same units as the data and is not bounded as are global spatial statistics and other statistics. At short distance lags, the values of semi-variance are also small (close to zero) indicating that the spatial structure is at its strongest intensity. As the distance lags increase, the semi-variance values rise to level off at a plateau called the sill.

When a variogram is computed from sampled data, it is called an experimental variogram (also called a sample or observed variogram); when it is modeled to fit the experimental variogram, it is called a theoretical variogram or a variogram model.

Three key parameters are estimated from an experimental variogram to fit a theoretical variogram, these are: the nugget effect c_0, the spatial range a and the sill c_1. The nugget is the intercept at the origin that is greater than zero. Theoretically, at $h = 0$, the semi-variance is also equal to 0. However, based on the shape of the experimental variogram, it can be unrealistic sometimes to force the theoretical variogram to go through 0. The nugget parameter is therefore used to account for the observed variability at short distances due to local random effects or measurement errors (e.g. accuracy of measurements, inappropriate sampling unit size, etc.). The spatial range indicates the distance up to which the spatial structure varies. In other words, the range indicates the maximal distance at which the variable is spatially autocorrelated.

Space-Time Autocorrelation Function

Another measure of global spatio-temporal autocorrelation is the space-time autocorrelation function (ST-ACF) (Pfeifer and Deutsch, 1980). It is actually the N^2 cross-covariances between all possible pairs of locations lagged in both time and space. Given the weighted l-th order spatial neighbors of any spatial location at time t and the weighted k-th order spatial neighbors of the same spatial location at time lags s (time $t+s$) in the future, the ST-ACF can be defined as:

$$ST-ACF = \frac{E\left(\frac{[W^l z(t)]'[W^k z(t+s)]}{N}\right)}{(E\left(\frac{[W^l z(t)]'[W^l z(t)]}{N}\right) E\left(\frac{[W^k z(t)]'[W^k z(t)]}{N}\right))^{0.5}} \tag{3.33}$$

where E is the expected value operator, N is the number of spatial locations, W^l and W^k are the $N \times N$ spatial weight matrices at spatial orders l, and $z(t)$ is the $N \times 1$ vector of observations z at time t, $z(t+s)$ is the $N \times 1$ vector of observations z at time $t+s$ and the symbol $'$ denotes matrix transposition.

Cross-Correlation Function

Besides the measures of spatio-temporal autocorrelation of one time series, a cross-correlation function (CCF) that measures the spatio-temporal autocorrelation of two time series has been proposed by Box et al (1994). CCF actually measures the local spatio-temporal autocorrelation between two locations. It treats two time series as a bivariate stochastic processes and measures the cross-covariance coefficients between each series at specified lags. Therefore, it is a measure of the similarity between two time series. Given two time series X and Y, the CCF at lag k is given as:

$$CCF_{xy}(k) = \frac{E[(x_t - \mu_x)(Y_{t+k} - \mu_y)]}{\sigma_x \sigma_y}, k = 0, \pm 1, \pm 2, \pm ... \tag{3.34}$$

where μ_x and μ_y are the means and σ_x and σ_y are the standard deviations of the two time series X and Y, respectively and k is the time lag. The values of the CCF usually vary in the range [-1, 1], where -1 means high negative spatio-temporal autocorrelation, 0 means no autocorrelation (random distribution of the data) and 1 means high positive spatio-temporal autocorrelation.

The CCF can be treated as a lagged specification of Pearson's correlation coefficient that measures cross-correlations in both directions, as denoted by subscript k; therefore, the temporal lag at which the CCF peaks can be used to determine a transfer function between two series. A simple way to interpret the CCF is by taking its squared value $CCF_{xy}(k)^2$ to give the coefficient of determination used in network data analysis.

3.3.2 Measures for Classification

For the classification task, similarly as for the spatial autocorrelation, the join count statistic can be used in the case of binary target variable. Therefore, an extended version of it, called **Spatio-temporal join count statistic** can be defined by considering values of the target variable between pairs of spatial examples observed at different times.

The two values of the variable are referred to as "black" (B) and "white" (W). A 'join' links two neighboring areas. So the possible types of joins are black-black (BB), black-white (BW), and white-white (WW). Join "length" is the pair of factors (space, time) that describes the distance between examples in space and in time, respectively. A spatio-temporal pattern is detected by comparing the number of joins of each 'length' in a particular lattice with the expected number from randomness. The simple null hypothesis is that the observed number can be accounted for by random occurrence. If more joins of a particular class occur than expected, this indicates a tendency for examples to be separated by a distance in space and in time.

3.4 Measures of Network Autocorrelation

Various ways of measuring the relational dependency on a network (or graph in more general context) exist. We present the most important measures of network autocorrelation, distinguishing between measures used for the classification and the regression task. Most of them have been developed and used for the task of classification, mainly in the field of collective inference, whereas less attention has been given to the task of regression and the development of measures of network autocorrelation for regression.

3.4.1 Measures for Regression

In the literature, very little attention has been given to the task of network regression and the development of measures of network autocorrelation for regression task. In the following, we give an overview of the existing measures.

Connectivity Index

A simple connectivity (topology) index which is limited to the consideration of the structure of a network (or graph) can be used as a measure of network autocorrelation for the regression task.

We exploit the extension of the classical Randic Connectivity Index (*CI*) to the case of weighted networks (Randic, 1998). Formally, the Connectivity Index *CI* is defined by Equation 3.35 as:

$$CI_Y(G) = \sum_{u,v \in V(G)} \frac{1}{\sqrt{d(u)d(v)}} \tag{3.35}$$

where $d(u)$ and $d(v)$ represent the degrees of nodes u and v respectively.

This index gives the connectedness (or branching) of a cluster C and can be used to compare the connectivity among clusters. It is typically used in chemistry, since it can be well correlated with a variety of physico-chemical properties of alkanes, such as boiling points, surface area and solubility in water. *CI* ranges in the interval $[-\infty, +\infty]$. High values of *CI* indicate high correlation, while low values of *CI* indicate low correlation (Randic, 1998).

Relational Autocorrelation

Relational Autocorrelation (*RA*) can also be used to assess the association between values of a continuous variable Y over connected nodes. One possibility is to use a variant of the Pearson's correlation coefficient that measures the correlation of a continuous variable Y with itself:

$$RA_Y = \frac{\sum_{ij \text{ } s.t. \text{ } (u_i,u_j) \in P_R} (Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_{ij \text{ } s.t. \text{ } (u_i,u_j) \in P_R} (Y_i - \overline{Y})^2} \tag{3.36}$$

The values of *RA* generally range from -1 to +1 and high positive (negative) values of *RA* indicate strong positive (negative) network autocorrelation. *RA* is very similar to the Global Moran' *I* used as a measure of spatial autocorrelation for the regression task.

Measures adapted from Spatial Data Analysis

In addition to the already mentioned measures of network autocorrelation for the regression task, the measures of spatial autocorrelation **Global Moran's** *I* (Equation 3.4) and **Global Geary's** *C* (Equation 3.7) can be also considered as measures of network autocorrelation.

This consideration requires that we adapt and redefine these measures in the network setting that we consider. We achieve this by applying the appropriate weighting function, while the equations used for calculating these measures stay the same. In the case of network autocorrelation, the weights more often have a natural representation. For more details on the weights see Section 2.3.2. Moreover, in Chapter 7 we show how we use these measures in real-world network domains for the regression task.

3.4.2 Measures for Classification

A lot of measures of network autocorrelation have been developed and used for the task of classification, mainly in the field of collective inference. These include Label Consistency, Randic Connectivity Index, Neville's Relational Autocorrelation, Newman's assortative mixing and Park and Barabasi's Dyadicity and Heterophilicity, as well as measures adapted from other scientific fields like measures of spatial autocorrelation used in spatial statistics. Here we present these measures ordered by their complexity: from the most simple to the most complex ones.

Label Consistency

Label Consistency (also called Local Consistency) (Gallagher et al, 2008) is the simplest measures of network autocorrelation that can be used for the task of classification. Actually, it is the percentage of links connecting nodes with the same class label. For a uniform class distribution, the actual probability of a link connecting two nodes i and j of the same label is defined as:

$$label\_consistency = P(u_i = u_j \mid (i,j) \in E) = dh + \frac{1-dh}{\mid C \mid} \tag{3.37}$$

where C is the set of possible class labels, E is the set of edges and $u_i \in U$ where U is the set of labeled nodes. dh is the degree of homophily, defined by Sen et al (2008).

Since it is represented as a percentage of links connecting nodes with the same class label, it ranges in the interval [0,100] %, where high values mean high network autocorrelation and low values mean low network autocorrelation.

Relational Autocorrelation

Relational Autocorrelation (*RA*) is a measure of the network autocorrelation which is adapted in collective classification to estimate the strength of statistical dependencies of a variable Y in a linked network (Angin and Neville, 2008). Any traditional measure of association, such as the χ^2 statistic or information gain, can be used to assess the association between interconnected values of Y.

One possibility is to use a variant of the Pearson's correlation coefficient that measures the correlation of a discrete variable Y with itself. In particular, it is possible to transform the discrete variable Y into q $0/1$ binary variables $Y_1, Y_2, \ldots, Y_q$ and then compute RA_Y as the average of RA computed for each Y_j $(j = 1, 2, \ldots, q)$, that is:

$$RA_Y == \frac{1}{q}\sum_{j=1}^{q} \frac{\sum_{ij \text{ } s.t. \text{ } (u_i,u_j)\in P_R}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_{ij \text{ } s.t. \text{ } (u_i,u_j)\in P_R}(Y_i - \overline{Y})^2} \tag{3.38}$$

RA is very similar to the Global Moran *I* from spatial statistics. Therefore, as in the Global Moran *I*, high values of *RA* indicate positive autocorrelation, while low values indicate negative autocorrelation. More specifically, the values of *RA* generally range from -1 to +1 and high positive (negative) values of *RA* indicate strong positive (negative) network autocorrelation.

Assortative Coefficient

Another measure of the network autocorrelation is assortative mixing (Newman, 2002). It is defined as the bias in favor of connections between network nodes with similar class label. Of particular interest is the phenomenon of assortative mixing by degree, meaning the tendency of nodes with high degree to connect to others with high degree, and similarly for low degree. Because degree is itself a topological property of networks, this type of assortative mixing gives rise to more complex structural effects than other types [1].

The assortative mixing can be expressed by the assortative coefficient, given as:

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j q_k)}{\sigma_q^2} \tag{3.39}$$

[1] see http://en.wikipedia.org/wiki/Assortativity for details

where the term e_{jk} refers to the joint probability distribution of the remaining degrees of the two vertices j and k. This quantity is symmetric on an undirected graph and follows the sum rules $\sum_{jk} e_{jk} = 1$ and $\sum_j e_{jk} = q_k$.

The term q_k represents the distribution of the *remaining* degree. This captures the number of edges leaving the node, other than the one that connects the pair. The distribution of this term is derived from the degree distribution p_k as

$$q_k = \frac{(k+1)p_{k+1}}{\sum_j jp_j} \tag{3.40}$$

The assortative coefficient r is actually the Pearson correlation coefficient of degree between pairs of linked nodes. Hence, positive values of r indicate a correlation between nodes of similar degree, while negative values indicate relationships between nodes of different degree. In general, r lies between -1 and 1. When $r = 1$, the network is said to have perfect assortative mixing patterns, while at $r = -1$ the network is completely disassortative.

Dyadicity and Heterophilicity

Another way to measure the relational dependency on a graph is the **Dyadicity and Heterophilicity**, proposed by Park and Barabasi (2007). The *Dyadicity D* is defined as the connectedness between nodes with the same class label compared to what is expected for a random configuration, whereas the *Heterophilicity H* is defined as the connectedness between nodes with different class labels compared to what is expected for a random configuration.

Consider the case where each node is characterized by a binary class label. Let $n_1(n_0)$ be the number of nodes with class label $1(0)$ so that the total number of nodes N satisfies $N = n_1 + n_0$. This allows three kinds of links in the network: $(1-1)$, $(1-0)$, and $(0-0)$. We label the number of each link type m_{11}, m_{10}, m_{00}, respectively, satisfying $M = m_{11} + m_{10} + m_{00}$, where M is the total number of links in the network. If the class is distributed randomly among the N nodes, i.e., if any node has an equal chance of possessing it, the expected values of m_{11} and m_{10} are

$$\overline{m_{11}} = \frac{n_1(n_1-1)p}{2} \tag{3.41}$$

and

$$\overline{m_{10}} = n_1(N-n_1)p \tag{3.42}$$

where $p = \frac{2M}{N(N-1)}$ is the *connectance*, representing the average probability that two nodes are connected.

Statistically significant deviations of m_{11} and m_{10} from their expected values imply that class is not distributed randomly. It is possible to quantify the magnitude of such effects via dyadicity D and heterophilicity H defined as $D = \frac{m_{11}}{\overline{m_{11}}}$ and $H = \frac{m_{10}}{\overline{m_{10}}}$. The class is dyadic if $D > 1$ (antidyadic if $D < 1$) and heterophilic if $H > 1$ (heterophobic if $H < 1$).

High values of D parameter indicates a network where nodes with a given function 1 are connected more strongly among themselves than expected by random distribution of functionalities. $D > 1$ shows a dyadic behavior. In contrast, $D < 1$ indicates an anti-dyadic coupling.

Equivalently, we can consider the case when nodes with a given property 1 have a low density of links connecting nodes with function 0 than expected by random chance. In such a case, $H > 1$ shows a heterophilic behavior and $H < 1$ indicates a heterophobic configuration. Phase diagrams of possible values m_{11} and m_{10} (D,H) can be plotted showing the possible configurations for a given set of functionalities and network structures.

Generally, if $D > 1$ and $H > 1$, the nodes with the same class label have a clear clustering tendency within the network. However, the values for the pair (D,H) cannot be arbitrary and are constrained by networked structure. Moreover, some values can admit distinct network configurations, i.e., degeneracy factor. More details on this measure can be found in (Park and Barabasi, 2007).

Measures adapted from Spatial Data Analysis

In addition to the already mentioned measures of network autocorrelation, the measures of spatial autocorrelation **Global Moran's** I (Equation 3.4) and **Global Geary's** C (Equation 3.7) can be also considered as measures of network autocorrelation. However, their consideration requires that we adapt and redefine these measures in the network setting that we consider.

In order to adapt these Measures for Classification, we redefine them similarly as the respective measures of spatial autocorrelation for the classification task (see Section 3.2.2 and Section 3.2.2). Let Y be a discrete variable which admits q distinct values in its domain. To compute the Moran's index for Y we resort to a one-versus-all solution. In particular, we transform the discrete variable Y into q $0/1$ binary variables $Y_1, Y_2, \ldots, Y_q$.

We compute the Global Moran's I as the average of Moran's I values, computed for each Y_j ($j = 1, 2, \ldots, q$), that is:

$$I_Y = \frac{1}{q} \sum_{j=1}^{q} I_{Y_j} \tag{3.43}$$

Similarly, the Global Geary's C for a discrete variable Y is:

$$C_Y = \frac{1}{q} \sum_{j=1}^{q} C_{Y_j} \tag{3.44}$$

Analogously, the values of the Global Moran's I in the interval [-1,1], where -1 means high negative autocorrelation, 0 means no autocorrelation (random distribution of the data) and 1 means high positive autocorrelation. The values of the Global Geary's C are in the interval [0,2], where 0 means high positive autocorrelation, 1 means no autocorrelation and 2 means high negative autocorrelation.

What changes while redefining these measures, in comparison to the measured of spatial autocorrelation for the classification task, is the definition and the meaning of the weighting function. In the case of network autocorrelation, the weights more often have a natural representation. For more details on the weights see Section 2.3.2. Moreover, in Chapter 7 we show how we use these measures in real-world network domains.

4 Predictive Modeling Methods that use Autocorrelation

The work presented in this chapter concerns the predictive modeling methods listed in the literature that consider autocorrelation. We organize the text into four separate parts, each related to the different form of autocorrelation that it considers. Namely, we distinguish among methods that consider temporal autocorrelation, spatial autocorrelation, spatio-temporal autocorrelation and network autocorrelation.

Each of these parts is then divided according to the predictive task that it concerns. In particular, we distinguish between methods that can be used for the classification and regression task.

4.1 Predictive Methods that use Temporal Autocorrelation

In the following section, we give an overview of the methods used in temporal data mining, as well in temporal analysis and modeling. Focusing on the classification and regression tasks, we give an overview of the existing studies that consider and incorporate temporal autocorrelation and report the main specific characteristics of their experimental setting.

4.1.1 Classification Methods

Time series analysis has quite a long history. Techniques for statistical modeling and spectral analysis of real or complex-valued time series have been in use for more than 60 years. Weather forecasting, financial or stock market prediction and automatic process control have been some of the oldest and most studied applications of such time series analysis (Box et al, 1994). Temporal data mining, however, is of a more recent origin with somewhat different constraints and objectives. Although classification is one of the most typical tasks in supervised learning and the most explored data mining subfield, it has not deserved much attention in temporal data mining and generally in the actual uses of temporal information.

Temporal autocorrelation is difficult to diagnose and correct when the response data are discrete (classification task), in that information about the error properties contained in the data is limited and the models are usually nonlinear. One approach to handle temporal autocorrelation is to use *effects models*, i.e., models using a lagged value of the response variable as a regressor.

For example, Varina and Vidoni (2006) developed effects models for predicting categorical time series data, as an alternative to Markov chain models. They solved the problem of parameter estimation through a simple pseudo-likelihood, i.e., pairwise likelihood and successfully applied this inferential methodology to the class of Autoregressive Ordered Probit (AOP) models. Similarly, Cui (2009) developed effects methods for stationary time series of integer counts with binomial, Poisson, geometric or any other discrete marginal distribution.

Hyndman (1999) considers three possible nonparametric additive effects models for binary time series prediction which allow for autocorrelation: a generalization of an AutoRegressive with eXogenous terms (ARX) model to allow explanatory variables, two generalizations of a regression model with AR errors which allows nonparametric additive covariates. The generalization of an ARX model provides easy marginal interpretation of the effect of covariates and inference can be obtained using the techniques developed for generalized additive models (GAMs). The potential usefulness was then illustrated on simple real data applications.

4.1.2 Regression Methods

Most of the work done in temporal data mining focuses on the regression task, where special models for time-series data have been developed and used in the last 60 years. Here, we present the most relevant ones, stressing the autocorrelation aspect.

Very similar to the SAR models described in Section 4.2.2, autoregressive models (AR) has been well used to measure temporal autocorrelation. The AR model is a linear regression of current values of a series against one or more prior values of the series. The AR observation at time t can be decomposed and becomes a linear combination of all previous input variables.

In particular, Bullmore et al (1996) proposed an autoregressive model of order 1 (AR(1)) to model temporal autocorrelation in linear modeling of functional magnetic resonance imaging (FMRI) time series data. The autocorrelation and the noise are estimated from the residuals after fitting the model and are used to create a filter that is applied to the data before re-fitting the model. Such autoregressive model is defined as:

$$\hat{Y}_t = (1 - \sum_{i=1}^{p} \phi_i)\mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \varepsilon_t \tag{4.1}$$

where ϕ_1 through ϕ_p are the coefficients of the time series Y_t, μ denotes the overall mean and ε_t is a purely random stationary process with mean μ_ε and variance σ_ε^2.

An advantage of such models is that they produce the most efficient estimation of the model parameters under the normality assumption. Moreover, an enormous advantage that the given AR relations, i.e., relations between autocorrelation function, model parameters and reflection coefficients (negative of the partial correlation (difference between the autocorrelation coefficient at a certain lag and its extrapolation from lower order correlations)) are also useful, when they are applied to estimated parameters and to estimated autocorrelation functions. However, a disadvantage is that they can produce biased parameter estimates if the autocorrelation is not estimated accurately (i.e., do not necessarily produce "minimum" bias estimators).

Likewise, AutoRegressive Moving Average (ARMA) and AutoRegressive Integrated Moving Average (ARIMA) models (Box et al, 1994) have been used to deal with autocorrelation in different types of time series data. Generally, the ARMA/ARIMA model for a time series is defined as:

$$\hat{Y}_t = \sum_{i=1}^{M} a_i Y_{t-i} + \sum_{i=1}^{N} b_i E_{t-i} + \varepsilon_t \tag{4.2}$$

where the first summation is the autoregressive part (AR), the second summation is the moving average part (MA), whereas the last part is an error term that is assumed to be a random variable sampled from a normal distribution.

The autoregressive part consists of weighted past observations and the moving average part consists of weighted past estimation errors. In other words, it is the difference between the actual value and the forecasted value in the previous observation.

The ARMA autocovariance function is a convolution of the separate autocovariances of the AR and the MA parts. The ARMA autocorrelation function is the ration of the autocovariance function and the autocovariance $r(0)$ at lag 0.

The difficult question that arises in this case is how many lags to include in the regression equation. Autocorrelation could be a guide as to how many lags to include i.e, high magnitude of autocorrelation indicates a lag that should be included in the regression equation. Thus, the autocorrelation function at higher lags is completely determined by the AR part.

ARIMA models require lengthy time series and the impulse-response function specification aligns time lags of a response variable with those of a covariance. In this case, the response variable is an attribute variable at some location (i.e., a location specific time series), and the covariances are the same attribute variable at other, frequently nearby, locations. This specification helps uncover lead and lag locations for some time series process and is especially informative for spatial diffusion processes.

These standard ARMA family of linear models are designed for stationary time series, although they can also be applied to non-stationary ones, after the latter are transformed to stationary time series or some suitable variant of the process (e.g. differences between successive terms) is assumed to be stationary. Another popular work-around for non-stationarity is to assume that the time series is piece-wise (or locally) stationary. The series is then broken down into smaller "frames" within each of which, the stationarity condition can be assumed to hold and then separate models are learnt for each frame.

4.2 Predictive Methods that use Spatial Autocorrelation

In the following section, we give an overview of the methods used in spatial data mining, as well in spatial statistics and modeling. Since our main focus are decision trees for classification and regression tasks, we give an overview of the existing data mining studies for classification and regression task that consider the effect of spatial autocorrelation.

4.2.1 Classification Methods

One of the first works that recognized the importance of considering spatial autocorrelation in spatial data mining, focusing on the classification task, is presented by Huang et al (2004). The authors propose and empirically validate methods based on logistic regression and Bayesian classification that explicitly take the spatial dimension into account. In particular, the algorithm that they propose includes a novel multi-resolution filter, which exploits the spatial auto-correlation property of spatial data to effectively reduce the search space. This means that, when the locations of the features tend to be spatially clustered, which is often true for spatial data due to spatial-autocorrelation, the computation cost of the algorithm can be significantly reduced.

In addition, a new interest measure for spatial co-location patterns is proposed. This measure is closely related to the cross-correlation function, which is often used as a statistical measure of independence of moving patterns, i.e., in this case it is a measure of the interactions among pairs of spatial features. Furthermore, it possesses an order-preserving property which can be exploited for computational efficiency.

However, this method considers only Boolean features and the choice of neighbor relation which impacts the performance of the proposed algorithms is not evaluated. Moreover, the method is not tested on real world datasets, especially ones that exhibit strong spatial autocorrelation.

Zhao and Li (2011) have proposed a spatial entropy-based decision tree that differs from a conventional tree in the way that it considers the phenomenon of spatial autocorrelation in the classification process adapted to geographical data. A spatially-tailored formulation of the traditional entropy measure (i.e., "spatial entropy" (Li and Claramunt, 2006)) is used in the tree induction. The measure evaluates the dispersion of the entropy measure over some neighborhoods by looking for a split which minimizes the intra-distance computed for the examples in the majority class and maximizes the inter-distance between the examples the majority class and the examples in different classes. This measure is presented in more details in Section 3.2.2.

One of the disadvantages of this method lies in the full replacement of the classical with a spatial entropy measure that is supposed to take into account the influence of space and spatial autocorrelation which may be problematic in many real applications where the effect of these two autocorrelation is not symmetric and non-stationary along the studied area. Another disadvantage is there the method has been tested only on three study areas which does not guarantee general applicability. The method has not been compared to other existing methods that also take autocorrelation into account nor to state-of-the-art predictive modeling methods. Moreover, it cannot be extended to meet the regression task, as it is designed to satisfy only single classification tasks.

A different formulation of spatially-aware entropy is provided by Rinzivillo and Turini (2004, 2007). For a given reference object they described its neighborhood by considering the attribute of the object itself and the objects related by the chosen relations. The resulting spatial transactions may be either considered like “traditional” transactions, by considering only the qualitative spatial relations, or their spatial extension can be exploited during the data mining process. They then propose to compute the entropy for a spatial measurement of each example and to use the information gain based on such spatial entropy measure for the induction of spatial decision trees. In particular, the spatial entropy is computed for each weighted sum of the spatially related (e.g., overlapping) examples. For more details, this measure is presented in Section 3.2.2. At the end, the method is implemented on top of a GIS tool and tested by analyzing real world data.

The limitations of this method lie in the facts that it is designed in a tight connection to a GIS tool which requires that data is represented in form of GIS layer or represented within a GIS database and that is designed to deal with single classification tasks.

Bel et al (2009) adapted the Breiman’s classification trees (Breiman et al, 1984) to the case of spatially dependent samples, focusing on environmental and ecological applications. They modified the algorithm to take into account the irregularity of sampling by weighting the data according to their spatial pattern.

Two approaches were considered: The first one takes into account the irregularity of the sampling by weighting the data according to their spatial pattern using two existing methods based on Voronoi tessellation and regular grid, and one original method based on kriging. The second one uses spatial estimates of the quantities involved in the construction of the discriminant rule at each step of the algorithm. These methods were tested on simulations and on a classical dataset to highlight their advantages and drawbacks.

However, the methods presented by Bel et al (2009) is specially designed to satisfy single classification tasks. In addition, the presented empirical evaluation does not include statistical significance tests to support the main conclusion of the study that are based on three experimental studies.

Finally, the problem of dealing with autocorrelation in mining spatial data has been also addressed in multi-relational data mining. For example, Ceci and Appice (2006) propose a spatial associative classifier that learns, in the same learning phase, both spatially defined association rules and a classification model (on the basis of the extracted rules).

In particular, they consider two alternative solutions for associative classification: a propositional and a structural method. In the former, the classifier obtains a propositional representation of training data even in spatial domains which are inherently non-propositional, thus allowing the application of traditional data mining algorithms. In the latter, the Bayesian framework is extended following a multi-relational data mining approach in order to cope with spatial classification tasks.

Both methods were evaluated and compared on two real-world spatial datasets. The obtained results show that the use of different levels of granularity permitted to find the best tradeoff between bias and variance in spatial classification. Moreover, spatial data mining applications benefit by preserving information about how data were originally structured in the spatial domain. This justifies the improved accuracy of the structural approach with respect to the propositional one. On the contrary, the propositional approach outperforms the structural one in terms of efficiency due to the reduction of the search space.

In addition, Malerba et al (2005) present a multi-relational clustering algorithm (CORSO) that expresses the spatial structure of data by resorting to the First Order Logic representation formalism and uses learning in the Normal ILP setting in order to take into account the autocorrelation in the data. In this case, the autocorrelation is implicit in spatial relations, such as overlap, to_right, to_left, between spatial objects to be clustered. In the clustering phase, the similarity of multi-relational descriptions is used in order to identify spatial objects that are in the neighborhood.

However, CORSO can handle only binary relations. The number of obtained clusters is not optimal and there can exist many similar cluster that need to be merged a-posteriori due to their similar properties. Concerning the cluster construction, the detection of initial set of candidate objects to become cluster can sometimes become restrictive. Furthermore, in some case CORSO cannot detect natural clusters in cases when not all the neighbors of a particular object belong to the cluster.

Furthermore, the problem of dealing with autocorrelation in mining spatial data has been also addressed in the multi-relational transduction setting. A relational approach to spatial classification in a transductive setting has been proposed by Ceci et al (2010). Transduction is also motivated by the contiguity of the concept of positive autocorrelation, which typically affect spatial phenomena, with the smoothness assumption which characterize the transductive setting.

In particular, a relational upgrade of the Naive Bayes classifier is proposed as discriminative model, an iterative algorithm is designed for the transductive classification of unlabeled data, and a distance measure between relational descriptions of spatial objects is defined in order to determine the k-nearest neighbors of each example in the dataset. In principle, a strong spatial autocorrelation is expected to counterbalance the lack of labelled data, when a relational classifier which takes spatial autocorrelation into account is learned in a transductive setting.

However, the obtained models can handle only stationary autocorrelation and cannot be applied in the same domain in different context.

4.2.2 Regression Methods

Most theoretical research in Statistics and Econometrics exploits the so called “spatial autoregressive” (SAR) model in order to measure autocorrelation in a “lattice”. More formally, the spatial autoregressive model is defined as:

$$\hat{e}_i = \rho \sum_{j=1}^{N} w_{ij}\, e_j + \varepsilon_i \qquad i = 1, \ldots, N \tag{4.3}$$

where N is the number of examples in a training set, $e_j = Y_j - \overline{Y}$ is the prediction residual (where prediction is based on the average), w_{ij} is the weight of the spatial proximity between the pair of examples i and j, ρ is a parameter that expresses the spatial dependence in the lattice and the error ε_i follows a normal distribution.

As recognized by Li et al (2007), in order to informally assess the strength of the spatial dependence, exploratory data analysis should be based on estimating ρ in the autoregressive model (see Equation 4.3). This means that the parameter ρ plays a crucial role in representing the autocorrelation in the data.

One common solution to estimate ρ is to use a modified least squares estimator, which is the solution to the following quadratic equation in ρ:

$$\mathbf{e}^T(\mathbf{I} - \rho\mathbf{W})^T\mathbf{W}(\mathbf{I} - \rho\mathbf{W})\mathbf{e} = 0 \tag{4.4}$$

where $\mathbf{W}$ is the matrix representation of w_{ij}, $\mathbf{I}$ is the identity matrix and $\mathbf{e}$ is the vector of e_i values. Although this estimator is consistent (Li et al, 2007), its computation is not straightforward, since it would require the computation of the Maximum Likelihood Estimator of ρ.

In a theoretical study, LeSage and Pace (2001) use the Maximum Likelihood Estimator to take into account autocorrelation in data. They stress that the presence of spatial dependence requires an appropriate treatment of spatial correlation effects. However, the computation of the Maximum Likelihood Estimator is impractical when large datasets need to be processed (Li et al, 2007).

Therefore, instead of computing an estimate of ρ, the Pearson's correlation coefficient and its variants, as well as entropy based measures, are commonly used in spatial data mining applications (LeSage and Pace, 2001). Especially because in the presence of spatial dependence, appropriate treatment of spatial autocorrelation effects is critical if these hidden relations are to be found.

Another way to quantify spatial autocorrelation in a regression model is to introduce an autoregressive parameter in the model specification of a CAR (Conditional AutoRegressive) model (Li et al, 2007). In contrast to SAR model that take into account the global spatial autocorrelation, CAR model is more appropriate for situations with first order dependency or relatively local spatial autocorrelation.

The CAR model assumes a limited spatial dependency, i.e., the state of a particular area is influenced by its neighbors and not neighbors of neighbors. This is due to the different way it specifies the variance-covariance matrixes in comparison to SAR. Thus, CAR provides a simpler way to model locally autocorrelated geo-referenced areal data.

Moreover, the CAR model specifies spatial autocorrelation as being in the error term, with a weaker degree and smaller spatial field that the SAR model, and with the attribute error at location i being a function of the sum of nearby error values. The parameters of a SAR model can also be estimated using maximum likelihood techniques.

Another standard way to take into account spatial autocorrelation in spatial statistics is Geographically Weighted Regression (GWR) (Fotheringham et al, 2002). GWR assumes a local form of the regression surface depending on the site (u,v). The linear regression model is extended by weighting each training observation in accordance with its proximity to point (u,v), so that the weighting of an example is no longer constant in the calibration but varies with (u,v). The local model for a response attribute y at site (u,v) takes the following form:

$$y(u,v) = \alpha_0(u,v) + \sum_k \alpha_k(u,v)x_k(u,v) + \varepsilon_{(u,v)} \tag{4.5}$$

where $\alpha_0(u,v)$ is the initial realization of the continuous function $\alpha_k(u,v)$, $\alpha_k(u,v)$ is a realization of the continuous function $\alpha_k(u,v)$ at point (u,v), $x_k(u,v)$ is the values of the target variable at point (u,v), while $\varepsilon_{(u,v)}$ denotes random noise. Each coefficient α_k is locally estimated at location (u,v) from measurements close to (u,v).

$$\alpha(u,v) = (\mathbf{X}^T\mathbf{W}_{(u,v)}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}_{(u,v)}y \tag{4.6}$$

where $\mathbf{W}_{(u,v)}$ is a diagonal weight matrix which states the influence of each observation for the estimation of $\alpha(u,v)$, defined by functions such as Gaussian or bi-square and $\mathbf{W}_{(u,v)}$ introduces the spatial lag depending on the position (u,v) in the regression model.

In this way GWR takes advantage of positive autocorrelation between neighboring examples in space and provides valuable information on the nature of the processes being investigated. Especially, this can be beneficiating if the explored autocorrelation is linear. GWR is a well-used spatial statistical methods that is implemented within a lot of standard GIS software.

However, GWR is very resource expensive and time consuming in comparison with the traditional regression methods. The models are dependent of the location and cannot be generalized or applied to another experimental setup or study area. Therefore, the GWR method is more suited for interpolation purposes, for spatial mapping within a GIS environment and similar geographical applications. Another disadvantage of this technique is that the models it builds are linear and cannot capture nonlinearity which is very common property of spatial data.

Kriging (Bogorny et al, 2006) is another spatial regression technique which explicitly takes advantage of autocorrelation. It applies an optimal linear interpolation method to estimate unknown response values $y(u,v)$ at each site (u,v). $y(u,v)$ is decomposed into three terms: a structural component, which represents a mean or constant trend, a random but spatially correlated component, and a random noise component, which expresses measurement errors or variation inherent to the attribute of interest.

In its simplest form, a kriging estimate of the field at an unobserved location is an optimized linear combination of the data at the observed locations. The coefficients of the kriging estimate and the associated error measure both depend on the spatial configuration of the data, the unobserved location relative to the data locations, and spatial correlation or the degree to which one location can be predicted from a second location as a function of spatial separation.

Similarly to GWR, kriging is also an resource expensive and time consuming technique which is quite difficult to apply. It is mostly suited for interpolation purposes. Its models are dependent of the location and cannot be generalized or applied to another experimental setup or study area. However, in contrast to GWR and other methods that we have already mentioned, kriging associates a probability with its predictions. In the case of two or more target variables, Ordinary Cokriging, Universal Cokriging, Simple Cokriging, Indicator Cokriging, Probability Cokriging, and Disjunctive Cokriging can be considered as multivariate extensions of the different types of kriging.

Finally, the problem of dealing with autocorrelation in mining spatial data has been also addressed in multi-relational data mining. Malerba et al (2005) present a relational regression method (Mrs-SMOTI) that captures both global and local spatial effects the predictive attributes, while building a regression model tightly integrated with a spatial database. The method deals with spatial autocorrelation on the input side (explanatory variables). It considers the geometrical representation and relative positioning of the spatial objects to decide the split condition for the tree induction (e.g., towns crossed by any river and towns not crossed by any river). For the splitting decision a classical heuristic based on error reduction is used.

The advantages of the proposed method lie in its tight connection to a spatial database and its ability to captures both global and local spatial effects the autocorrelation phenomenon. This gives Mrs-SMOTI the opportunity to mine both spatial relationships and spatial attributes which are implicit in spatial data. Indeed, this implicit information is often responsible for the spatial variation over data and it is extremely useful in regression modeling.

Moreover, the search strategy is modified in order to mine models that capture the implicit relational structure of spatial data. This means that spatial relationships (intra-layer and inter-layer) make possible to consider explanatory attributes that influence the response attribute but do not necessarily come from a single layer. In particular, intra-layer relationships make available spatially lagged response attributes in addition to spatially lagged explanatory attributes.

However, the obtained models are very specific and depend on the geographical coordinates of the considered objects. Therefore, the models cannot be applied for different tasks in the same domain.

However, when resorting to multi-relational data mining, it is possible that the presence of autocorrelation in spatial phenomena can bias feature selection (Jensen and Neville, 2002). In particular, the distribution of scores for features formed from related objects with concentrated linkage (i.e., high concentration of objects linked to a common neighbor) has a surprisingly large variance when the class attribute has high autocorrelation. This large variance causes feature selection algorithms to be biased in favor of these features, even when they are not related to the class attribute, that is, they are randomly generated. In this case, conventional hypothesis tests, such as the χ^2-test for independence, which evaluate statistically significant differences between proportions for two or more groups in a dataset, fail to discard uninformative features.

4.3 Predictive Methods that use Spatio-Temporal Autocorrelation

In the following section, we give an overview of the methods used in spatio-temporal data mining, noting that at the moment, research accommodating both spatial and temporal data mining is sparse. Focusing on the classification and regression tasks, we give an overview of the existing studies that consider and incorporate spatio-temporal autocorrelation and describe their main characteristics.

4.3.1 Classification Methods

To the best of our knowledge, there exist no previous works on classification that consider spatio-temporal autocorrelation. However, we present two directions of research work on regression that that consider spatio-temporal autocorrelation of a binary target variable. The first direction tries to embed a temporal awareness in spatial methods, while the second direction accommodates the space dimension into temporal data mining methods. Here, we give a short overview on several related works that use spatio-temporal autocorrelation, following these directions.

One of the works that tries to embed the temporal awareness in a spatial method is presented by Zhu et al (2005), where a temporal component is added to an autologistic regression model for spatial-temporal binary data. An autologistic regression model consists of a logistic regression of a response variable on explanatory variables and an auto-regression on responses at neighboring locations on a lattice.

The obtained spatial-temporal autologistic regression model is a Markov random field with pairwise spatial dependence. It is a popular tool for modeling spatial binary responses since is able to capture the relationship between a binary response and potential explanatory variables, while adjusting for both spatial dependence and temporal dependence simultaneously by a space-time Markov random field. The model parameters are estimated by maximum pseudo-likelihood.

Finally, the predictions of future responses on the lattice are obtained by using a Gibbs sampler. This method is then applied to study the outbreaks of southern pine beetle in North Carolina, where the explanatory variables are time-invariant, including the weather information.

Similarly, Zhu et al (2008) tried to embed the temporal awareness in a spatial method, where a temporal component is added to an autologistic regression model for spatial-temporal binary data, extending their previous study (Zhu et al, 2005).

For a given time point t, the response variable Y_t is assumed to follow a Markov random field, under a preselected spatial neighborhood structure. That is,

$$p(Y_{i,t} \| Y_{j,t} : j \neq i, Y_{t'} : t' = t-1, t-2, ..., t-S) = p(Y_{i,t} \| Y_{j,t} : j \in N_i, Y'_t : t' = t-1, t-2, ..., t-S) \quad (4.7)$$

where N_i consists of the indexes of the neighbors of the i-th site.

The binary data was measured on a spatial lattice and repeatedly over discrete time points. A Monte Carlo was used to obtain maximum likelihood estimates of the model parameters and predictive distributions at future time points. Path sampling was used in order to estimate the unknown normalizing constant and approximate an information criterion for model assessment.

This methodology was then applied to study the outbreaks of southern pine beetle in western Canada. The performed data analysis supports their initial hypothesis that there is a significant relationship between beetle outbreaks and abiotic factors such as summer and winter temperatures and elevation. A plausible explanation for the strong spatial and temporal dependence is the dispersal of the beetle populations across space and over time, as during beetle outbreaks, beetle populations are capable of dispersal at the scale of hundreds of kilometers.

4.3.2 Regression Methods

Most of the work done in spatio-temporal data mining focuses on the regression task. Here, we present the most relevant ones, stressing the autocorrelation aspect. We note that all current models assume that the spatio-temporal autocorrelation in data can be adequately described by globally fixed parameters, i.e., the extent to which observations are autocorrelated with each other is fixed in space and/or time.

For example, in Spatio-Temporal AutoRegressive Integrated Moving Average (STARIMA) models, space-time processes are stationarized through transformation and differencing and autocorrelation is accounted for in the autoregressive (AR) and moving average (MA) terms. The AR and/or MA orders are fixed globally both spatially and temporally, and a single parameter is estimated for each. STARIMA can be seen as an extension of the ARIMA model for time-series (See Section 4.1.2).

Moreover, the space-time autocorrelation function, defined in Section 3.3.1, has been used in STARIMA to calibrate the order of moving average, which define the range of spatial neighborhoods that contribute to the current location at a specific time lag (Pfeifer and Deutsch, 1980) .

Although STARIMA has been employed in a number of spatio-temporal studies, for example in traffic flow forecasting (Kamarianakis and Prastacos, 2005; Wang et al, 2010), the two assumptions of stationarity and fixed spatio-temporal neighborhoods are in fact very difficult to be satisfied for very common real-world data and applications. For example, this is the case of dynamic network data (Cheng et al, 2011).

Embedding of a temporal awareness in spatial methods and accommodation of space into temporal data mining methods are very frequent when dealing with the regression task. For example, Griffith and Heuvelink (2009) used a spatial Vector AutoRegressive (VAR) model which includes spatial as well as temporal lags among a vector of stationary state variables in order to capture joint spatial and temporal dynamics, assuming that spatial data consists of a small spatial series with a lengthy time series. The spatial data can have the full spatial covariance matrix estimated by exploiting the time dimension and using it to establish asymptotes. An estimated spatial VAR model can be then used to calculate impulse responses between variables over time and across space.

Similarly, a mixed spatio-temporal model containing both fixed and random effect terms is presented by Baltagi (2005), where the random term is spatially structured in order to account for both temporal and spatial autocorrelation. The spatially structured component accounts for spatial autocorrelation, whereas the sum of the spatially structured and unstructured components accounts for temporal autocorrelation.

The random effects can be estimated by treating each locational time series as a set of repeated measures, allowing them to be separated from residuals. Practical situations often constitute quasi-spatial datasets because attributes for the same set of objects within an areal unit are not being measured through time. Rather, attributes for areal unit aggregates of changing collections of objects are being measured. However, some drawbacks care related to data collection issues, i.e., sampling design.

By incorporating temporal effects into the geographically weighted regression (GWR) model, an extended GWR model is introduced by Huang et al (2010). Geographically and Temporally Weighted Regression (GTWR) has been developed to deal with both spatial and temporal non-stationarity simultaneously in real estate market data. Unlike the standard GWR model, GTWR integrates both temporal and spatial information in the weighting matrices to capture spatial and temporal heterogeneity. GTWR shows statistically significant improvement from GWR in terms of goodness-of-fit and other statistical measures, for the sample data used in the study.

As an alternative to GWR, Hardisty and Klippel (2010) employed eigenvector spatial filtering to construct geographically varying regression coefficients. The filtering is a flexible and powerful tool for dealing with correlated data, thus allowing model building to proceed as if the observations are independent. It can be then used within a visualization tool that allows spatio-temporal statistics to be used in an interactive manner and therefore be incorporated in visual analytic. The filtering allows interactions with the spatio-temporal structure uncovered by the statistics that contains a novel coordination strategy. Its capabilities were then demonstrated on a study of H1N1 flu pandemic that resulted with the finding that there was a critical spatio-temporal "inflection point" at which the pandemic changed its character in the United States.

Spatio-temporal kriging (De Iaco et al, 2002) is another regression technique which explicitly takes advantage of autocorrelation. It is fundamentally the same as conventional spatial kriging except in the management of covariance modeling, i.e., it applies an optimal linear interpolation method to estimate unknown response values $y(u,v)$ at each site (u,v). By instinct, one may seek to treat time as if it is an extra dimension in spatial kriging and then lump the lag distances along the spatial and temporal axes into a single Euclidean space-time metric.

However, this has been shown to be theoretically and practically problematic. The key reason is because the time axis is by nature different (i.e., it is not geometric) and not necessarily orthogonal to the other three principal axes. Environmental processes occurring in time almost always have some dependence on processes occurring in space, which accounts for some of the non-orthogonality between the time and spatial axes.

In practice, when dealing with real-world data, space-time variograms (Griffith and Heuvelink, 2009) are fitted by introducing simplifying statistical assumptions. Basically, two main groups of approaches exist: (a) separable (purely spatial and purely temporal models) and (b) non-separable (space-time) approach (Huang et al, 2007; Ma, 2005). Similar as STARIMA, the purpose of using these models is to quantify the spatial and temporal dependence reflected in a data set. The resulting model might then be used for spatial interpolation and/or temporal prediction.

Different measure of spatio-temporal autocorrelation have been incorporated in several spatio-temporal data mining studies. For example, a spatial-temporal autocorrelation measure has been adapted by Zhang et al (2003) to efficiently process similarity based range queries and joins and take autocorrelation into account when retrieving spatial time series.

Yue and Yeh (2008) empirically demonstrated that the cross-correlation function *CCF* (Box et al, 1994) can be used to determine the spatio-temporal relationship between a road link and its neighbors. This is, however, dependent on sufficient spatial and temporal resolution in the data. In such traffic networks, a peak at lag zero, for example, indicates that the current resolution does not capture the direction of influence of one location on another, but the locations behave very similarly at the same time. This usually happens when the network is highly congested in the "morning" peak.

Moreover, Cheng et al (2011) provide an exploratory spatio-temporal autocorrelation analysis that examines the spatio-temporal autocorrelation structure of road networks in order to determine likely requirements for building a suitable spatio-temporal forecasting model. They use two indices: the ST-ACF (Pfeifer and Deutsch, 1980) that measures global spatio-temporal autocorrelation and the cross-correlation function *CCF* (Box et al, 1994) that measures local spatio-temporal autocorrelation between two locations. These indices are extensions of the temporal autocorrelation function (see Equation 3.1) and were selected as they are easily interpretable and have a practical application in established spatio-temporal modeling frameworks.

When resorting to the relational data mining, McGovern et al (2008) introduced a learning approach in spatially and temporally varying relational data. The approach is based on the decision trees that can vary in both space and time called Spatio-Temporal Relational Probability Trees (SRPTs). Details on how this algorithm deals with autocorrelation are not provided.

The SRPT learning algorithm directly addresses the difficulties inherent in both the conceptualization of the data and in learning the model and deals with the exponential increase in search complexity through sampling. It focuses its distinctions on the temporal nature of the data rather than on the spatial nature of the data. The spatial distinctions arise from the spatial relationships defined in the data.

SRPT learning algorithm can identify the critical relations and temporal variations of the attributes on each relation, but it cannot currently identify spatial or spatiotemporal variations, such as the increase in positive tilting as a function of altitude (and possibly time). The method is validated using a simulated data set and empirically tested on two real-world data sets.

4.4 Predictive Methods that use Network Autocorrelation

In the recent years, numerous algorithms have been designed for modeling a partially labeled network and providing estimates of unknown labels associated with nodes. In the following Section, we give a short overview of the existing methods that consider networks autocorrelation, focusing on the classification and the regression task.

4.4.1 Classification Methods

In the literature, significant attention has been given to the task of classification in network domains and the development of classification methods that consider network autocorrelation.

In general, network learning assumes that data for inference are already in the form of a network and exploit the structure of the network to allow the collective inference. Collective inference targets learning algorithms where various interrelated values are inferred simultaneously such that estimates of neighboring labels influence one another (Gallagher et al, 2008; Macskassy and Provost, 2007; Sen et al, 2008). Since exact inference is known to be an NP-hard problem and there is no guarantee that data network satisfy the conditions that make exact inference tractable for collective learning, most of the research in collective learning has been devoted to the development of approximate inference algorithms.

Collective inference algorithms can exploit network (relational) autocorrelation. For example, Macskassy and Provost (2007) show that models that consider only autocorrelation in class labels can perform very well when only a small fraction of the class labels are known.

Popular approximate inference algorithms include iterative inference, Gibbs sampling, loopy belief propagation and mean-field relaxation labeling. An outline of strengths and weakness of these algorithms is reported in Sen et al (2008). In general, one of the major advantages of collective learning lies in its powerful ability to learn various kinds of dependency structures (e.g., different degrees of correlation) (Jensen et al, 2004). However, as pointed out in Neville and Jensen (2007), when the labeled data is very sparse, the performance of collective classification might be largely degraded due to the lack of sufficient neighbors. This is overcome by incorporating informative “ghost edges” into the networks to deal with sparsity issues (Macskassy, 2007; Neville and Jensen, 2007).

Interestingly, learning problems similar to the tasks addressed in network learning have been recently addressed outside the areas of network learning and graph mining. In particular, in the area of semi-supervised learning and transductive learning (Vapnik, 1998) a corpus of data without links is given to the algorithms. The basic idea is to connect data into a weighted network by adding edges (in various ways) based on the similarity between entities and to estimate a function on the graph which guarantees the consistency with the label information and the smoothness over the whole graph (Zhu et al, 2003). The constraint on smoothness implicitly assumes positive autocorrelation in the graph, that is nearby nodes tend to share the same class labels (i.e., homophily).

Rahmani et al (2010) use a transductive learning approach in order to classify the function of proteins in protein-protein interaction (PPI) networks by using only data related to the structure of the networks. The classification task aims at predicting the particular function associated with a node (protein) on four yeast benchmark datasets: DIP-Core, VonMering, Krogan and MIPS.

They introduced a node description formalism that has not been used previously for protein function prediction and which takes global information into account. The use of global information for such methods is unexplored, since existing inductive learning methods for predicting protein functions in PPI networks use local information.

A potential disadvantage of this method is that in large graphs, one gets very high-dimensional descriptions, and not all learners handle learning from high-dimensional spaces well. It is possible, however, to reduce the dimensionality of the vector by only retaining the shortest-path distance to a few "important" nodes. The introduced formalism is used in order reduce the number of features needed for the description.

In fact, Popescul and Ungar (2003) and Hasan et al (2006) have demonstrated that using aggregated relational features in an iterative way can be quite effective not only for the collective classification, but also for the link prediction task. They integrated statistical modeling and feature selection into a search over the space of database queries generating feature candidates involving complex iteration between object in a given relational database. The features selected in this way provide useful insights into the nature of the analyzed problem.

The construction of aggregated features has been also investigated in Grčar and Lavrač (2011), where the authors present an efficient classification algorithm for categorizing video lectures in heterogeneous document information networks. They first transform documents into bag-of-words vectors, then decompose the corresponding heterogeneous network into separate graphs and compute structural-context feature vectors. Finally, they construct a common feature vector space to be used in the mining phase.

A limitation of such methods that use aggregated relational features is that the created features are in a tight connection to the data. Moreover, the obtained models only consider the cases where training and testing data (nodes) belong to the same network. This means that the prediction phase requires complete knowledge of the network arrangement (e.g., connections to other nodes of the network) of any unlabeled node to be predicted.

In the context of social networks, Weng et al (2010) proposed TwitterRank, an algorithm for finding topic-sensitive influential *twitterers*, which is based on the assumptions that the presence of homophily implies that there are *twitterers* who are selective in choosing *friends* to follow.

Here, homophily means that a *twitterer* follows a *friend* because he/she is interested in some topics the *friend* is publishing and the *friend* follows back the *twitterer* if he/she shares similar topics of interest. Therefore, identifying the influential *twitterers* based on "following" relationship is meaningless since it does not carry string indication of influence. On the other hand, the presence of homophily indicates that the "following" relationship between *twitterers* ai related to their topic similarity.

Next, Kwak et al (2010) investigated homophily in two contexts: geographic location and popularity (the number of *r*-friends' followers). They considered the time zone of a user as an approximate indicator for the location of the user and the number of followers as a measure for user's popularity.

Twitter diverges from well-known traits of social networks: its distribution of followers is not power-law, the degree of separation is shorter than expected, and most links are not reciprocated. However, among reciprocated users there is some level of homophily.

4.4.2 Regression Methods

In the literature, less attention has been given to the task of regression and the development of methods that consider network autocorrelation. In the following subsection, we present relevant network regression methods.

Appice et al (2009) address the task of the network regression with a transductive approach that follows the main idea of iterative inference described in Sen et al (2008). Their regression inference procedure is based on a co-training approach according to which separate model trees are learned from the attributes of a node and the attributes aggregated in the neighborhood of the nodes. During an iterative learning process, each of these trees is used to label the unlabeled nodes for the other.

The learning is based on the intuition that when a high network autocorrelation affects the dependent variable, the semi-supervised smoothness assumption is likely to hold and the transductive setting can return better results than the inductive setting. On one hand, this intuition is clearer in spatial domains, where closeness of points corresponds to a spatial distance measure and network autocorrelation is a manifestation of the (positive) spatial autocorrelation. On the other hand, transduction is most useful when the standard i.i.d. assumption is violated.

The learning process is robust to sparse labeling and low label consistency and improves traditional model tree induction across a range of geographical data networks. As with other transductive learning approach, no final model is produced. This idea of using aggregated relational features to enhance the node-wise similarity measurement is popular in network mining and it is not restricted to the transductive setting.

Steinhaeuser et al (2011) follows an approach close to the idea of predictive clustering. In fact, they combine a descriptive data mining task (clustering) with a predictive data mining task (regression) and argue that using networks as a data representation provides a unified framework for identifying and characterizing patterns in climate data. In their case, the network is built a-posteriori on the basis of the values of the Pearson's correlation coefficients between pair of nodes on time series collected for the same variable.

Clustering then boils down to a community detection problem that allows the proposed approach to group interconnected nodes so that the pairwise walking distance between two nodes in the cluster is minimized. Finally, the prediction is obtained by means of a linear regression model built on the cluster to which the nodes to be predicted belong (spatially).

The innovative aspects of our NCLUS method (see chapter 7 for details) with respect to this work are manifold. First, clustering is addressed as a hierarchical task. Second, the correlation is measured on nodes which are interconnected in an existing network, while Steinhaeuser et al (2011) measure the correlation to define virtual edges between examples. Finally, in our idea, clusters are not constrained from the structure of the training network.

5 Learning Predictive Clustering Trees (PCTs)

Our work is based on the concept of learning predictive clustering trees. In this Chapter, we give an overview of the predictive clustering trees framework, focusing on the different predictive task that it can handle, from standard classification and regression tasks to multi-target classification and regression, as well as hierarchial multi-label classification, as a special case of predictive modeling with structured outputs.

5.1 The Algorithm for Building Predictive Clustering Trees (PCTs)

In this section, we discuss the concept of predictive clustering and give a shot overview of the Predictive Clustering Trees (PCTs) framework.

In literature, clustering and predictive modeling are treated as two separate techniques. The connection between them is usually made by machine learning methods that partition the instances into subsets, such as decision trees and decision rules. These methods can be considered both as predictive and as clustering methods (Langley, 1996).

The concept of predictive clustering was first introduced by (Blockeel and Struyf, 2002). Predictive clustering combine elements from both prediction and clustering. As in clustering, clusters of data that are similar to each other are identified, but in addition a predictive model is associated to each cluster. New instances are assigned to clusters based on the cluster descriptions. The predictive model provides a prediction for the target variable of new examples that are recognized to belong to the cluster. Predictive clustering is similar to conceptual clustering (Michalski and Stepp, 2003) since, besides the clusters themselves, it also provides symbolic descriptions (in the form of conjunctions of conditions) of the constructed clusters. However, in contrast to conceptual clustering, predictive clustering is a form of supervised learning.

The 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 (Blockeel and Struyf, 2002) [1].

PCTs can be induced with a standard "top-down induction of decision trees" (TDIDT) algorithm (Breiman et al, 1984): It takes as input a set of training instances (E) and outputs a tree (Algorithm 1).

The algorithm searches for the best acceptable test that can be put in a node. If such a test can be found, then the algorithm creates a new internal node and calls itself recursively to construct a subtree for each subset (cluster) in the partition induced by the test on the training instances. To select the best test, the algorithm scores the tests by the heuristic, i.e., the reduction in variance (defined below) they induce on the instances. Maximizing variance reduction maximizes cluster homogeneity and improves predictive performance. If no acceptable test can be found, that is, if no test significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a representative case, or prototype, of the given instances.

The main loop (Algorithm 1, lines 6-10) searches for the best attribute-value test v^* that can be associated to a node t. It associates the best test v^* with the internal node t and calls itself recursively to

[1]The CLUS system is available for download at `http://sourceforge.net/projects/clus/`.

Algorithm 1 *Top-down induction of CLUS.*

```
1:  procedure PCT($E$) returns tree
2:  if stop(E) then
3:      return leaf(Prototype($E$))
4:  else
5:      $(v^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6:      for each Boolean test $v$ do
7:          $\mathcal{P}$ = partition induced by $v$ on $E$
8:          $h = (E, \mathcal{P})$
9:          if $(h > h^*)$ then
10:             $(v^*, h^*, \mathcal{P}^*) = (v, h, \mathcal{P})$
11:         end if
12:     end for
13:     for each $E_k \in \mathcal{P}^*$ do
14:         $tree_k$ = PCT($E_k$)
15:     end for
16:     return node($v^*$, $\bigcup_k \{tree_k\}$)
17: end if
```

construct a subtree for each subset (cluster) in the partition P^* induced by v^* on the training example set.

5.2 PCTs for Single and Multiple Targets

In this section, we discuss the predictive clustering trees algorithm that deals with the standard classification and regression tasks, as well as more complex tasks of multi-target classification and regression. The PCTs that are able to predict single discrete/continuous target are called single target decision trees (STDTs), whereas the ones that are able to predict multiple (tuple of discrete/continuous variables) target simultaneously are called multiple targets decision trees (MTDTs). We consider both STDTs and MTDTs and we refer to as PTCs.

The task of mining predictive clustering trees (PCTs), for single and multiple targets, can be formalized as follows:
Given:

- A descriptive space X that consists of tuples of values of primitive data types (boolean, discrete or continuous), i.e., $\mathbf{X} = \{X_1, X_2, \ldots X_m\}$ spanned by m independent (or predictor) variables X_j,
- A target space $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_q\}$ spanned by q dependent (or target) variables Y_j,
- A set T of training examples, (x_i, y_i) with $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$

Find: a tree structure τ which represents:

- A set of hierarchically organized clusters on T such that for each $u \in T$, the clusters $C_{i_0}, C_{i_1}, \ldots, C_{i_r}$ exist for which $u \in C_{i_r}$ and the containment relation $C_{i_0} \supseteq C_{i_1} \supseteq \ldots \supseteq C_{i_r}$ is satisfied. Clusters $C_{i_0}, C_{i_1}, \ldots, C_{i_r}$ are associated to the nodes $t_{i_0}, t_{i_1}, \ldots, t_{i_r}$, respectively, where each $t_{i_j} \in \tau$ is a direct child of $t_{i_{j-1}} \in \tau$ $(j = 1, \ldots, r)$ and t_{i_0} is the root.

- A predictive piecewise function $f : \mathbf{X} \to \mathbf{Y}$, defined according to the hierarchically organized clusters. In particular,

$$\forall u \in \mathbf{X},\ f(u) = \sum_{t_i \in leaves(\tau)} D(u,t_i) f_{t_i}(u) \tag{5.1}$$

where $D(u,t_i) = \begin{cases} 1 & \text{if } u \in C_i \\ 0 & \text{otherwise} \end{cases}$

and $f_{t_i}(u)$ is a (multi-objective) prediction function associated to the leaf t_i.

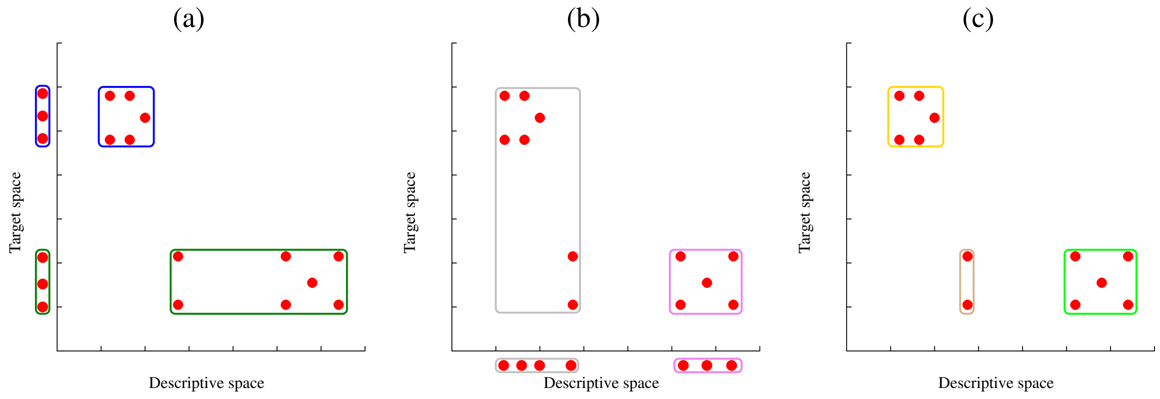


Figure 5.1: *An illustration of predictive clustering.* Illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and descriptive spaces. Note that the target and descriptive spaces are presented here as one-dimensional axes for easier interpretation, but can be actually of higher dimensionality. Figure taken from (Blockeel, 1998).

Clusters are identified according to both the descriptive space and the target space $\mathbf{X} \times \mathbf{Y}$ (Figure 5.1(c)). This is different from what is commonly done in predictive modeling (Figure 5.1(a)) and classical clustering (Figure 5.1(b)), where only one of the spaces is generally considered.

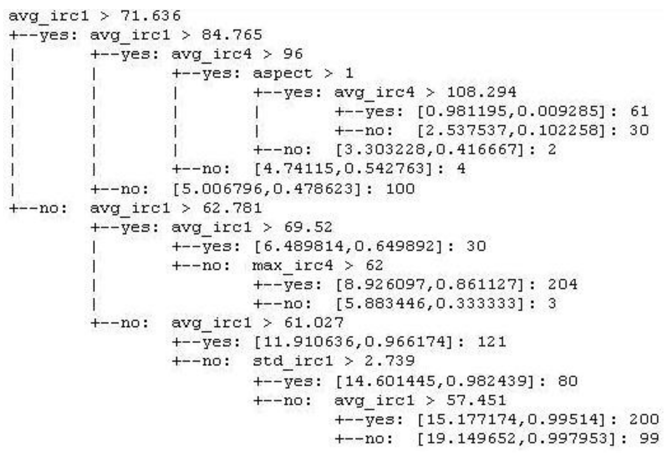
We can now proceed to describe the top-down induction algorithm for building PCTs for single and multiple discrete and continuous targets. It is a recursive algorithm which takes as input the set of example and the function $\eta : V \mapsto \mathbf{X} \times \mathbf{Y}$ and partitions the set of nodes V until a stopping criterion is satisfied.

The construction of PCTs is not very different from that of standard decision tree (see, for example, the C4.5 algorithm proposed by (Quinlan, 1993)): at each internal node t, a test has to be selected according to a given evaluation function. The main difference is that PCTs select the best test by maximizing the (inter-cluster) variance reduction, defined as:

$$\Delta_Y(C,\mathcal{P}) = Var_Y(C) - \sum_{C_k \in \mathcal{P}} \frac{|\,C_k\,|}{|\,C\,|} Var_Y(C_k) \tag{5.2}$$

where C represents the cluster associated to t and $\mathcal{P}$ defines the partition $\{C_1, C_2\}$ of C. The partition is defined according to a Boolean test on a predictor variable in $\mathbf{X}$. By maximizing the variance reduction, the cluster homogeneity is maximized, improving at the same time the predictive performance. $Var_Y(C)$ is the variance computed on the Y variable (class) in the cluster C.

If the variance $Var(\cdot)$ and the predictive function $f(\cdot)$ are considered as parameters, instantiated for the specific learning task at hand, it is possible to easily adapt PCTs to different domains and different tasks. The PCT framework allows different definitions of appropriate variance functions for different types of data and can thus handle complex structured data as targets.



```
avg_irc1 > 71.636
+--yes: avg_irc1 > 84.765
|       +--yes: avg_irc4 > 96
|       |       +--yes: aspect > 1
|       |       |       +--yes: avg_irc4 > 108.294
|       |       |       |       +--yes: [0.981195,0.009285]: 61
|       |       |       |       +--no:  [2.537537,0.102258]: 30
|       |       |       +--no:  [3.303228,0.416667]: 2
|       |       +--no:  [4.74115,0.542763]: 4
|       +--no:  [5.006796,0.478623]: 100
+--no:  avg_irc1 > 62.781
        +--yes: avg_irc1 > 69.52
        |       +--yes: [6.489814,0.649892]: 30
        |       +--no:  max_irc4 > 62
        |               +--yes: [8.926097,0.861127]: 204
        |               +--no:  [5.883446,0.333333]: 3
        +--no:  avg_irc1 > 61.027
                +--yes: [11.910636,0.966174]: 121
                +--no:  std_irc1 > 2.739
                        +--yes: [14.601445,0.982439]: 80
                        +--no:  avg_irc1 > 57.451
                                +--yes: [15.177174,0.99514]: 200
                                +--no:  [19.149652,0.997953]: 99
```

Figure 5.2: *An example of a multi-target regression tree.* The image presents a multi-target regression tree where the splits are binary and the predictions of the two targets are given in brackets in each node of the tree (Stojanova, 2009).

To construct a single classification tree, the variance function $Var(\cdot)$ returns the Gini index of the target variable of the examples in the partition E (i.e., $Var(E) = 1 - \sum_{y \in Y} p(y,E)^2$, where $p(y,E)$ is the probability that an instance in E belongs to the class y), whereas the predictive function of a nominal target variable is the probability distribution across its possible values.

To construct a single regression tree, the variance function $Var(\cdot)$ returns the variance of the response values of the examples in a cluster E (i.e., $Var(E) = Var(Y)$), whereas the predictive function is the average of the response values in a cluster.

The instantiation of the variance and prototype functions for the multiple targets regression trees is done as follows. In the case of multiple targets classification trees, the variance function is computed as the sum of the Gini indexes of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Gini(E,Y_i)$. Furthermore, one can also use the sum of the entropies of class variables as variance function, i.e., $Var(E) = \sum_{i=1}^{T} Entropy(E,Y_i)$ (this definition has also been used in the context of multi–label prediction (Clare, 2003)). The prototype function returns a vector of probabilities that an instance belongs to a given class value for each target variable. Using this probability, the majority class for each target attribute can be calculated.

In the case of multiple targets regression trees, the variance is calculated as the sum of the variances of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Var(Y_i)$. The variances of the targets are normalized, so each target contributes equally to the overall variance. The prototype function (calculated at each leaf) returns as a prediction a vector of the mean values of the target variables. The prediction is calculated using the training instances that belong to the given leaf.

In addition to these instantiations of the variance function for classical classification and regression problems, the CLUS system also implements other variance functions, such as reduced error, information gain, gain ratio and m-estimate.

Finally, the algorithm evaluates all possible tests to be put in a node. If no acceptable test can be found, that is, if no test significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a representative case, or prototype, of the given instances.

Figure 5.2 gives an example of multi target regression tree. The splits are binary and the predictions of the two targets are given in brackets in each node of the tree. In particular, it presents an example dataset for outcrossing rate prediction. The descriptive variables describe different environmental and ecological properties of the study area, while the targets are transgenic male-fertile (MF) and the non-transgenic male-sterile (MS) line of oilseed rape measurements across the study area. More details on the dataset are given in (Demšar et al, 2005).

5.3 PCTs for Hierarchical Multi-Label Classification

Hierarchical classification differs from the traditional classification in that the classes are organized in a hierarchy. This means that there can exist possible connections/relations between the classes. This task is then called hierarchical multi-label classification (HMC) (Silla and Freitas, 2011; Vens et al, 2008).

Hierarchical multi-label classification (HMC) extends the traditional classification in several directions. First, the classes are organized in a hierarchy. The similarity of class labels at higher levels of the hierarchy is more important than the similarity of class labels at lower levels. Second, a *hierarchy constraint* implying that an example that belongs to a given class automatically belongs to all its superclasses. This means that as a class is predicted, all of its parent classes are predicted as well. Third, an example can belong simultaneously to multiple classes that can follow multiple paths from the root class.

The predictive clustering framework offers a unifying approach for dealing with different types of structured outputs and the algorithms developed in this framework construct the predictive models very efficiently. Thus, it can handle the HMC tasks very efficiently. Moreover, PCTs can be easily interpreted by a domain expert, thus supporting the process of knowledge extraction.

We formally define the task of hierarchical multi-label classification as follows:
Given:

- A description space X that consists of tuples of values of primitive data types (Boolean, discrete or continuous), i.e., $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, ..., x_{i_D})$, where D is the size of the tuple (or number of descriptive variables),
- a target space S, defined with a class hierarchy $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order (structured as a rooted tree) representing the superclass relationship ($\forall\, c_1, c_2 \in C : c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2),
- a set E, where each example is a pair of a tuple and a set from the descriptive and target space respectively, and each set satisfies the hierarchy constraint, i.e., $E = \{(X_i, S_i) \mid X_i \in X, S_i \subseteq C, c \in S_i \Rightarrow \forall\, c' \leq_h c : c' \in S_i, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = \mid E \mid$), and
- a quality criterion q, which rewards models with high predictive accuracy and low complexity.

Find: a function $f : D \times X \longmapsto 2^C$ (where 2^C is the power set of C) such that f maximizes q and $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$, i.e., predictions made by the model satisfy the *hierarchy constraint*.

Silla and Freitas (2011) describe the algorithms for hierarchical classification as 4-tuple $\langle \Delta, \Sigma, \Omega, \Theta \rangle$. In this 4-tuple, Δ indicates whether the algorithm makes prediction for a single or multiple paths in the hierarchy, Σ is the depth of the predicted classes, Ω is the taxonomy structure of the classes that the algorithm can handle and Θ is the type of the algorithm (local or global). Using this categorization, the algorithm we present next can be described as follows:

- $\Delta =$ multiple path prediction: the algorithm can assign multiple paths or predicted classes to each instance.
- $\Sigma =$ non-mandatory leaf-node prediction: an instance can be labeled with a label at any level of the taxonomy.
- $\Omega =$ tree or directed acyclic graph: the algorithm can handle both tree-shaped or DAG hierarchies of classes.

- $\Theta =$ global classifier: the algorithm constructs a single model valid for all classes.

To apply PCTs to the task of Hierarchical Multi-label Classification (HMC), the variance and prototype are defined as follows (Vens et al, 2008).

First, the set of labels of each example is represented as a vector with binary components; the i'th component of the vector is 1 if the example belongs to class c_i and 0 otherwise. It is easily checked that the arithmetic mean of a set of such vectors contains as i'th component the proportion of examples of the set belonging to class c_i.

The variance of a set of examples E is defined as the average squared distance between each example's class vector (S_k) and the set's mean class vector ($\overline{S}$), i.e.,

$$Var(E) = \frac{\sum_k d(S_k, \overline{S})^2}{\mid E \mid}.$$

In the HMC context, the similarity at higher levels of the hierarchy is more important than the similarity at lower levels. To that aim, the distance measure used in the above formula is a weighted Euclidean distance:

$$d(S_1, S_2) = \sqrt{\sum_i w(c_i) \cdot (S_{1,i} - S_{2,i})^2},$$

where $S_{k,i}$ is the i'th component of the class vector S_k of an instance X_k, and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. More precisely, $w(c) = w_0 \cdot \text{avg}_\text{j} \{w(p_j(c))\}$, where $p_j(c)$ denotes the j'th parent of class c and $0 < w_0 < 1$).

For example, consider the toy class hierarchy shown in Figure 5.3(a,b), and two data examples: (X_1, S_1) and (X_2, S_2) that belong to the classes $S_1 = \{c_1, c_2, c_{2.2}\}$ (boldface in Figure 5.3(b)) and $S_2 = \{c_2\}$, respectively. Using a vector representation with consecutive components representing membership of class c_1, c_2, $c_{2.1}$, $c_{2.2}$ and c_3, in that order (pre-order traversal of the tree), the distance is calculated as follows:

$$d(S_1, S_2) = d([1,1,0,1,0],[0,1,0,0,0]) = \sqrt{w_0 + w_0^2}.$$

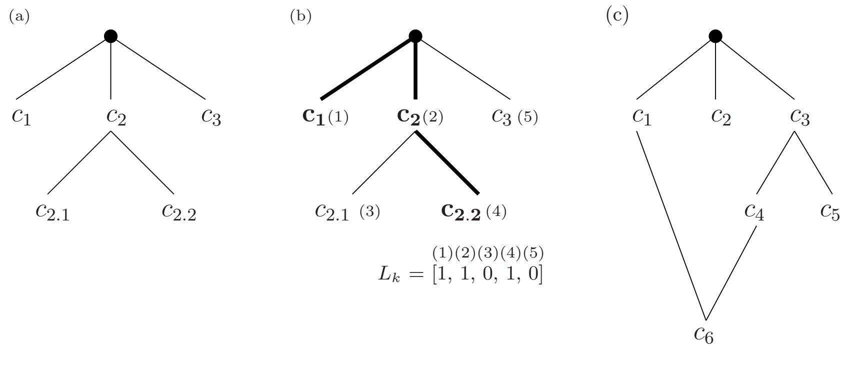


Figure 5.3: *A simple of tree and DAG hierarchy.* Toy examples of hierarchies structured as tree and DAG. (a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2. (b) The set of classes $\{c_1, c_2, c_{2.2}\}$, indicated in bold in the hierarchy, and represented as a vector. (c) A class hierarchy structured as a DAG. The class c_6 has two parents: c_1 and c_4. Figure taken from (Vens et al, 2008).

Note that this definition of $w(c)$ allows the classes to be structured in a directed acyclic graph (DAG). Figure 5.3(c) depicts an example of DAG structured hierarchy. In general, a DAG hierarchy can have two interpretations: if an example belongs to a given class c, then it also belongs to all super-classes of

c, or it belongs to at least one superclass of c. Here, we focus on the first case: the multiple inheritance interpretation.

The variance function used for tree-shaped hierarchies uses the weighted Euclidean distance between the class vectors, where the weight of a class depends on its depth in the hierarchy. When the hierarchy is a DAG, then the depth of a class is not unique: classes do not have single path from the top-node (for example see class c_6 in Fig. 5.3(c)). To resolve this issue, (Vens et al, 2008) suggest four aggregation schemes of the possible paths from the top-node to a given class: average, maximum, minimum and sum. The aggregation schemes use the observation that $w(c) = w_0^{depth(c)}$ can be rewritten as the recursive relation $w(c) = w_0 \cdot w(par(c))$, with $par(c)$ as the parent class of c, and the weights of the top-level classes equal to w_0. After an extensive experimental evaluation, (Vens et al, 2008) recommend to use the average as aggregation function ($w(c) = w_0 \cdot \mathrm{avg_j}\{w(par_j(c))\}$).

A classification tree stores in a leaf the majority class, which will be the tree's prediction for all examples that will arrive in the leaf. In the case of HMC, an example may have multiple classes, thus the notion of 'majority class' does not apply in a straightforward manner. Instead, the mean $\bar{S}$ of the class vectors of the examples in the leaf is stored as prediction. Note that the value for the i-th component of $\bar{S}$ can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i.

A. METABOLISM
A.1 amino acid metabolism
A.2 nitrogen, sulfur, selenium met.
A.1.3 assimilation of ammonia
A.1.3.1 metabolism of glutamine
A.1.3.1.1 biosynthesis of glutamine
A.1.3.1.2 degradation of glutamine
...
B. ENERGY
B.1 glycolysis and gluconeogenesis
C. CELL CYCLE and DNA PROCESSING
D. TRANSCRIPTION
D.1 RNA synthesis
D.2 RNA processing
D.3 transcriptional control

Figure 5.4: *A part of the FUN hierarchy.* A part of the FUN hierarchy Mewes et al (1999).

The prediction for an example that arrives in the leaf can be obtained by applying a user defined threshold τ on the probability; if the i-th component of $\bar{S}$ is above τ then the examples belong to the class c_i. When a PCT is making a prediction it preserves the hierarchy constraint (the predictions comply to the parent child relationships from the hierarchy) by choosing the value for the threshold τ as follows: $\tau_i \leq \tau_j$ whenever $c_i \leq_h c_j$.

The threshold is selected depending on the context. The user may set the threshold such that the resulting classifier has high precision at the cost of lower recall or vice versa, to maximize F-score, to maximize the interpretability or plausibility of the resulting model etc. Instead of committing to a particular rule for choosing the threshold, in this work, we use a threshold-independent measure (precision-recall curves) to evaluate the performance of the models.

Finally, the algorithm evaluates all possible tests to be put in a node. If no acceptable test can be found, that is, if no test significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a representative case, or prototype, of the given instances. The resulting algorithm is called CLUS-HMC.

Figure 8.2(a) gives an example of FunCat annotations of the hierarchy of yeast (*Saccharomyces Cerevisiae*) protein classes according to the FunCat hierarchy (Mewes et al, 1999).

6 Learning PCTs for Spatially Autocorrelated Data

In this chapter, we propose an extension of the predictive clustering framework that deals with spatial autocorrelated data. The system that we propose is called **SCLUS**, for Spatial Predictive Clustering System. It explicitly considers spatial autocorrelation when learning predictive clustering models.

First, we motivate the research approach presented in this chapter. Then we describe the proposed algorithm in details and investigate different spatial autocorrelation measures and their use in the case of spatial autocorrelated data. Next, we stress the importance of the selection of the bandwidth parameter analyze its time complexity.

Finally, we compare and discuss the predictive performance of the SCLUS algorithm for single and multi-target classification and regression tasks. The empirical evaluation is performed on a set of real-life spatial datasets. We discuss the evaluation results in terms of their accuracy, as well as in terms of the properties of the predictive models by analyzing the model sizes, the autocorrelation of the errors of the predictive models and their learning times.

6.1 Motivation

In traditional PCTs, the clustering phase is performed by maximizing variance reduction. This heuristic guarantees, in principle, accurate models since it reduces the error on the training set. However, it neglects the possible presence of spatial autocorrelation in the training data. To address this issue, we propose a different clustering phase which uses distances appropriately defined for the spatial domains in order to exploit the spatial structure of the data in the PCT induction phase and obtain predictive models that naturally deal with the phenomenon of spatial autocorrelation.

Although there is no theoretical proof that the consideration of autocorrelation may increase the accuracy of the learned models, our intuition is that it should still improve predictions that are consistently clustered across the space. This is due to the fact that clusters try to preserve the spatial arrangement of the data. In spite of the fact that the learned models are not obtained by purely minimizing the error on the training set, it is possible that considering autocorrelation in the training data makes the learned predictive model able to contain better knowledge of the underlying data distribution. This knowledge may yield better performance in the testing set than that obtained by a predictive model induced by maximizing only the variance reduction on the training data.

We build spatially-aware PCTs that use the spatial information as background knowledge and can be used to obtain spatially coherent predictions. The main assumption is that if there is a high autocorrelation between the examples in the dataset, not only the examples have similar target values but also they are in the same spatial neighborhood.

The contributions of this chapter are in:

- The development of an approach that uses these spatial autocorrelation measures in the induction of PCTs for the classification and regression tasks, by taking into account variations in the global/local data distribution across the spatial structure;

- An extensive evaluation of the effectiveness of the proposed approach on classification and regression problems in real-life spatial data;

- New empirical evidence is provided on the importance of considering spatial autocorrelation in predictive tasks and in particular on the ability of the proposed method to capture autocorrelation within the learned PCTs by analyzing the autocorrelation of the errors on an extensive set of different data.

- The scope of the performance evaluation is broadened to include clustering evaluation in terms of spatial dispersion of extracted clusters.

The algorithm proposed in this chapter extends the predictive clustering framework implemented in the CLUS system (Blockeel et al, 1998). It allows CLUS to handle spatial data in the setting outlined in Section 2.3.1.

6.2 Learning PCTs by taking Spatial Autocorrelation into Account

In this section, we provide a detailed description of the implemented algorithm for learning PCTs by taking spatial autocorrelation into account. Besides the algorithm itself, we also explore the properties of the spatial autocorrelation in SCLUS, emphasize the importance of this phenomenon. In addition, we stress the importance of the selection of the bandwidth parameter, as an important part of the SCLUS algorithm. At last, we give an analysis of the time complexity of the proposed algorithm.

6.2.1 The Algorithm

We can now proceed to describe the top-down induction algorithm for building Spatial PCTs (Algorithm 2). It is a recursive algorithm which takes as input a set of spatial training examples E, the context space D (that can be expressed by the spatial coordinates or the standard Euclidian distance between two examples in space) and the function $\eta : V \mapsto \mathbf{X} \times \mathbf{Y}$ and partitions the descriptive space until a stopping criterion is satisfied (Algorithm 2, line 2). Since the implementation of this algorithm is based on the implementation of the CLUS algorithm, we will call this algorithm SCLUS (for Spatial CLUS).

Algorithm 2 *Top-down induction of SCLUS.*

```
1: procedure SpatialPCT($E$) returns tree
2: if stop(E) then
3:     return leaf(Prototype($E$))
4: else
5:     $(v^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6:     for each Boolean test $v$ do
7:         $\mathcal{P}$ = partition induced by $v$ on $E$
8:         $h = \alpha \cdot \widehat{\Delta_Y}(E, \mathcal{P}) + (1-\alpha) \cdot S_Y(E, \mathcal{P})$
9:         if $(h > h^*)$ then
10:            $(v^*, h^*, \mathcal{P}^*) = (v, h, \mathcal{P})$
11:        end if
12:    end for
13:    for each $E_k \in \mathcal{P}^*$ do
14:        $tree_k$ = SpatialPCT($E_k$)
15:    end for
16:    return node($v^*$, $\bigcup_k \{tree_k\}$)
17: end if
```

The main loop (Algorithm 2, lines 6-10) searches for the best attribute-value test v^* that can be associated to a node t. It associates the best test v^* with the internal node t and calls itself recursively to construct a subtree for each subset (cluster) in the partition P^* induced by v^* on the training example set.

As discussed above concerning *i)*, splits are derived only from the space $\mathbf{X}$ of the predictor variables. Possible tests are of the form $X \leq \beta$ for continuous variables, and $X \in \{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ (where $\{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ is a subset of the domain D_X of X) for discrete variables. For continuous variables, possible values of β are found by sorting the distinct values of X in the training set associated with t, then considering a threshold between each pair of adjacent values. Therefore, if the cases in t have d distinct values for X, at most $d-1$ thresholds are considered. When selecting a subset of values for a discrete variable, we rely on a non-optimal greedy strategy (Mehta et al, 1996). It starts with an empty set $Left_t = \oslash$ and a full set $Right_t = D_X$, where D_X is the domain of X. It moves one element from $Right_t$ to $Left_t$, such that the move results in an increased heuristic h. It is noteworthy that this greedy strategy to explore the space of candidate splits on a discrete variable does not require the definition of an apriori ordering on the possible values of D_X as used in traditional decision tree induction (Breiman et al, 1984).

The algorithm evaluates the best split according to the formula (6.1) reported in Algorithm 2, line 8.

$$h = \alpha \cdot \widehat{\Delta_Y}(E, \mathcal{P}) + (1-\alpha) \cdot S_Y(E, \mathcal{P}) \tag{6.1}$$

This formula is a linear combination of the scaled variance reduction $\widehat{\Delta_Y}(E, \mathcal{P})$ (see Equation 7.3) and autocorrelation measure $S_Y(E, \mathcal{P})$ (see Equation 6.2). Both, variance and autocorrelation are computed for the Y variable (the class). In the case of multiple target variables, the average values of both, variance reduction $\Delta_Y(E, \mathcal{P})$ and autocorrelation $S_Y(E, \mathcal{P})$ are taken over the set of target variables, where each target variable contributes equally to the overall h value.

The influence of these two parts of the linear combination when building the PCTs is determined by a user-defined coefficient α that falls in the interval [0, 1]. When $\alpha = 0$, SCLUS uses only spatial autocorrelation, when $\alpha = 0.5$ it weights equally variance reduction and spatial autocorrelation, while when $\alpha = 1$ it works as the original CLUS algorithm. If autocorrelation is present, examples with high spatial autocorrelation (close to each other in space) will fall in the same cluster and will have similar values of the response variable. In this way, we are able to keep together spatially close examples without forcing spatial splits (which can result in losing generality of the induced models).

In spatial analysis, several spatial autocorrelation statistics have been defined. The most common ones: Global Moran's I and Global Geary's C (Legendre, 1993). They are used in the case of continuous target variable.

In the context of this dissertation, we redefined these measures also for the case of discrete target variable in order to be able to use them for classification tasks. The measures of spatial autocorrelation are presented in Section 3.2. In the context of our algorithm we use Global Moran's I and Global Geary's C given by the Equation 3.4, Equation 3.7 and Equation 3.20, Equation 3.21, for the classification and for the regression task, respectively.

While both statistics reflect the spatial dependence of values, they do not provide identical information: C emphasizes the differences in values between pairs of observations, while I emphasizes the covariance between the pairs. This means that Moran's I is smoother, whereas Geary's C is more sensitive to differences in small neighborhoods.

The weights w_{ij} used in equations (Equation 3.20, Equation 3.21, Equation 3.4 and Equation 3.7) are defined as a function of the spatial distance measure. The basic idea is that the examples close to a specific example have more influence in the estimation of its response value than examples farther away.

Whatever weighting schema is employed, the choice of the bandwidth b plays a crucial role. This means that the main problem is how to select the optimal bandwidth b. This problem is described and tackled in Section 6.2.3.

Namely, $S_Y(E,\mathcal{P})$ can be defined in terms of either the Moran's I or the Geary's C. However, since I_Y and C_Y range in different intervals (even though are consistently monotonic), it is necessary to appropriately scale them. Since variance reduction is non-negative, we decided to scale both in the interval [0, 1], where 1 means high positive autocorrelation and 0 means high negative autocorrelation.

For Moran's I, $S_Y(E,\mathcal{P})$ is:

$$S_Y(E,\mathcal{P}) = \frac{1}{|E|} \cdot \sum_{E_k \in \mathcal{P}} |E_k| \cdot \widehat{I_Y}(E_k) \tag{6.2}$$

where $\widehat{I_Y}(E_k)$ is the scaled Moran's I computed on E_k, i.e.,

$$\widehat{I_Y}(E_k) = \frac{I_Y(E_k)+1}{2} \tag{6.3}$$

This scales Moran's I from the interval [-1, 1] to [0, 1].

For Geary's C , $S_Y(E,\mathcal{P})$ is:

$$S_Y(E,\mathcal{P}) = \frac{1}{|E|} \cdot \sum_{E_k \in \mathcal{P}} |E_k| \cdot \widehat{C_Y}(E_k) \tag{6.4}$$

where $\widehat{C_Y}(E_k)$ is the scaled Geary's C computed on E_k, i.e.,

$$\widehat{C_Y}(E_k) = \frac{2-C_Y(E_k)}{2} \tag{6.5}$$

This scales Geary's C from the interval [0, 2] to [0, 1].

The choice of the scaling interval does not affect the heuristic computation, therefore other scaling intervals are possible as well, provided that, in all cases, the same scaling is performed and the monotonicity of the scaled measure is maintained.

Moreover, in order to guarantee a fair combination of the variance reduction and the autocorrelation statistic $S_Y(\mathcal{P},E)$, we also need to scale the variance reduction to the interval [0, 1]. For that purpose, we use a common scaling function:

$$\widehat{\Delta_Y}(E,\mathcal{P}) = \frac{\Delta_Y(E,\mathcal{P}) - \Delta min}{\Delta max - \Delta min} \tag{6.6}$$

where Δmax and Δmin are the maximum and the minimum values of $\Delta_Y(E,\mathcal{P})$ over the possible splits.

The search stops when the number of examples falling in a leaf is smaller than $\sqrt{N}$, which is considered as a good locality threshold that does not permit to lose too much in accuracy also for rule based classifiers (Gora and Wojna, 2002). A further stopping criterion is based on the exact Fisher test (F-test) that is performed to check whether a given split/test in an internal node of the tree results in a reduction in h that is statistically significant at a given significance level. In order to estimate the optimal significance level among the values in the set $\{1, 0.125, 0.1, 0.05, 0.01, 0.005, 0.001\}$, we optimize the MSE obtained with an internal 3-fold cross validation. When one of the stopping criterion is satisfied, the algorithm creates a leaf and labels it with a predictive function (in the regression case, the average; and in the classification case, the mode of the class values) defined for the examples falling in that leaf (see lines 2-3 of Algorithm 2).

In SCLUS, the pruning is the pessimistic error pruning strategy which is also implemented in several regression/model tree learners (including M5' and CLUS). This means that a subtree is added only if the error committed at the leaf is greater than the errors commented by the subtree multiplied by a scaling factor (Wang and Witten, 1997). The results that we present in this chapter are those of the pruned tree models learned by SCLUS.

6.2.2 Exploiting the Properties of Autocorrelation in SCLUS

The consideration of spatial autocorrelation in clustering has been already investigated in the literature (Glotsos et al, 2004; Jahani and Bagherpour, 2011). Motivated by the demonstrated benefits of considering autocorrelation, in this chapter, we exploit some characteristics of autocorrelated data to improve the quality of PCTs.

The use of spatial autocorrelation in predictive clustering offers several advantages since it allows to:

- determine the strength of the spatial effects on the variables in the model;
- consider tests on assumptions of stationarity and heterogeneity in the spatial data;
- identify the possible role of the spatial interaction/distance decay on the predictions associated to each of the nodes of the tree;
- focus on the "spatial neighborhood" to better understand the effects that it can have on other neighborhoods and vice versa.

Moreover, as recognized by (Griffith, 2003), autocorrelation implicitly defines a zoning of a (spatial) phenomenon: Taking this into account reduces the effect of autocorrelation on prediction errors. Therefore, we propose to perform clustering by maximizing both variance reduction and cluster homogeneity (in terms of autocorrelation), at the same time when considering the different attempts for adding a new node to the predictive clustering.

In PCTs induced by considering spatial autocorrelation, different effects of autocorrelation can be identified and considered at each node of the tree. This non-stationary view of autocorrelation is global at the root node and local at the leaves. At the same time, the tree structure allows us to deal with the so-called "ecological fallacy" problem (Robinson, 1950), according to which individual sub-regions do not have the same data distribution as the entire region.

With regard to the statistical properties of the measures of spatial autocorrelation, most of the theoretical research in Statistics and Econometrics exploits the so called "autoregressive model" in order to measure spatial autocorrelation in lattices (Griffith, 2003). More formally, the autoregressive model is defined as:

$$\hat{e}_i = \rho \sum_j w_{ij}\, e_j + \varepsilon_i \tag{6.7}$$

where $e_j = Y_j - \overline{Y}$ is the prediction error (where prediction is based on the average), ρ is a parameter that expresses the spatial dependence, w_{ij} are the elements of the neighborhood matrix W and the error ε_i follows a Gaussian (normal) distribution (Engle, 1982).

In this case, the informal notion of spatial dependence is often implicitly based on an autoregressive framework, where the goal is to assess the predictive ability of the neighboring values of the data. As recognized by (Li et al, 2007), in order to informally assess the strength of the spatial dependence, exploratory data analysis should be based on estimating ρ in the autoregressive model (see Equation 6.7). This means that the parameter ρ plays a crucial role in representing autocorrelation in the data.

One common solution for estimating ρ is to use a modified least squares estimator, which is the solution to the following quadratic equation in ρ:

$$\mathbf{e}^T(\mathbf{I} - \rho\mathbf{W})^T\mathbf{W}(\mathbf{I} - \rho\mathbf{W})\mathbf{e} = 0 \tag{6.8}$$

where $\mathbf{W}$ is the matrix representation of w_{ij}, $\mathbf{I}$ is the identity matrix and $\mathbf{e}$ is the vector of e_i values.

Although this estimator is consistent (Li et al, 2007), its computation is not straightforward. Particularly, it would require the computation of the Maximum Likelihood Estimator of ρ which is impractical when large datasets need to be processed (Li et al, 2007). Therefore, instead of computing an estimate of ρ, Moran's I is commonly used in spatial data mining applications. Indeed, as proved by (Jin, 2010), under the assumption that $w_{ij} = w_{ji}$ (**W** is symmetric), Moran's I is monotonic in ρ. The same applies for Geary's C. Unfortunately, this result is not valid when $w_{ij} \neq w_{ji}$ (some counterexamples can be found). Moreover, (Li et al, 2007) empirically proved that Moran's I is a good (not unbiased) estimator of ρ in the case when ρ approaches zero.

This means that Moran's I and Geary's C are a good indicators of the spatial dependence under some conditions. This analysis also confirms that a model that is able to take autocorrelation into account should lead to small values of ρ, that is, according to Equation 6.7, it should lead to a reduction of the effect of an error in the neighborhood.

6.2.3 Choosing the Bandwidth

The choice of the bandwidth (denoted by b in Equation 2.5) is perhaps the most critical decision to be taken in the modeling process. This parameter controls the degree of smoothing. A small bandwidth results in a very rapid distance decay, whereas a larger value results in a smoother weighting scheme. At the same time, this parameter influences the calculation of autocorrelation.

The bandwidth may be defined manually or by using some adaptive method on the whole training space, such as cross validation and the corrected Akaike Information Criterion (AIC) used in GWR (Fotheringham et al, 2002). A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the complexity of the algorithm. In this study, for the selection of the bandwidth, we minimize the leave-one-out cross validated Root Mean Square Error (RMSE). Moreover, in this automatic determination of the bandwidth, the selection is not performed directly on the bandwidth b, but on $b^{\%}$, that is, the bandwidth expressed as a percentage of the maximum distance between two examples. This means that the algorithm implicitly considers different bandwidth values b at different nodes of the tree depending on the maximum distance between connected examples falling in that node of the tree. The bandwidth $b^{\%}$ ranges in the interval $[0, 100\%]$.

Minimization is performed by means of the Golden section search (Brent, 1973) that recursively partitions the $b\%$ domain. Golden section search is similar to binary search, improving it by splitting the range in two intervals with a length ratio of γ instead of 1 (equal parts). *Golden ratio* has the value $\gamma = \frac{1+\sqrt{5}}{2}$.

The share maintains a pair of minimum and maximum bandwidth values, $b_1^{\%}$ and $b_2^{\%}$ (at the first iteration, they are initialized as the minimum and maximum bandwidth in the interval $[0, 100\%]$). At each step, the algorithm identifies a point $b_3^{\%}$ between them, according to the golden ratio and computes the cross-validated error for that point ($error_{b_3^{\%}}$). The values of the function at these points are $f(b_1^{\%})$, $f(b_3^{\%})$, and $f(b_2^{\%})$ and, collectively, these are known as a "triplet". The algorithm than identifies the only parabola with a vertical axis that intersects the points $\{(b_1^{\%}, error_{b_1^{\%}}), (b_3^{\%}, error_{b_3^{\%}}), (b_2^{\%}, error_{b_2^{\%}})\}$. On the basis of the position of the minimum of this parabola, the system decides whether to consider ($b_1^{\%}$, $b_3^{\%}$) or ($b_3^{\%}$, $b_2^{\%}$) as the next pair of (minimum and maximum) $b^{\%}$ values.

The search stops when there is no cross-validated error reduction. In the regression case, error is the RMSE computed by fitting a weighted linear model for the example left out. In the case of classification, similarly to the regression case, the error is the RMSE on the example left out, but it is computed as the average of RMSEs obtained by fitting a weighted linear model for each binary target variable obtained after the binarization of the discrete response variable. Weights are defined according to Equation 2.5.

6.2.4 Time Complexity

The computational complexity of the algorithm depends on the computational complexity of adding a splitting node t to the tree, which in fact depends on the complexity of selecting a splitting test for t. A splitting test can be either continuous or discrete. In the former case, a threshold β has to be selected for a continuous variable. Let N be the number of examples in the training set E; Then the number of distinct thresholds can be N-1 at worst. They can be determined after sorting the set of distinct values. If m is the number of predictor variables, the determination of all possible thresholds has a complexity $O(m * N * \log N)$ when an optimal algorithm is used for sorting.

For each variable, the system has to compute the evaluation measure h for all possible thresholds. This computation has, in principle, time-complexity $O((N-1)*(N+N^2))$; where $N-1$ is the number of thresholds, $O(N)$ is the complexity of the computation of the variance reduction $\Delta_Y(E,\mathcal{P})$ and $O(N^2)$ is the complexity of the computation of autocorrelation $S_Y(E,\mathcal{P})$. However, for each threshold, it is not necessary to recompute values from scratch since partial sums in both variance reduction computation [1] and in autocorrelation computation can be used. In particular, partial sums can be incrementally updated depending on the examples that are iteratively moved from the right to the left branch. This optimization makes the complexity of the evaluation of the splits for each variable $O(N^2)$. This means that the worst case complexity of creating a splitting node on a continuous attribute, in the case of a continuous target variable is $O(m*(N\log N+N^2))$ and in the case of a discrete target variable is $O(m*(N\log N+q*N^2))$, where q is the number of classes.

Similarly, for a discrete splitting test (for each variable), the worst case complexity, in the case of a continuous target variable, is $O((d-1)*(N+N^2))$, where d is the maximum number of distinct values of a discrete descriptive variable ($d \leq N$) and in the case of a discrete target variable is $O((d-1)*(N+q*N^2))$. This complexity takes the same optimization proposed for continuous splits into account.

Therefore, finding the best splitting node (either continuous or discrete), in the case of continuous target variable, has a complexity of $O(m*(N\log N+N^2))$+$O(m*(d-1)*(N+N^2))$, that is $O(m*N*(\log N+d*N))$. Analogously, finding the best splitting node (either continuous or discrete), in the case of discrete target variable, has a complexity of $O(m*N*(\log N+q*d*N))$.

The cost of passing all instances and calculating the needed statistics for all descriptive attributes over a single target variable is $O(d*N)$. Once the best test is selected, splitting the instances into the respective nodes costs $O(N)$. To sum up, the computational complexity of creating a node in the tree is $O(m*N*(\log N+q*d*N))+O(d*N)+O(N)$. Furthermore, if we take into consideration that $m=O(d)$, then the computational complexity of the process of creating a node in the tree is $O(d*N*(\log N+q*d*N))+O(d*N)+O(N)$.

We assume, as in (Witten and Frank, 2005), that the tree is balanced and bushy. This means that the number of nodes is in the order of $\log N$, i.e., $O(\log N)$. Having this in mind, the total computational cost of tree construction is $O(d*N*(log^2 N+q*d*N))+O(d*N*\log N)+O(N*\log N)$. The upper bound of this cost is determined by the first term of the overall computational complexity, i.e., $O(d*N*(log^2 N+q*d*N))$.

In the task of predicting multiple targets, a tree is constructed for each target separately, thus the computational complexity is T times higher than the computational complexity of a tree for a single target attribute, i.e., $O(d*N*(log^2 N+q*d*N))+O(d*N*\log N)+O(N*\log N)$, where N is the number of examples and q is the maximum number of distinct values of discrete variables d.

6.3 Empirical Evaluation

In this section, we present an empirical evaluation of the system SCLUS that implements the method SpatialPCTs presented in Section 6.2. After we provide a description of the used datasets and experimental settings, we proceed to presenting empirical results. The results target single and multi-target classification and regression tasks.

[1] both in terms of variance and Gini index

First, we investigate the predictive performance of the system along the dimensions of the weighting functions and autocorrelation measures used in the evaluation of splits, as well as the sensitivity of SCLUS to the choice of the bandwidth b and to the setting of α. Second, we evaluate the system for automatic determination of the bandwidth, presented in Section 6.2.3. Third, we compare SCLUS performance to the baseline CLUS and competitive (spatial) regression and classification methods on real-world datasets.

At the end, we present a more detailed analysis of the properties of the obtained predictive models. Specifically, we evaluate properties of the obtained clusters in the case when spatial autocorrelation is considered in the clustering phase. Empirical results show that our algorithm performs better than PCTs learned by completely disregarding spatial information and PCTs that are tailored for spatial data, but do not take into autocorrelation into account.

6.3.1 Datasets

In this experimental evaluation, we use real-world data that includes a spatial component. We consider nine datasets for the regression task and four datasets for the classification task. The regression datasets are FF, NWE, SIGMEA_MS and SIGMEA_MF, Foixa, GASD, River, Kenya and Malawi. The classification datasets are Foixa_01, Foixa_045, Prim and Kras.

The **Forest Fires** (FF) (Cortez and Morais, 2007) dataset is publicly available for research purposes from the UCI Machine Learning Repository [1]. It contains 517 forest fire observations from the Montesinho park in Portugal. The data, collected from January 2000 to December 2003, includes the coordinates of the forest fire sites, the burned area of the forest given in *ha* (response variable), the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), the Drought Code (DC), the Initial Spread Index (ISI), the temperature in degrees Celsius, the relative humidity, the wind speed in km/h and the outside rain in mm within the Montesinho park map.

The **NWE** (North-West England)[2] dataset contains census data concerning North West England. The data include the percentage of mortality (target variable) and measures of deprivation level in the ward, including index scores such as the Jarman Underprivileged Area Score, Townsend score, Carstairs score and the Department of the Environment Index. The spatial coordinates of the ward centroid are given as well. The spatial unit of analysis is a ward, i.e., a sub-area of a region.

The **SIGMEA_MS** and **SIGMEA_MF** (MS and MF) (Demšar et al, 2005) datasets are derived from one multi-target regression dataset containing measurements of pollen dispersal (crossover) rates from two lines of plants (target variables), that is, the transgenic male-fertile (MF) and the non-transgenic male-sterile (MS) line of oilseed rape. The predictor variables are the cardinal direction and distance of the sampling point from the center of the donor field, the visual angle between the sampling plot and the donor field, and the shortest distance between the plot and the nearest edge of the donor field, as well as the coordinates of the sampling point.

The **Foixa** dataset (Debeljak et al, 2012) contains measurements of the rates of outcrossing (target variable) at sampling points located within a conventional field that comes from the surrounding genetically modified (GM) fields within a 400 ha maize-oriented production area in the Foixa region in Spain. The independent variables include the number and size of the surrounding GM fields, the ratio of the size of the surrounding GM fields and the size of conventional field, the average distance between conventional and GM fields, as well as the coordinates of the sampling points.

The **GASD** dataset (USA Geographical Analysis Spatial Dataset) (Pace and Barry, 1997) contains observations on US county votes cast in the 1980 presidential election. Specifically, it contains the total number of votes cast per county (target variable), the population above 18 years of age in each county,

[1] `http://archive.ics.uci.edu/ml/`

[2] http://www.ais.fraunhofer.de/KD/SPIN/project.html

the number of owner-occupied housing units, the aggregate income and the coordinates of the centroid of the county.

The **River** (Macchia et al, 2011; Ohashi et al, 2010) dataset contains water information of the Portuguese rivers Douro and Paiva in 2009. This dataset includes measurements of the pH level, conductivity, turbidity and common bacteria like *Escheria Coli* and *Coliformi Bacteria* taken at control points along the rivers. In addition, the navigation distance between the control points is also available. The goal is to predict river pollution and the pH level is considered as the target variable, since it is recognized to be a good indicator of river pollution.

The **Kenya** and **Malawi** datasets contain observations of the Headcount poverty index (target variable) and other data found in the poverty mapping reports for the countries Kenya and Malawi [1]. Specifically, they contain the total number and density of poor people, the average number of years of schooling of adults, the number of active community groups, the accessibility of resources like water, roads and electricity, the average resource consumption, and the coordinates of the spatial unit (administrative level).

The **Foixa_01** and **Foixa_045** (Debeljak et al, 2012) classification datasets are derived from the **Foixa** dataset through a discretization proposed by a domain-expert. Discretization is performed on the target variable (outcrossing rate) and is based on the thresholds 0.1 % and 0.45 %. These thresholds correspond to the farmers' internal standard that keeps them on the safe side of maize production (0.45 %) and the requirements of the starch industry for the purity of maize at the entrance to the production process (0.1 %). In particular, in Foixa_01 dataset, the class *low* refers to fields with outcrossing rate in the interval [0, 0.1 %], whereas the class *high* refers to fields with outcrossing rate in the interval [0.1 %, 100 %]. Similarly, in Foixa_045 dataset, the class *low* refers to fields with outcrossing rate in the interval [0, 0.45 %], whereas the class *high* refers to fields with outcrossing rate in the interval [0.45 %, 100 %].

The **Kras** and **Prim** (Stojanova et al, 2012) classification datasets contain data on fire outbreaks (target variable) in the Kras and Primorska region in Slovenia. The data consists of GIS data (spatial coordinates, altitude, forest coverage, percentage of agricultural areas, percentage of urban areas, distance from roads, highways, railways, cities, etc.), multi-temporal MODIS data (average temperature and average net primary production), weather forecast data from ALADIN (temperature, humidity, solar energy, evaporation, speed and direction of the wind, transpiration, etc.). For the Kras dataset, we additionally have vegetation height and vegetation structure data obtained from LIDAR and LANDSAT images to which a predictive models learned from LIDAR data and LANDSAT images was applied. Together with the **Foixa_01** and **Foixa_045** datasets described above, we consider four classification datasets.

A description of the datasets in terms of general properties is provided in Table 6.1, where we also report the automatically chosen bandwidth as described in Section 6.2.3. Spatial autocorrelation of the MF dataset is illustrated in Figure 6.4 (a).

6.3.2 Experimental Setup

The experiments were performed on an Intel Xeon CPU @2.00GHz server running the Linux Operating System. For each dataset, we evaluate the effectiveness of SCLUS in terms of accuracy, model complexity, learning time as well as quality of extracted clusters. All of these performance measures are estimated by using 10-fold cross validation (always on the same folds). In particular, the accuracy is measured in terms of the Root Mean Squared Error (RMSE) for regression tasks and Precision and Recall for classification tasks, while the model complexity is measured in terms of the number of leaves in the learned trees. The computation time is measured in seconds. The quality of the clusters is measured in terms of their *spatial dispersion*. More formally, let $C = \{C_1, \ldots, C_s\}$ be the set of clusters associated

[1] `http://sedac.ciesin.columbia.edu/povmap/ds_info.jsp`

Table 6.1: *Descriptions of the spatial datasets used in the empirical evaluation.* For each dataset, we give the Task (classification or regression), N – number of examples, *#Attr.* – number of descriptive attributes and b – automatically chosen bandwidth values, given in (%).

Dataset	Task	N	*#Attr.*	auto.chosen $b^{\%}$
FF	Regression	517	12	100%
NWE	Regression	970	4	7.7 %
MS	Regression	817	4	4.8 %
MF	Regression	817	4	9.1 %
Foixa	Regression	420	7	64.6 %
GASD	Regression	3106	4	2.5 %
River	Regression	32	8	100 %
Kenya	Regression	121	9	63.8 %
Malawi	Regression	3004	10	8.1 %
Kras	Classification	1439	158	100 %
Prim	Classification	2024	104	100 %
Foixa_01	Classification	420	7	100 %
Foixa_045	Classification	420	7	100 %

with the leaves of a PCT, similar to what is suggested in (Sampson and Guttorp, 1992), we compute the *spatial dispersion* as the average intra-cluster distance, that is:

$$SD = avg_{\ C_k \in C} \left(\sum_{o_i, o_j \in C_k} \frac{euclideanDist(o_i, o_j)}{(N_k^2)} \right) \tag{6.9}$$

where N_k is the number of examples in C_k and $euclideanDist(o_i, o_j)$ is the spatial distance between the objects o_i and o_j. According to our assumptions, the smaller the value of SD, the better the clustering.

We first evaluate the performance of SCLUS using different bandwidth values (b=1 %, 5 %, 10 %, 20 %) and weighting functions (Euclidean (Euc.), Modified Euclidean (Mod.) and Gaussian (Gauss.)) in order to understand their impact on the accuracy of the models. Then, SCLUS is run with automatic bandwidth determination, with the two different spatial autocorrelation measures, Global Moran's I (SCLUS_Moran) and Global Geary's C (SCLUS_Geary), and with $\alpha \in \{0, 0.5\}$. These different configurations of SCLUS are considered as a part of its definition and depend on the nature of the modeling problems considered. SCLUS is compared with the original CLUS (Blockeel et al, 1998) algorithm. We also compare SCLUS to a modification of CLUS, where the response variable set is extended with the spatial coordinates as additional response variables. This is done only for the computation of the split evaluation measure. In this way, we are able to implicitly take autocorrelation into account. Henceforth, we refer to this configuration of CLUS as CLUS*. For the regression task, SCLUS is compared to other competitive regression algorithms, i.e., M5' Regression Trees (RT) and Support Vector Regression (SVR) (implemented in the WEKA software suite (Witten and Frank, 2005)) and Geographically Weighted Regression (GWR) (Fotheringham et al, 2002). Only GWR, SCLUS and CLUS* take autocorrelation into account.

6.3.3 Results and Discussion

We present an empirical evaluation of the system SCLUS that implements the method SpatialPCTs presented in Section 6.2. First, we investigate the performance of the system along the dimensions of the weighting functions and autocorrelation measures used in the evaluation of splits, as well as the sensitivity of SCLUS to the choice of the bandwidth b and to the setting of α. Second, we evaluate the system for automatic determination of the bandwidth, presented in Section 6.2.3. Third, we compare the performance of SCLUS to the baseline performance of CLUS, as well as to the performance of the competitive (spatial) regression and classification methods on real world datasets. Finally, we give a qualitative analysis of the results on several ecological datasets (MS, MF and FOIXA), both in terms of the structure of the learned models and in terms of the visual differences in their predictions.

Regression Tasks

In the following subsection, we compare and discuss the predictive performance of the SCLUS algorithm for single and multi-target regression tasks.

Table 6.2 shows the effect of the bandwidth and of the weighting function as used within the autocorrelation part of the splitting criterion. The bandwidth (b=1 %, 5 %, 10 %, 20 %) is given as a percentage of the maximum spatial distance between any two examples in space. Specifically, in Table 6.2, we report the RMSE results of the SCLUS_Moran models learned with different weighting functions (Euclidean (Euc.), Modified Euclidean (Mod.) and Gaussian (Gauss.)), evaluation measures (SCLUS_Moran and SCLUS_Geary), as estimated by 10-fold CV. For comparison, we consider only spatial autocorrelation and ignore the variance reduction in the splitting criterion ($\alpha = 0$).

While the selection of the bandwidth very much influences the results, there is no larger difference in performance when using different weighting functions (see Equation 2.5, Equation 2.6 and Equation 2.7). Experiments show that manual (non-automatic) tuning of the bandwidth is difficult since there is no unique bandwidth value which shows the best predictive capabilities for each dataset/weighting schema. On the other hand, the mechanism we propose for the automatic choice of bandwidth seems to be effective since it generally leads to higher accuracy.

In Table 6.3, we report the RMSE results, as estimated by 10-fold CV, for SCLUS (with Global Moran's I and Global Geary's C as measures of spatial autocorrelation, with an automatically chosen bandwidth and with $\alpha \in \{0, 0.5\}$). We also report the results for CLUS*, GWR, SVR and M5' Regression Trees. The implementation of CLUS* supports only datasets that contain coordinates and not relative distances between the examples in the dataset which is the case with the River dataset. Note that SCLUS does not have such implementation problems, whereas CLUS does not use the spatial information at all. M5' Regression Trees and SVR do not consider spatial autocorrelation while GWR incorporates it and accounts for non-stationarity.

The results show that the difference in the performance of SCLUS with Global Moran's I and Global Geary's C as measures of spatial autocorrelation is not great. This is not surprising, due to the fact that both measures evaluate the strength of spatial autocorrelation by emphasizing the covariance (differences in values) between pairs of observations. On the other hand, the selection of the user-defined parameter α is a very important step with a strong external influence on the learning process since it is not dependent on the properties of the data, as in the case of the bandwidth and the measures of spatial autocorrelation. The simplest solution is to set this parameter to 0 (consider only the spatial statistics) or 1 (consider only the variance reduction for regression, as in the original CLUS algorithm). Any other solution will combine the effects, allowing both criteria to influence the split selection.

In addition, the results in Table 6.3 show that, in most cases, these is at least one configuration of SCLUS that outperforms CLUS in terms of RMSE error (except for the GASD dataset) and that there is great difference among the results for some of the datasets in favor of SCLUS. Moreover, the two modifications of CLUS, SCLUS and CLUS*, outperform by a great margin the standard regression method GWR that incorporates spatial autocorrelation and accounts for non-stationarity. Furthermore, SCLUS compares very well to standard regression tree-based methods like M5' Regression Trees, as well as non tree-based methods as SVR that do not consider autocorrelation.

Next, focusing on the comparison between SCLUS and CLUS*, we have to emphasize the fact that both SCLUS and CLUS* are modifications of CLUS that are designed to improve (if possible) both the spatial homogeneity and the accuracy of the CLUS models by modifying/enhancing the heuristic (variance reduction) used to evaluate each split in the process of tree construction. Whereas SCLUS accounts for autocorrelation that is often present in the data, CLUS* accounts for the spatial coordinates (usually presented in pairs (x, y) or (latitude, longitude)) in the case of the data obtained from spatial datasets by considering them as response variables in addition to the actual ones. This means that CLUS* aims at generating PCTs that will maximize the inter-cluster variance reduction of both the target and the coordinates[1]. Moreover, much higher importance is given to the spatial information in CLUS* than to the actual response, as they all are normalized at the beginning of the modeling process and equal importance is given to them: with a single tree target and two coordinates x and y as additional targets. This solution makes the predictions of the CLUS* models more coherent in space than those of the CLUS models and (if possible) increases the accuracy of the models.

[1]This is possible as CLUS is designed to deal with multi-target prediction problems.

Table 6.2: *The RMSEs of the SCLUS_Moran models.* The different models are learned with different weighting functions, evaluation measures, bandwidth values and α=0. The models are estimated by 10-fold CV. The best results for each bandwidth value are given in bold.

Dataset	1 %			5 %			10 %			20 %			auto. chosen b (%)		
	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.
FF	47.22	47.22	47.22	47.22	47.22	47.22	47.22	47.22	47.22	47.22	47.22	47.22	**42.82**	47.21	47.21
NWE	2.48	2.48	2.48	2.48	2.48	2.48	2.46	2.46	2.46	2.45	2.45	2.45	**2.16**	2.36	2.39
Foixa	2.58	**2.57**	2.58	2.58	2.64	**2.55**	2.58	**2.55**	2.56	2.57	2.59	**2.54**	2.53	2.55	**2.41**
GASD	**0.17**	**0.17**	0.18	**0.17**	**0.17**	0.19	**0.17**	0.19	0.19	0.19	0.19	0.19	0.18	0.18	**0.17**
MF	2.46	**2.19**	2.47	2.45	**2.19**	2.47	2.34	**2.19**	2.47	2.44	**2.19**	2.47	2.47	**2.04**	2.47
MS	6.00	**4.77**	5.74	5.97	**3.92**	5.97	5.81	**4.02**	5.81	5.81	**4.02**	5.81	**5.44**	5.66	5.59
River	0.33	**0.28**	**0.28**	0.33	**0.28**	**0.28**	0.33	**0.28**	**0.28**	0.33	**0.28**	**0.28**	**0.28**	0.33	0.33
Kenya	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	**0.15**	0.17
Malawi	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	**0.10**	0.12	**0.10**	0.69	0.12	0.11	**0.10**

However, CLUS* models cannot deal with non-stationary autocorrelation (i.e., when autocorrelation varies significantly throughout the space), while SCLUS models deal with non-stationarity appropriately. In SCLUS, two different geographical regions that have the same distribution of the independent variables and target values which exhibit autocorrelation, can be covered by one leaf of the tree: In CLUS*, the data will need to be split into different regions due to the spatial homogeneity. Moreover, CLUS* cannot handle different definitions of the regression problem that can arise from different definitions of the space, e.g., using different similarity/proximity measures.

To summarize the results presented in Table 6.3, we count the number of wins/losses of the twelve configurations of SCLUS when compared to all other methods.

In Table 6.4, we report an overall number representing the number of wins minus the number of losses, when each of the twelve configurations of SCLUS (see Table 6.3) are compared to CLUS*, GWR, SVR and M5' Regression Trees. A positive number means that SCLUS has more wins than losses than the other method, whereas a negative number means the opposite. The results are in favor of SCLUS in most cases. The best results are obtained using SCLUS with Global Moran's I, $alpha = 0.5$ and Modified /Gaussian weighing function. Section 6.3.3 discusses some further characteristics of the learned PCTs.

Classification Tasks

In the following subsection, we compare and discuss the predictive performance of the SCLUS algorithm for the classification tasks.

In Tables 6.5 and 6.6, we report the precision and recall, estimated by 10-fold CV, obtained by applying SCLUS, CLUS and CLUS* to the spatially defined classification datasets. For this comparison, we have run SCLUS using the automatically chosen bandwidth and the Euclidean weighting schema. In terms of precision, Geary's C seems to lead to more accurate (precise) predictive models than Moran's I, whereas no difference is observed when analyzing the recall. When comparing errors obtained by SCLUS with errors obtained by CLUS and CLUS*, we note that, differently from in the case of regression tasks, there are only few cases where SCLUS outperforms the other systems. However, in the remaining cases, the precision and recall values of SCLUS are comparable to those of the competitors. Also for classification, Section 6.3.3 discusses some further characteristics of the learned PCTs.

Properties of the Models

In the following subsection, we present some properties of the models obtained by the SCLUS algorithm, for single and multi-target classification and regression tasks. Specifically, we present the model sizes, the autocorrelation of the errors of the predictive models and their learning times.

Table 6.7 shows the average size of the trees (number of leaves) built by SCLUS, CLUS, CLUS* and M5' Trees (only on the regression problems). The results suggest several considerations.

First, the average size of the trees learned by SCLUS is independent of the autocorrelation measure used. Exceptions are the datasets FF and Foixa_045, where trees obtained with Geary's C are significantly greater than their counterparts obtained with Moran's I.

Second, the average size of trees learned by SCLUS depends of the use of the autocorrelation measure used in the clustering phase. Trees induced by SCLUS by using only the autocorrelation measure in the clustering phase ($\alpha = 0$) are smaller, in over 90% of the cases, than trees computed by CLUS ($\alpha = 1$) and CLUS*. Similarly, trees induced by SCLUS using $\alpha = 0.5$ in the clustering phase are smaller, in 75 % of the cases than trees computed by CLUS ($\alpha = 1$) and CLUS*. This means that clustering spatial data according to autocorrelation guarantees smaller trees. Smaller trees are usually more comprehensible and simpler to interpret.

Table 6.3: *The RMSEs of the predictive models.* The models are obtained with SCLUS, CLUS, CLUS*, GWR, SVR and M5' Trees and estimated by 10-fold CV. Best results are given in bold. Results for NWE are multiplied by 10^3. The implementation of CLUS* supports only datasets that contain coordinates and not relative distances between the examples in the dataset which is the case with the River dataset. Note that SCLUS does not have such implementation problems, whereas CLUS does not use the spatial information at all.

Dataset	auto. chosen	SCLUS_Moran						SCLUS_Geary						CLUS	CLUS*	GWR	SVR	M5'
	b (%)	α=0			α=0.5			α=0			α=0.5			($\alpha = 1$)				
		Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.	Mod.	Gauss.					
FF	100	**42.82**	47.21	47.21	56.55	56.55	57.76	**42.82**	56.55	57.76	**42.82**	57.76	57.76	49.21	47.22	373.3	64.58	63.70
NWE	7.7	**2.16**	2.36	2.39	2.48	2.40	2.45	2.47	2.40	2.40	2.49	2.55	2.53	2.46	2.47	2.38	2.50	2.50
Foixa	64.2	2.53	2.55	2.41	**2.34**	2.55	2.44	2.56	2.58	2.55	2.55	2.56	2.59	2.65	2.52	2.54	2.95	2.86
GASD	2.5	0.18	0.18	0.17	0.17	0.18	0.17	0.18	0.18	0.17	0.18	0.17	0.17	0.16	0.16	0.35	**0.14**	0.16
MF	9.1	2.47	**2.04**	2.47	2.29	2.29	2.29	2.47	2.65	3.00	2.47	2.30	2.30	2.35	2.54	2.85	2.80	2.46
MS	4.8	**5.44**	5.66	5.59	5.81	5.81	5.81	6.60	5.50	5.50	5.80	5.83	5.55	5.64	6.68	6.77	8.60	6.37
River	100	**0.28**	0.33	0.33	0.30	**0.28**	**0.28**	**0.28**	0.31	0.30	0.32	0.30	0.30	0.30	-	0.52	0.30	0.30
Kenya	63.8	0.17	0.15	0.17	0.18	**0.14**	0.15	0.16	0.15	0.15	0.15	0.15	**0.14**	0.15	0.15	0.32	0.15	0.15
Malawi	8.1	0.12	0.11	0.10	**0.07**	0.09	0.09	0.12	0.12	0.11	0.08	**0.07**	0.09	1.21	1.26	0.09	0.32	0.38

Table 6.4: *A summary statistic of the number of wins minus the number of losses.* Each of the twelve configurations of the SCLUS algorithm (see Table 6.3) is compared to all other models (CLUS*, GWR, SVR and M5' Regression Trees). The statistics is an aggregate over all datasets for the regression task. A positive number means that SCLUS has more wins than losses than the other method, whereas a negative number means the opposite.

	SCLUS vs CLUS	SCLUS vs CLUS*	SCLUS vs GWR	SCLUS vs SVR	SCLUS vs M5'
Moran α=0.0, Euc.	3	2	7	5	3
Moran, α=0.0, Mod.	0	3	3	4	2
Moran, α=0.0, Gauss.	1	4	5	3	1
Moran, α=0.5, Euc.	-2	-2	6	4	4
Moran, α=0.5, Mod.	3	2	4	7	7
Moran, α=0.5, Gauss.	2	3	6	6	6
Geary, α=0.0, Euc.	-1	1	3	5	1
Geary, α=0.0, Mod.	0	-1	1	2	2
Geary, α=0.0, Gauss	1	-1	1	3	3
Geary, α=0.5, Euc.	0	1	5	4	4
Geary, α=0.5, Mod.	-3	-1	5	3	3
Geary, α=0.5, Gauss.	2	0	4	4	4

Table 6.5: *Precision of the models obtained with SCLUS, CLUS and CLUS*.* The models are estimated by 10-fold CV. Best results are given in bold.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*
	α=0	α=0.5	α=0	α=0.5	($\alpha = 1$)	
Kras	0.35	0.35	0.35	**0.42**	0.41	0.40
Prim	0.78	0.69	0.69	0.76	0.78	**0.80**
Foixa_01	0.78	0.76	0.75	0.79	0.77	**0.80**
Foixa_045	0.30	0.35	0.43	**0.46**	0.45	0.40

Table 6.6: *Recall of the models obtained with SCLUS, CLUS and CLUS*.* The models are estimated by 10-fold CV. Best results are given in bold.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*
	α=0	α=0.5	α=0	α=0.5	($\alpha = 1$)	
Kras	0.99	0.99	0.99	0.98	**1.00**	**1.00**
Prim	0.97	0.97	0.98	0.97	**1.00**	**1.00**
Foixa_01	0.98	0.98	0.98	0.98	0.98	0.98
Foixa_045	**1.00**	**1.00**	**1.00**	**1.00**	0.99	0.99

Table 6.7: *Average size of the trees learned by SCLUS, CLUS, CLUS* and M5' Trees.* The models are estimated by 10-fold CV, both for the classification and the regression task.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*	M5'
	α=0.0	α=0.5	α=0.0	α=0.5	($\alpha = 1$)		
FF	1.0	1.4	1.5	4.8	1.8	1.0	1.0
NWE	1.3	4.4	1.7	5.3	5.6	2.3	4.0
Foixa	1.8	2.3	2.4	2.3	4.7	6.1	3.0
GASD	10	31.2	8.0	28.9	27.7	23.8	49.0
MF	1.0	4.2	1.1	4.2	5.1	19.7	6.0
MS	1.4	6.1	1.1	5.3	6.0	19.2	6.0
River	1.0	1.1	1.0	1.0	1.1	-	1.0
Kenya	2.1	3.7	2.7	2.9	3.4	3.1	5.0
Malawi	12.9	67.8	10.5	69.8	30.1	26.4	70.0
Kras	3.5	3.4	4.4	3.2	6.3	5.0	-
Prim	3	4.0	1.3	3.9	4.2	4.0	-
Foixa_01	6.4	9.2	9.8	9.3	12.6	13.0	-
Foixa_045	12.7	15.7	21.2	22.0	15.0	18.0	-

Table 6.8: *Autocorrelation of the prediction errors.* Average autocorrelation of the prediction errors committed on the testing sets, performed by PCTs learned by SCLUS, CLUS, CLUS* and M5', as well as SVR and GWR. The models are estimated by 10-fold CV, both for the classification and the regression task. For each dataset, the best results (the smallest in absolute value) are given in bold.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*	GWR	SVR	M5'
	α=0.0	α=0.5	α=0.0	α=0.5	($\alpha = 1$)				
FF	**-0.02**	**-0.02**	**-0.02**	**-0.02**	1.00	**-0.02**	**-0.02**	**-0.02**	0.98
NWE	**0.00**	-0.01	0.06	0.07	0.84	-0.01	-0.01	-0.01	-0.01
Foixa	-0.02	-0.02	-0.02	**0.01**	0.96	-0.02	**0.01**	-0.03	-0.02
GASD	0.19	0.19	0.26	0.15	1.00	0.08	0.39	**0.01**	0.37
MF	**-0.01**	0.15	0.08	0.20	0.88	0.15	0.19	**-0.01**	0.14
MS	0.13	0.24	0.24	0.19	0.66	0.13	0.38	**-0.01**	0.34
River	-0.03	-0.03	-0.03	-0.03	-0.03	-	-0.03	-0.03	-0.03
Kenya	**-0.05**	-0.06	**-0.05**	-0.07	0.85	0.94	0.85	-0.08	0.92
Malawi	**0.03**	-0.20	0.32	-0.16	0.52	0.34	0.82	0.30	0.82

The models built with CLUS* are smaller (in 61 % of the cases) than the ones built with CLUS, but this is not systematic because CLUS* cannot handle different definitions of the regression problem that can arise from different definitions of the space which differs in different datasets. The predictions of the PCTs learned by CLUS* are more coherent in space in comparison with the PCTS learned by CLUS, but differently from SCLUS, this happens at the price of increasing the size of the tree models. While SCLUS can consider two different geographical regions that have the same distribution of attribute and target values including autocorrelation in one leaf of the tree, CLUS* will split these due to the spatial homogeneity. This is the reason of the increase of the tree size.

A final consideration concerns M5' that, although it builds model trees (theoretically a model tree is more accurate than a regression tree in prediction), the fact that it ignores the spatial dimension of the data leads to the construction of larger trees. In 83 % of the cases SCLUS builds trees that are smaller than the ones built with M5'.

In Table 6.8 we present the autocorrelation of the errors achieved by SCLUS, CLUS and CLUS*. In addition, we also present the errors obtained by using the competitive solutions M5' Regression Trees, SVR and GWR. Here, autocorrelation is computed by means of the Moran's I on the errors estimated on the test set. We analyze the obtained models in terms of this measure in order to show that PCTs learned by SCLUS can capture the autocorrelation present in the data and generate predictions that exhibit small (absolute) autocorrelation in the errors.

The analysis of the results reveals that SCLUS handles the autocorrelation better than CLUS. The autocorrelation of the errors of PCTs learned by SCLUS is much lower than the one obtained by CLUS.

Table 6.9: *The spatial dispersion SCLUS, CLUS and CLUS*. clusterings.* The models are estimated by 10-fold CV, both for the classification and the regression task. For each dataset, the best results (the smallest in absolute value) are given in bold.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*
	α=0.0	α=0.5	α=0.0	α=0.5	($\alpha = 1$)	
FF	**0.00**	0.02	**0.00**	**0.00**	0.04	**0.00**
NWE	**71.22**	962.72	201.54	562.71	222.56	348.20
Foixa	**25.69**	40.90	69.03	969.34	571.54	450.33
GASD	124146.78	232436.63	**76724.80**	204045.23	180685.45	88681.55
MF	**0.01**	0.20	0.21	0.45	0.55	0.66
MS	**0.17**	0.32	0.22	0.42	0.21	0.59
River	**0.00**	**0.00**	**0.00**	**0.00**	7.44	-
Kenya	**0.02**	0.03	1.49	1.48	0.03	**0.02**
Malawi	**1762.93**	7156.39	2879.73	10204.51	2812.67	2499.54
Kras	378.22	**366.74**	542.82	435.31	1438.42	890.46
Prim	594.15	619.56	625.70	618.91	553.88	**379.36**
Foixa_01	232.12	280.48	329.71	329.71	245.96	**200.39**
Foixa_045	571.80	743.26	1977.52	1296.96	1355.76	**522.83**

Table 6.10: *Learning times for SCLUS, CLUS, CLUS*, GWR, SVR and M5' Trees.* The time is given in seconds.

Dataset	SCLUS_Moran		SCLUS_Geary		CLUS	CLUS*	GWR	SVR	M5'
	α=0.0	α=0.5	α=0.0	α=0.5	($\alpha = 1$)				
FF	1.50	1.03	2.57	2.19	0.04	0.04	11.33	1.03	1.23
NWE	2.31	1.62	0.85	0.58	0.11	0.11	55.90	1.22	1.94
Foixa	0.69	0.49	1.88	2.01	0.02	0.02	31.25	0.49	0.88
GASD	27.86	20.79	20.25	23.56	0.04	0.04	1808.53	30.45	20.43
MF	2.68	1.39	2.08	2.81	0.04	0.03	24.17	1.00	2.25
MS	3.39	1.41	2.03	1.73	0.04	0.03	27.12	0.59	3.55
River	0.04	0.04	0.04	0.04	0.04	-	1.46	0.09	0.05
Kenya	1.18	1.08	1.19	1.34	0.02	0.02	12.53	0.02	1.20
Malawi	62.92	69.33	66.64	100.73	0.46	0.42	23.45	10.28	7.36
Kras	508.09	700.00	355.97	700.00	0.91	1.09	-	-	-
Prim	743.89	1154.38	449.87	699.79	1.53	1.48	-	-	-
Foixa_01	0.61	0.80	0.87	0.91	0.02	0.02	-	-	-
Foixa_045	0.62	0.77	0.85	1.02	0.02	0.02	-	-	-

In fact, coherently with the discussion reported in Section 4.2, SCLUS is able to correctly remove the effect of autocorrelation when making predictions. Thus, it is able to obtain spatially consistent predictions. This analysis also reveals that CLUS* is able to capture autocorrelation better than CLUS, but less than SCLUS. CLUS* gives lower autocorrelation of the errors than CLUS in 78 % of the cases and than SCLUS in 30 % of the cases. This is expected, according to the differences between SCLUS and CLUS*, already been discussed in this section. Moreover, as expected, the autocorrelation of the errors is lower when $\alpha = 0$.

Classical data mining methods like M5', have some level of spatial autocorrelation left in the errors which means that the errors obtained by using these methods may be underestimated (Davis, 1986) and the i.i.d. assumption violated. On the other hand, in the case of GWR, the level of spatial autocorrelation in the residuals of the linear model is much lower that M5', but higher than the one measured in the SCLUS models. Only in 17 % of the cases autocorrelation in the errors is lower than the autocorrelation left in the errors of the SCLUS models. This means that SCLUS can better capture the autocorrelation in the errors of the models, whereas GWR failures to include or adequately measure autocorrelated variables, in most cases.

The spatial dispersion of the clusterings produced by SCLUS, CLUS and CLUS* are reported in

Table 6.9. This measure is computed for the clustering as applied to the testing unseen data[1]. The analysis of these results reveals that, overall, SCLUS (in 70 % of the cases) and CLUS* (in 75 % of the cases) clusterings are more compact than the CLUS clusterings. Moreover, as expected, most of the PCTs induced by SCLUS with $\alpha = 0$ are more compact than PCTs induced with $\alpha = 0.5$, as the former uses only spatial autocorrelation as a splitting criterion when building the PCT. For $\alpha = 0.5$ in SCLUS, the produced clusters are less dispersed (have higher spatial dispersion), than those learned by CLUS*.

Finally, the overall spatial dispersion of the clusters induced by CLUS* and SCLUS with $\alpha = 0.0$ are comparable. This is expected, considering how the spatial dispersion is computed. In fact, its definition relies on the average Euclidean intra-distance for each pair of examples that fall in the same cluster and CLUS*, by considering coordinates as target variables, tends to add splits to the tree which group examples with similar (close) coordinates. Note that the models that have better spatial dispersion tend to have lower autocorrelation of their errors.

Finally, Table 6.10 reports the average learning times for SCLUS, CLUS, CLUS*, GWR, SVR and M5' Regression Trees. Overall, the smallest learning times are obtained by using the CLUS algorithm. The learning times for CLUS* are similar (slightly larger) than the running times of CLUS, as in this configuration CLUS is run by considering the spatial coordinates as responses and the time complexity of PCTs induction remains roughly the same. The learning times for SCLUS are longer than the learning times for CLUS, because the consideration of the autocorrelation introduces additional computations and increases the complexity of building a PCT. This is in line with the time complexity analysis reported in Section 6.2.4. The learning times for SVR and M5' Trees are smaller than those of SCLUS and longer than those of CLUS. The learning times for GWR are significantly longer than those of all other algorithms, because GWR creates many different local models.

Comparison of the predictive models

Besides having different predictive performance (accuracy), the obtained models also have different structure. In this section, we show the differences among the models learned by CLUS and SCLUS on ecological datasets (such as MF, MS and FOIXA). In Figure 6.1(a), 6.1(b), Figure 6.2(a) and 6.2(b), we analyze the regression trees learned from the MF and MS datasets by using CLUS/SCLUS (with Global Moran's I and automatically estimated bandwidth). These datasets contain measurements of pollen dispersal (crossover) rates from two lines of plants (oilseed rape): the transgenic male-fertile (MF) and the non-transgenic male-sterile (MS). The measurements are taken at sampling points concentrically distributed around a central donor field.

We observe that the pair of trees learned by CLUS from the MF and MS datasets (see Figure 6.1(a) and Figure 6.2(a)), as well as the pair of trees learned by SCLUS from same datasets (see Figure 6.1(b) and Figure 6.2(b)), leave the same split nodes at their roots. The MF and MS datasets share the same predictor variables and there exists a strong dependence between response variables of both datasets.

The trees learned by the same algorithm, which uses the same heuristic function, are thus similar across the two datasets. CLUS and SCLUS, which use different heuristics, learn differently structured trees from the same dataset. In both datasets, SCLUS chooses the spatially-aware distance predictor variable (i.e., the distance of the sampling point from the center of the donor field) at the top level(s) of the tree, with other predictors at lower levels of the tree (see the case of the MS/MF dataset in Figures 6.2(b)/6.1(b)). In the models obtained by CLUS, the distance to the donor field is used only to specify the target values at the bottom levels of tree, whereas the visual angle between the sampling plot and the donor field is chosen at the top level(s) of the tree. The choice of the distance attribute in SCLUS seems to be the one that better captures a global trend in the data, i.e., the concentric distribution of the pollen

[1]Zero values in Table 6.9 are due to the rounding down of intra-distances which are less than 10^{-3} to zero

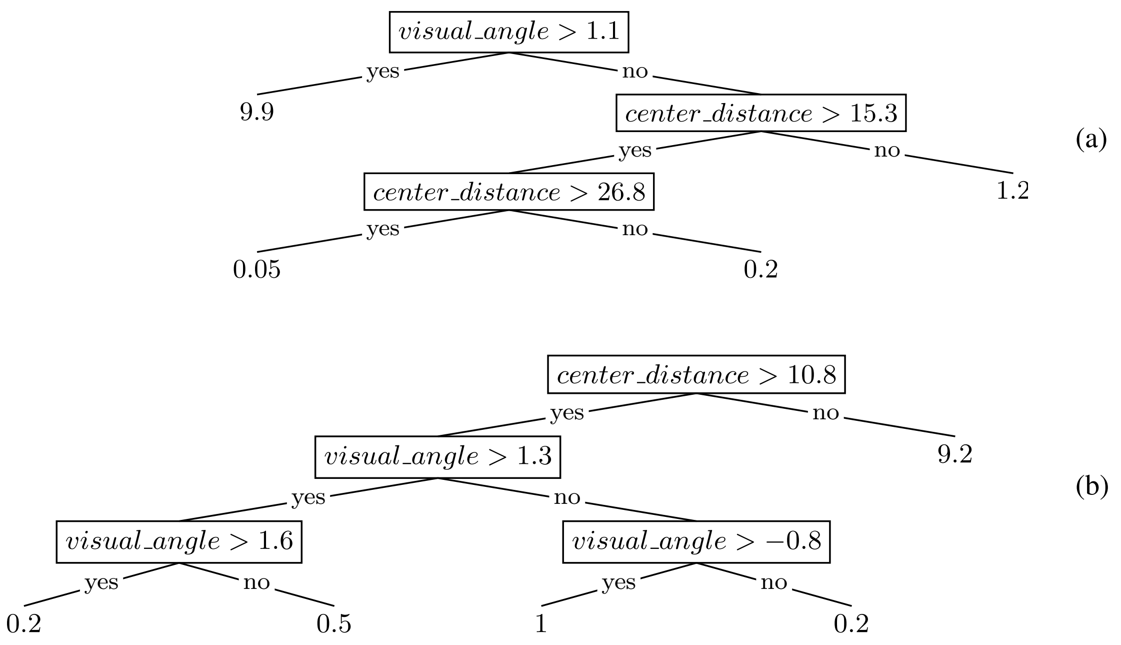


Figure 6.1: *PCTs for the MF dataset.* Two PCTs learned on one of the training folds (a) The ordinary PCT learned by CLUS (b) The spatially-aware PCT learned by SCLUS.

dispersal (crossover) rates (see Figure 6.4(a)). This makes the spatially-aware PCT more interpretable and understandable than the corresponding ordinary PCT.

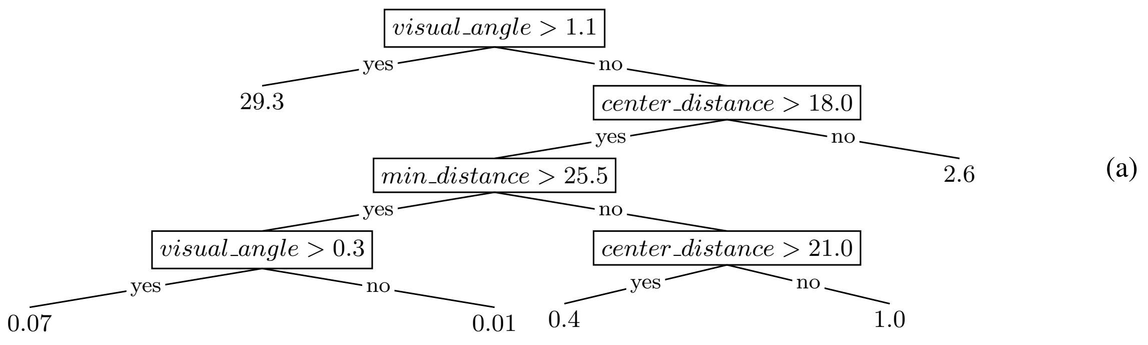
Further on, in Figure 6.3(a) and Figure 6.3(b), we show the regression trees for the FOIXA dataset, obtained by using CLUS and SCLUS (with Global Moran I and b=20 %), respectively. This dataset contains measurements of the rates of outcrossing at sampling points located within a conventional field, due to pollen inflow from the surrounding genetically modified (GM) fields.

Similarly as for the MF and MS datasets, CLUS and SCLUS learn differently structured trees. The models have different splits at the root node. The tree obtained by CLUS has the size of the surrounding GM fields (totalgmarea) at the root of the tree, as well as in most of the nodes of the tree, which means that the predictions in the leaves are based only on the values of one predictor variables and the other variables are practically of no importance. In contrast, the spatially-aware PCT has the number of surrounding GM fields (numberGMfields) at the root of the tree and the size of the surrounding GM fields (totalgmarea) in the other nodes of the tree. In this case, the predictions in the leaves are based on the values of two predictor variables and according to domain expertise, as in the case of MF/MS dataset, the spatially-aware PCT is more interpretable and understandable than the corresponding ordinary PCT.

A more detailed analysis of the training examples falling in the leaves of both the ordinary PCT and the spatially-aware PCT revealed that the leaves of both trees cluster examples with similar response values (this is due to the variance reduction). However, the training examples falling in the leaves of the spatially-aware PCTs are also close in space. This guarantees spatially smoothed predictions, where the predictions that are close to each other in space tend to have similar values.

Visual Differences in the Predictions Made by the Models

In order to visualize the differences between the predictions of the SCLUS models and the models obtained by the other approaches, we use the MS, MF and Foixa datasets. The maps given in Figures 6.4



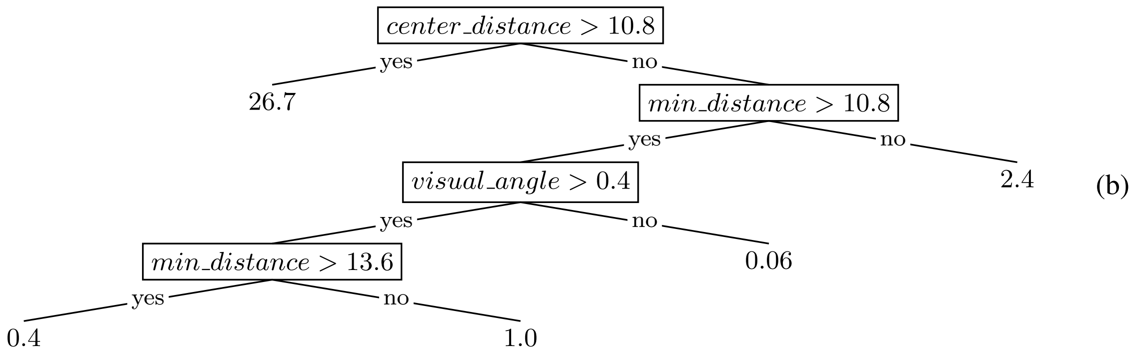
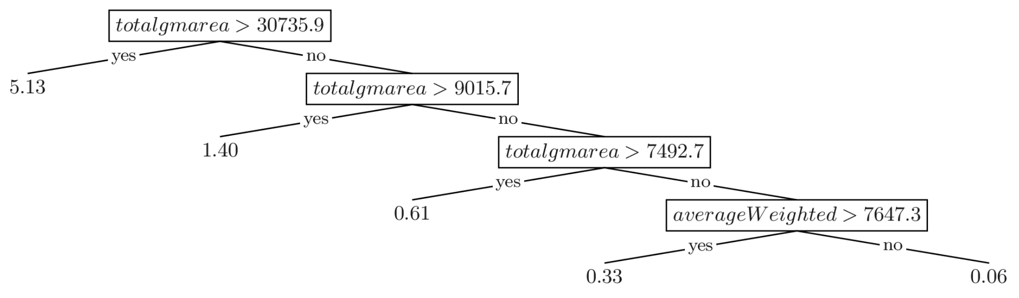


Figure 6.2: *PCTs for the MS dataset.* Two PCTs learned on one of the training folds (a) The ordinary PCT learned by CLUS (b) The spatially-aware PCT learned by SCLUS.



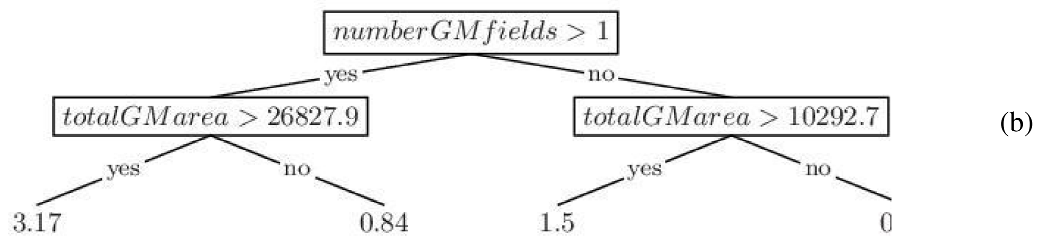


Figure 6.3: *PCTs for the Foixa dataset.* Two PCTs learned on one of the training folds (a) The ordinary PCT learned by CLUS (b) The spatially-aware PCT learned by SCLUS.

and 6.5 represent the predictions for the outcrossing rates for the testing (unseen) examples for the MF and MS datasets, respectively.

For the MF dataset, the real (measured) target are given in Figure 6.4(a), whereas the predictions obtained by the models learned with CLUS, SCLUS and GWR are given in Figures 6.4(b), 6.4(c) and Figure 6.4(d), respectively. The predictions made by CLUS (Figure 6.4(b)) tend to over-generalize the training data in the attempt of maximizing the variance reduction. This results in forming ideal concentric circles (Figure 6.4(a)), which roughly approximate the spatial distribution of the target variable. However, by simultaneously analyzing Figure 6.1(a) and Figure 6.4(b), it is possible to see that this spatial awareness is simply due to the presence of the attribute *center_distance* in the tree. On the other hand, the predictions made by SCLUS (Figure 6.4(c)) are spatially-aware and, at the same time, follow the spatial distribution of the real target values much more closely. GWR (Figure 6.4(d)) has poor performance and fails to capture the global spatial distribution of the target. This is due to the fact that GWR accounts only for local properties of data.

For the MS dataset, the predictions obtained of the models learned with CLUS, SCLUS and GWR are given in Figure 6.5(b), 6.5(c) and Figure 6.5(d), respectively. These figures suggest the same conclusions we have drawn from the MF dataset. However, the predictions made by CLUS and SCLUS appear more similar for the MS than for the MF dataset.

Finally, the geographical maps given in Figure 6.6 present the predictions for the outcrossing rates for the testing (unseen) examples for the Foixa dataset. The real target is given in Figure 6.6(a), whereas the predictions obtained by models learned with CLUS, SCLUS, and GWR are shown in Figures 6.6(b), 6.6(c) and 6.6(d), respectively. The map of predictions obtained by using the model learned by SCLUS for the Foixa dataset (Figure 6.6(c)) shows that the predictions are smoother than the ones of the CLUS model. This means that SCLUS predictions that are close to each other in space tend to have similar values (very high/low outcrossing rates are very close to each other). When plotted on a map, they form a nice smooth continuous surface without sharp edges and discontinuities. In contrast, there are sharp

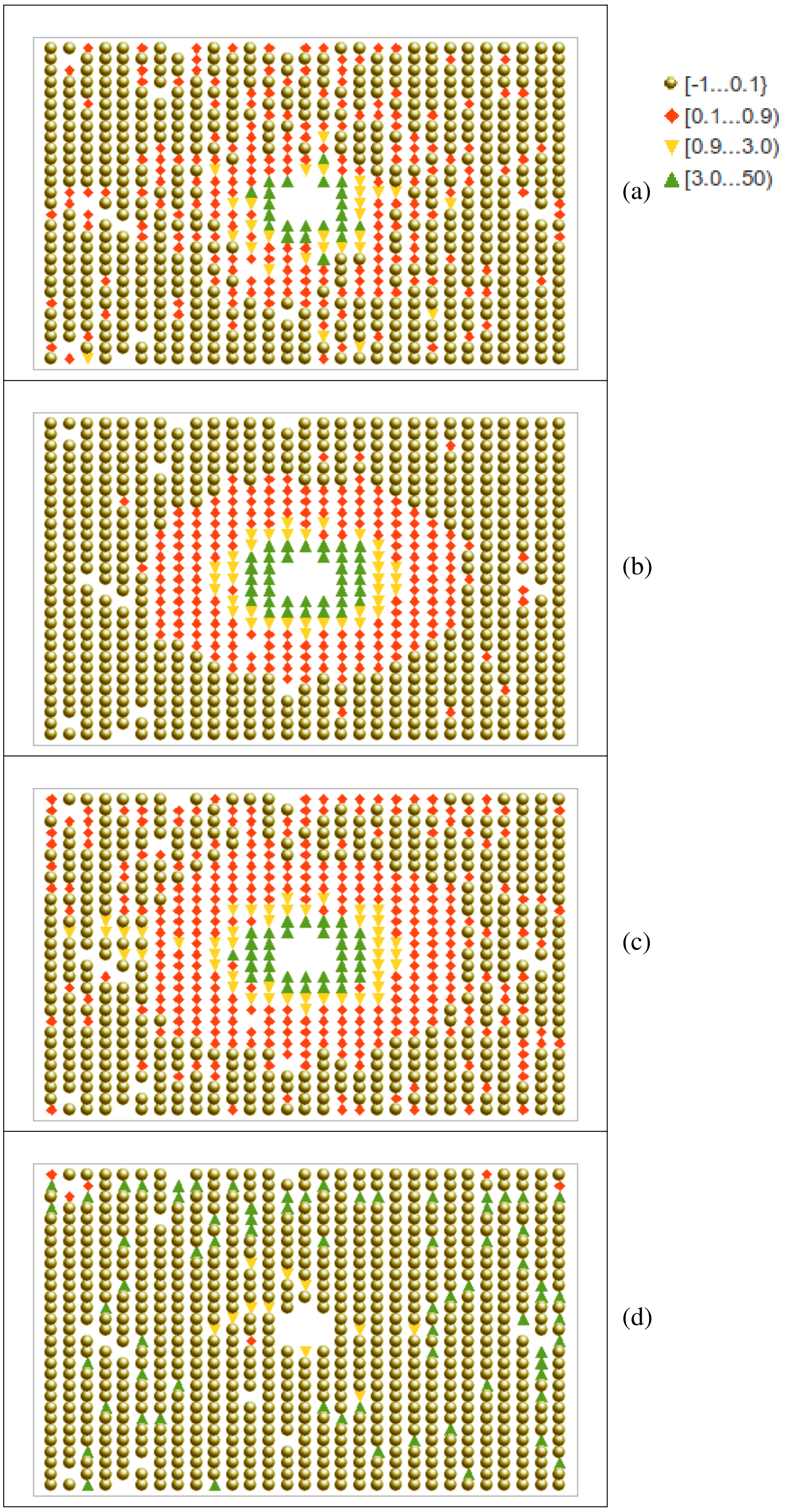


Figure 6.4: *Predictive models for the MF dataset.* The pollen dispersal (crossover) rates of the MF dataset: (a) measured values (b) values predicted by the CLUS model, (c) values predicted by the SCLUS model and (d) values predicted by the GWR model.

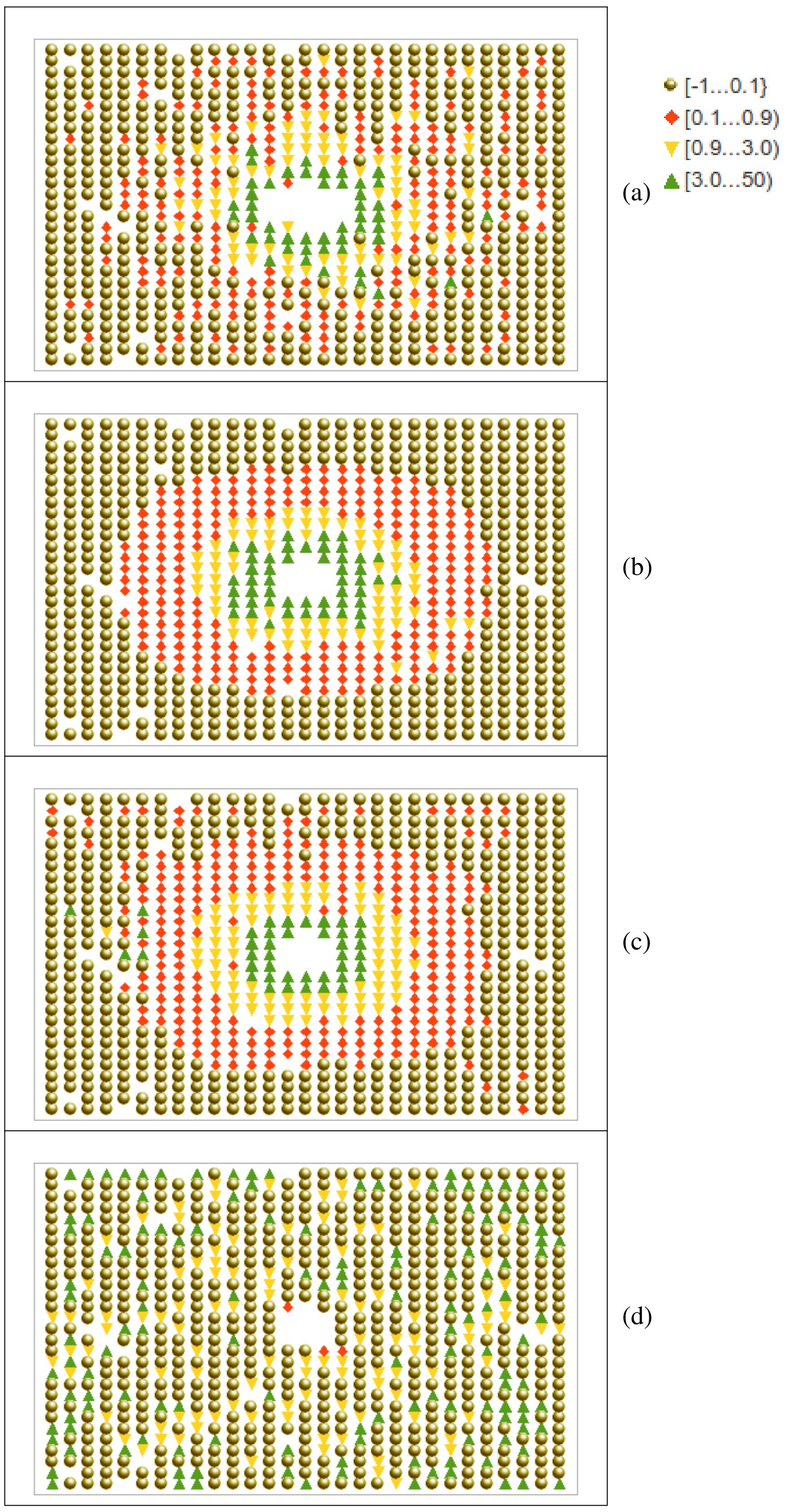


Figure 6.5: *Predictive models for the MS dataset.* The pollen dispersal (crossover) rates of the MS dataset: (a) measured values (b) values predicted by the CLUS model, (c) values predicted by the SCLUS model and (d) values predicted by the GWR model.

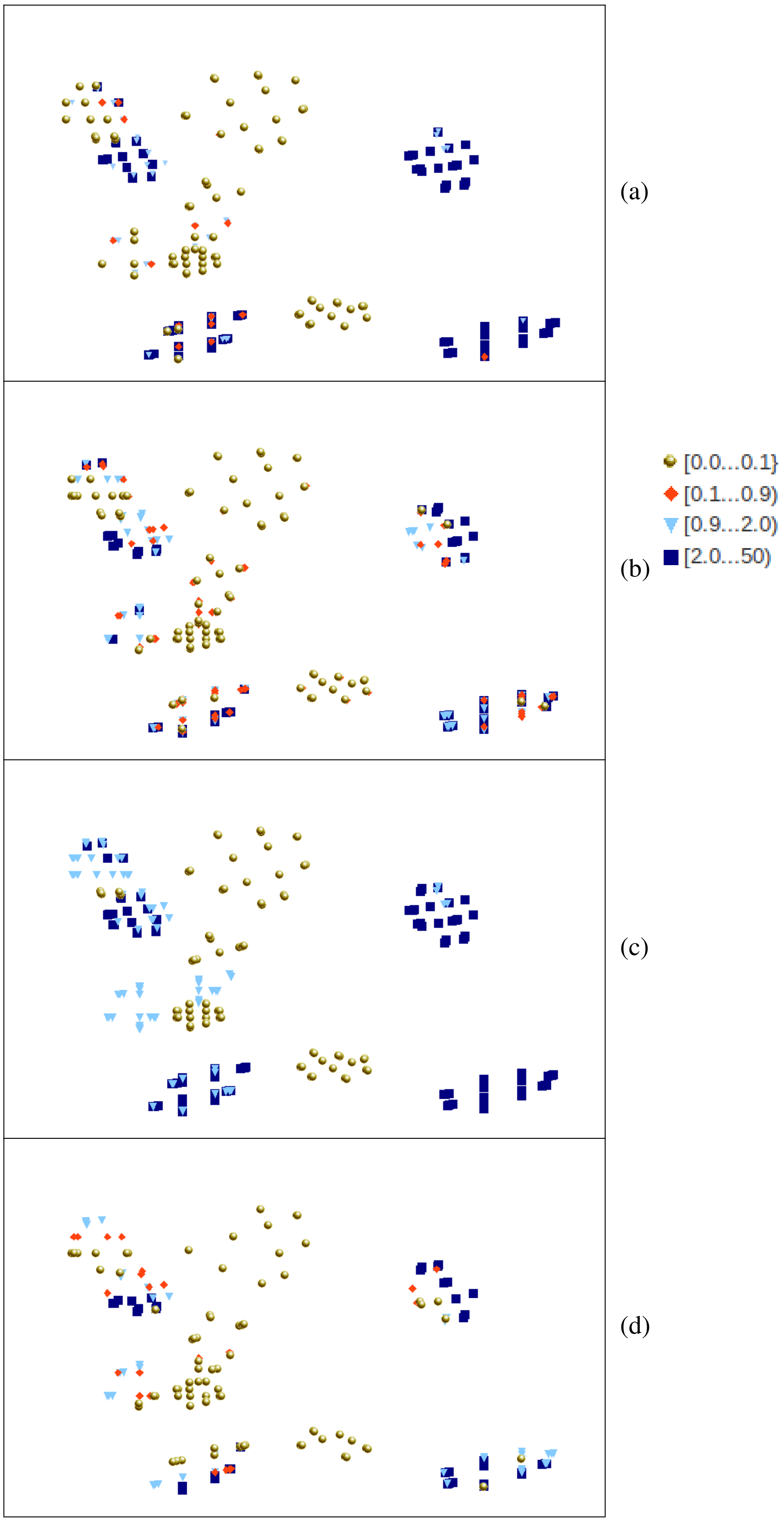


Figure 6.6: *Predictive models for the Foixa dataset.* The outcrossing rate for selected sampling points of the FOIXA dataset: (a) measured values (b) values predicted by the CLUS model, (c) values predicted by the SCLUS model and (d) values predicted by the GWR model.

and discontinuousness in the CLUS predictions, so that the corresponding map looks like a mixture of "salt and pepper". In addition, the SCLUS models are more accurate (in terms of the obtained errors, Table 6.3) and more interpretable than the competitors' models (Figure 6.3(b) and Figure 6.6(c)). The map of predictions for the Foixa dataset obtained by using the model learned by GWR (Figure 6.6(d)) also shows sharp edges and discontinuities. This is due to the fact that GWR builds local models at each point: these are independent of the models built at the neighboring points. While the GWR models exploit the positive autocorrelation between neighboring points and in this way accommodate stationary autocorrelation, their predictions are still less accurate than those obtained with the SCLUS models. In sum, the SCLUS models are more useful than those of the (both spatial and a-spatial) competitors because both the tree models and the predictions are more realistic, easier to interpret, and more accurate.

6.4 Summary

In this chapter, we proposed an approach that builds Predictive Clustering Trees (PCTs) and explicitly considers non-stationary spatial autocorrelation. The novelty of our approach is that it approaches clustering by maximizing both variance reduction and cluster homogeneity (in terms of autocorrelation), when the addition of a new node to the tree is considered. The effects of autocorrelation are identified and taken into account, separately at each node of the tree. The resulting models adapt to local properties of the data, providing, at the same time, spatially smoothed predictions. Due to the generality of PCTs, our approach works for different predictive modeling tasks, including classification and regression, as well as some clustering tasks.

The approach can consider different weighting schemes (degrees of smoothing) when calculating spatial autocorrelation as well as different sizes of neighborhoods (bandwidth). A procedure for the automated determination of the appropriate bandwidth is also explored in our study. A novelty of our approach is in the use of well-known measures of spatial autocorrelation, such as Moran's I and Geary's C. Previous related work on using autocorrelation in decision trees was based on special purpose measures of spatial autocorrelation, such as spatial entropy, and were limited to classification.

An extensive experimental evaluation has been performed to empirically prove the effectiveness of the proposed approach. Nine geo-referenced data sets are used for regression tasks while four geo-referenced data sets are employed for classification tasks. The experimental results show that the proposed approach performs better than standard spatial statistics techniques such as geographically weighted regression, which considers spatial autocorrelation but can capture global regularities only. SCLUS can identify autocorrelation, when present in data, and thus generate predictions that exhibit smaller autocorrelation in the errors than other methods: It can also generate clusterings that are more compact and trees that are smaller in size. Furthermore, the spatial maps of the predictions made by SCLUS trees are smother and do not require further post-smoothing for successful use in practice.

Several directions for further work remain to be explored. The automated determination of the parameter α that sets the relative importance of variance reduction and autocorrelation during tree construction deserves immediate attention. In a similar fashion, one might consider selecting an appropriate spatial autocorrelation measure. Finally, we plan to focus on multi-target problems, explicitly taking into account autocorrelation on the combination of several target variables.

7 Learning PCTs for Network Autocorrelated Data

In this chapter, we propose an extension of the predictive clustering framework that works in a network setting. The system that we propose is called **NCLUS** for Network Predictive Clustering System. It explicitly considers network autocorrelation when learning predictive clustering models.

Since regression inference in network data is a challenging task, the proposed algorithm deals with single and multi-target regression tasks. Its development was motivated by the development of the spatially aware classification and regression PCTs, presented in Chapter 6. Moreover, spatial datasets can be represented as spatial data networks and dealt with the NCLUS algorithm that we discuss in this section.

First, we motivate the research approach presented in this chapter. Then we describe the proposed algorithm in details and investigate different autocorrelation measures and their use in the case of network structured data. Finally, we compare and discuss the predictive performance of the NCLUS algorithm for single and multi-target regression tasks.

The experiment presented in this chapter are divided in two parts according to the type of the data that they refer to. The first part are experiments performed on social network data, whereas the second part are experiments performed on networks that come from spatial data. We discuss the evaluation results in terms of their accuracy, as well as in terms of the properties of the predictive models by analyzing the model sizes, the autocorrelation of the errors of the predictive models and their learning times.

7.1 Motivation

In network studies, the violation of the i.i.d assumption has been identified as one of the main reasons responsible for the poor performance of traditional data mining methods (LeSage and Pace, 2001; Neville et al, 2004). To remedy the negative effects of the violation of the independence assumption, network autocorrelation has to be explicitly accommodated in the learned models.

In particular, recent research has explored the use of collective inference techniques to exploit this phenomenon when learning predictive models. According to (Sen et al, 2008), collective inference refers to the combined classification of a set of nodes using the attribute value of a node and the labels of interlinked nodes. This means that, differently from traditional algorithms that make predictions for data instances individually, without regard to the relationships or statistical dependencies among instances, collective inference techniques collectively predict the labels of nodes simultaneously using similarities that appear among groups of interlinked nodes.

We develop an approach for modeling non-stationary network autocorrelation by using PCTs. PCTs are learned by plugging distances, which exploit the network structure, in the PCTs induction and obtaining predictive models that will be able to deal with autocorrelated data. This is done by maximizing the variance reduction and maximizing cluster homogeneity (in terms of autocorrelation), at the same time when evaluating the candidates for adding a new node to the tree, thus improving the predictive performance of the obtained models.

The contributions of this chapter are in:

- The investigation of the different autocorrelation measures introduced in the literature and their use in the case of network structured data;

- The development of an approach that uses these autocorrelation measures in the induction of PCTs by taking into account variations in the global/local data distribution across the network;
- An extensive evaluation of the effectiveness of the proposed approach on regression problems (single- or multi-target) in real network data and network data derived from spatial data.
- New empirical evidence is provided on the importance of considering network autocorrelation in predictive tasks and in particular on the ability of the proposed method to capture autocorrelation within the learned PCTs by analyzing the autocorrelation of the errors on an extensive set of different data.

The algorithm proposed in this chapter extends the predictive clustering framework implemented in the CLUS system (Blockeel et al, 1998). It allows CLUS to handle network data in the setting outlined in Section 2.3.1. The software and data are available for download at: `http://http://kt.ijs.si/daniela_stojanova/DMKDpaper/`.

7.2 Learning PCTs by taking Network Autocorrelation into Account

In this section, we provide a detailed description of the implemented algorithm for learning PCTs by taking network autocorrelation into account. Besides the algorithm itself, we also explore the properties of the network autocorrelation in NCLUS, emphasize the importance of this phenomenon. In addition, we stress the importance of the selection of the bandwidth parameter, as an important part of the HCLUS algorithm. At last, we give an analysis of the time complexity of the proposed algorithm.

7.2.1 The Algorithm

Before we proceed to describe the top-down induction algorithm for building PCTs from network data, we define the network. Formally, the network is defined as an undirected edge-weighted graph *G=(V,E)*, where V is the set of labeled *nodes*, while $E \subseteq \{\langle u,v,w\rangle \mid u,v \in V, w \in \mathbb{R}^{+}\}$ is the set of *edges* (i.e., the context space D), such that each edge $u \leftrightarrow v$ is assigned a nonnegative real number w, called the *weight* of the edge. It can be represented by a symmetric adjacency matrix $\mathbf{W}$, whose entries are positive ($w_{ij} > 0$) if there is an edge connecting i to j in G, and zero ($w_{ij} = 0$) otherwise. Here, edge weights express the strength of the interactions between proteins. Although the proposed method works with any non-negative weight values, we anticipate that in our experiments only binary $(0/1)$ weights could be used, due to limitations of available data on protein interactions. The edges represent the context space D which in this case in defined only using the distance function $d(\dot{;})$.

The Algorithm 3 is a recursive algorithm which takes as input the network $G = (V,E)$ and the function $\eta : V \mapsto \mathbf{X} \times \mathbf{Y}$ and partitions the set of nodes V until a stopping criterion is satisfied (Algorithm 3 line 2). Since the implementation of this algorithm is based on the implementation of the CLUS algorithm, we will call this algorithm NCLUS (for Network CLUS).

The main loop (Algorithm 3, lines 7-14) searches for the best attribute-value test c^* that can be associated to a node t. The algorithm associates the best test c^* to the internal node t and calls itself recursively to construct a subtree for each subnetwork in the partition $\mathcal{P_V}^*$ induced by c^* on the training nodes.

Possible tests are of the form $X \leq \beta$ for continuous attributes, and $X \in \{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ (where $\{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ is a subset of the domain Dom_X of X) for discrete attributes. For continuous attributes, possible values of β are found by sorting the distinct values of X in the training set associated to t, then considering a threshold between each pair of adjacent values. Therefore, if the cases in t have d distinct values for X, at most $d-1$ thresholds are considered.

Algorithm 3 *Top-down induction of NCLUS.*

```
1: procedure NCLUS($G = (V, E), \eta(\cdot)$) returns tree
2: if stop($V, \eta(\cdot)$) then
3:     return leaf(Prototype($V, \eta(\cdot)$))
4: else
5:     $(c^*, h^*, \mathcal{P}^*, \mathcal{P_V}^*) = (null, 0, \emptyset, \emptyset)$
6:     $C = \{\eta(v) \mid v \in V\}$
7:     for each possible Boolean test $c$ according to values of $\mathbf{X}$ on $C$ do
8:         $\mathcal{P} = \{C_1, C_2\}$ partition induced by $c$ on $C$
9:         $\mathcal{P_V} = \{V_1, V_2\}$ = partition induced by $\mathcal{P}$ on $V$;
10:        $h = \frac{\alpha}{q} \sum_{Y \in \mathbf{Y}} \Delta_Y(C, \mathcal{P}) + \frac{(1-\alpha)}{q} \sum_{Y \in \mathbf{Y}} A_Y(G, \eta(\cdot), \mathcal{P})$
11:        if $(h > h^*)$ then
12:            $(c^*, h^*, \mathcal{P}^*, \mathcal{P_V}^*) = (c, h, \mathcal{P}, \mathcal{P_V})$
13:        end if
14:    end for
15:    $\{V_1, V_2\} = \mathcal{P_V}^*$
16:    $tree_1$ = NCLUS($(V_1, E), \eta(\cdot)$)
17:    $tree_2$ = NCLUS($(V_2, E), \eta(\cdot)$)
18:    return node($c^*, tree_1, tree_2$)
19: end if
```

For discrete attributes, possible subsets of values are selected by relying on a non-optimal greedy strategy (Mehta et al, 1996). Starting with an empty set $Left_t = \emptyset$ and a full set $Right_t = Dom_X$, where Dom_X is the domain of X, this iterative procedure moves one element from $Right_t$ to $Left_t$, such that the move results in an increased reduction of variance for the target variable Y. This differs from the classical solution by (Breiman et al, 1984), where some ordering on the possible values of Dom_X is defined apriori, according to the data distribution. However, the classical solution cannot deal with multi-target prediction tasks as PCTs can. If the examples in t have d distinct (discrete) values, $\sum_{i=2}^{d-1} i = \frac{d^2 - 3d}{2}$ splits are considered.

The algorithm evaluates the best split according to the formula (7.1) reported in Algorithm 3, line 10.

$$h = \frac{\alpha}{q} \sum_{Y \in \mathbf{Y}} \Delta_Y(C, \mathcal{P}) + \frac{(1-\alpha)}{q} \sum_{Y \in \mathbf{Y}} A_Y(G, \eta(\cdot), \mathcal{P}) \tag{7.1}$$

This formula is a linear combination of the variance reduction $\Delta_Y(C, \mathcal{P})$ and the autocorrelation measure $A_Y(G, \eta(\cdot), \mathcal{P})$.

In spatial analysis, several spatial autocorrelation statistics have been defined. The most common ones: Global Moran's I and Global Geary's C (Legendre, 1993). They are used in the case of continuous target variable. In the context of our algorithm we use Global Moran's I given by the Equation 3.4, *RA* given by the Equation 3.36 and *CI* given by the Equation 3.35 as network autocorrelation measures presented in Section 3.4. Note that we redefined the measures of spatial autocorrelation, Global Moran's I and Global Geary's C, presented in Section 3.2.

Both, variance and autocorrelation are computed for the Y variable (the class) over the cluster C. In the case of multiple target variables, the average values of both, variance reduction $\Delta_Y(C, \mathcal{P})$ and autocorrelation $A_Y(G, \eta(\cdot), \mathcal{P})$ are taken over the set of target variables, where each target variable contributes equally to the overall h value.

The influence of these two parts of the linear combination when building the PCTs is determined by a user-defined coefficient α that falls in the interval [0, 1]. When $\alpha = 0$, NCLUS uses only autocorrelation, when $\alpha = 0.5$ it weights equally variance reduction and autocorrelation, and when $\alpha = 1$ it ignores autocorrelation and works such as the original CLUS algorithm.

However, since they all range in different intervals (but are consistently monotonic), it is necessary to appropriately scale them. Since variance reduction is non-negative, we decided to scale them both to the interval [0, 1], where 1 means high positive autocorrelation and 0 means high negative autocorrelation. The choice of the scaling interval does not affect the heuristic computation, therefore other scaling intervals are possible as well, provided that, in all cases, the same scaling is performed and the monotonicity of the scaled measure is maintained.

For example, for Moran's I,$A_Y(G,\eta(\cdot),\mathcal{P})$ is defined as:

$$A_Y(G,\eta(\cdot),\mathcal{P}) = \sum_{C_k \in \mathcal{P}} \frac{|C_k|}{|C|} \widehat{I_Y}(C_k) \tag{7.2}$$

where $\widehat{I_Y}(D_k)$ is the scaled Moran's I computed on D_k.

Moreover, in order to guarantee a fair combination of the variance reduction and the autocorrelation statistic $A_Y(G,\eta(\cdot),\mathcal{P})$, we also need to scale the variance reduction to the interval [0, 1]. For that purpose, we use a common scaling function:

$$\widehat{\Delta_Y}(C,\mathcal{P}) = \frac{\Delta_Y(C,\mathcal{P}) - \Delta min}{\Delta max - \Delta min} \tag{7.3}$$

where Δmax and Δmin are the maximum and the minimum values of $\Delta_Y(C,\mathcal{P})$ for a particular split.

The search stops when one of the two defined stopping criteria is satisfied. The first criterion stops the search when the number of examples in a leaf is smaller than $\sqrt{N}$, which is considered a good locality threshold that does not lose too much in accuracy (also for rule based classifiers) (Gora and Wojna, 2002). The second criterion uses the exact Fisher test (F-test) to check whether a given split/test in an internal node of the tree results in a reduction in $\Delta_Y(C,\mathcal{P})$ that is statistically significant at a given significance level. To choose the optimal significance level among the values in the set $\{1, 0.125, 0.1, 0.05, 0.01, 0.005, 0.001\}$, we optimize the MSE obtained with an internal 3-fold cross validation.

When the first stopping criterion is not satisfied, we evaluate the second criterion[1]. In the case one of the two stopping criteria is satisfied, the algorithm creates a leaf and labels it with a predictive function (in this case the average of the response variable(s)) defined over the examples falling in that leaf. When predicting multiple response variables, the predictive function returns the vector of the averages of the responses values.

In NCLUS, a pruning strategy to prevent trees from over-fitting data is implemented. This strategy is the pessimistic error pruning strategy, which is also implemented in several regression/model tree learners (including M5' and CLUS). According to this strategy, a subtree is kept only if the error at the leaf is greater than the error of the subtree multiplied by a scaling factor, which takes into account the statistical support and the complexity of the model (Wang and Witten, 1997). The results that we present here are those of the pruned tree models learned by NCLUS.

7.2.2 Exploiting the Properties of Autocorrelation in NCLUS

The consideration of autocorrelation in clustering has been the subject of some recent work in spatial clustering (Glotsos et al, 2004) and network clustering (Jahani and Bagherpour, 2011). Motivated by the

[1]In this way, the F-test is performed on large enough number of examples to make reasonable the assumption of normal distribution.

demonstrated benefits of autocorrelation, we exploit some properties of autocorrelation to improve the quality of the PCTs.

The use of autocorrelation in predictive clustering offers several advantages since it allows to:

- determine the strength of the network effects on the variables in the model;
- consider tests on assumptions of stationarity and heterogeneity in the network;
- identify the possible role of the network interaction/distance decay on the predictions associated to each of the nodes of the tree;
- focus on the "node neighborhood" to better understand the effects that it can have on other neighborhoods and vice versa.

These properties, identified by (Arthur, 2008), hold for spatial clustering. However, they also hold for the case of PCTs. Moreover, as recognized by (Griffith, 2003), autocorrelation implicitly defines a zoning of a (spatial) phenomenon and reduces the effect of autocorrelation in the prediction errors.

Variance reduction leads to more accurate models since it reduces the error on the training set. However, it does not have all the properties implicitly introduced by autocorrelation. This is due to the fact that variance reduction does not take the distribution of connections among the nodes in the network into account.

With regard to the statistical properties of the measures of autocorrelation, most of the theoretical research in Statistics and Econometrics exploits the so called "autoregressive model" in order to measure autocorrelation in networks (or lattices, as they are sometimes called) (Griffith, 2003). More formally, the autoregressive model is defined as:

$$\hat{e}_i = \rho \sum_j w_{ij}\, e_j + \varepsilon_i \tag{7.4}$$

where $e_j = Y_j - \overline{Y}$ is the prediction error (where prediction is based on the average), ρ is a parameter that expresses the network dependence, w_{ij} are the elements of the neighborhood matrix W and the error ε_i follows a Gaussian (normal) distribution (Engle, 1982).

In this case, the informal notion of network dependence is often implicitly based on an autoregressive framework, where the goal is to assess the predictive ability of the neighboring values of the data. As recognized by (Li et al, 2007), in order to informally assess the strength of the network dependence, exploratory data analysis should be based on estimating ρ in the autoregressive model (see Equation 7.4). This means that the parameter ρ plays a crucial role in representing autocorrelation in the data.

One common solution for estimating ρ is to use a modified least squares estimator, which is the solution to the following quadratic equation in ρ:

$$\mathbf{e}^T(\mathbf{I} - \rho\mathbf{W})^T\mathbf{W}(\mathbf{I} - \rho\mathbf{W})\mathbf{e} = 0 \tag{7.5}$$

where $\mathbf{W}$ is the matrix representation of w_{ij}, $\mathbf{I}$ is the identity matrix and $\mathbf{e}$ is the vector of e_i values.

Although this estimator is consistent (Li et al, 2007), its computation is not straightforward. Particularly, it would require the computation of the Maximum Likelihood Estimator of ρ which is impractical when large datasets need to be processed (Li et al, 2007). Therefore, instead of computing an estimate of ρ, Moran's I is commonly used in spatial data mining applications. Indeed, as proved by (Jin, 2010), under the assumption that $w_{ij} = w_{ji}$ ($\mathbf{W}$ is symmetric), Moran's I is monotonic in ρ. Unfortunately, this result is not valid when $w_{ij} \neq w_{ji}$ (some counterexamples can be found). Moreover, (Li et al, 2007) empirically proved that Moran's I is a good (not unbiased) estimator of ρ in the case when ρ approaches zero. This means that Moran's I is a good indicator of the network dependence under some conditions.

The same conclusions can be drawn for the Relational Autocorrelation (*RA*) that is similar to the Moran's I, but considers the weights in a binary form. On the contrary, the Connectivity Index (*CI*) cannot be considered as a good estimator of ρ and, for this reason, we expect different results by varying the autocorrelation measure used in the predictive model. This analysis also confirms that a model that is able to take autocorrelation into account should lead to small values of ρ, that is, according to (see Equation 7.4), it should lead to a reduction of the effect of an error in the neighborhood.

7.2.3 Choosing the Bandwidth

The choice of the bandwidth (denoted by b in Equation 2.5) is perhaps the most critical decision to be taken in the modeling process. This parameter controls the degree of smoothing, with larger bandwidths causing stronger smoothing. An oversmoothed model will predict similar values of the target variable all across the network, while an undersmoothed model will predict values with so much local variation that it would be difficult to determine whether there are any patterns at all. At the same time, this parameter influences the calculation of autocorrelation.

The bandwidth may be defined manually or by using some adaptive method on the whole training network, such as cross validation and the corrected Akaike Information Criterion (AIC) used in GWR (Fotheringham et al, 2002). A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the NCLUS complexity. In this study, for the selection of the bandwidth, we minimize the leave-one-out cross validated - Root Mean Square Error (RMSE). Moreover, in this automatic determination of the bandwidth, the selection is not performed directly on the bandwidth b, but on $b^{\%}$, that is, the bandwidth expressed as a percentage of the maximum dissimilarity between two connected nodes. This means that the algorithm implicitly considers different bandwidth values b at different nodes of the tree depending on the maximum dissimilarity between connected examples falling in that node of the tree. The bandwidth $b^{\%}$ ranges in the interval $[0, 100\%]$.

Minimization is performed by means of the Golden section search (Brent, 1973) that recursively partitions the $b\%$ domain. Golden section search is similar to binary search, improving it by splitting the range in two intervals with a length ratio of γ instead of 1 (equal parts). *Golden ratio* has the value $\gamma = \frac{1+\sqrt{5}}{2}$.

The share maintains a pair of minimum and maximum bandwidth values, $b_1^{\%}$ and $b_2^{\%}$ (at the first iteration, they are initialized as the minimum and maximum bandwidth in the interval $[0, 100\%]$). At each step, the algorithm identifies a point $b_3^{\%}$ between them, according to the golden ratio and computes the cross-validated error for that point ($error_{b_3^{\%}}$). ($b_1^{\%} < b_3^{\%} < b_2^{\%}$). The values of the function at these points are $f(b_1^{\%})$, $f(b_3^{\%})$, and $f(b_2^{\%})$ and, collectively, these are known as a "triplet". The algorithm than identifies the only parabola with a vertical axis that intersects the points $\{(b_1^{\%}, error_{b_1^{\%}})$, $(b_3^{\%}, error_{b_3^{\%}})$, $(b_2^{\%}, error_{b_2^{\%}})\}$. On the basis of the position of the minimum of this parabola, the system decides whether to consider ($b_1^{\%}$, $b_3^{\%}$) or ($b_3^{\%}$, $b_2^{\%}$) as the next pair of (minimum and maximum) $b^{\%}$ values.

The search stops when there is no reduction of cross-validated RMSE. In the algorithm, the RMSE is computed by fitting a weighted linear model for the example left out. Having decided to consider only the Euclidean weighting function (see Equation 2.6), we optimize $b^{\%}$ only for this case.

7.2.4 Time Complexity

The computational complexity of the algorithm depends on the computational complexity of adding a splitting node t to the tree, which in fact depends on the complexity of selecting a splitting test for t. A splitting test can be either continuous or discrete. In the former case, a threshold β has to be selected for a continuous variable. Let N be the number of examples in the training set; Then the number of distinct thresholds can be N-1 at worst. They can be determined after sorting the set of distinct values.

If m is the number of descriptive variables, the determination of all possible thresholds has a complexity $O(m*N*\log N)$, assuming an optimal algorithm is used for sorting.

For each variable, the system has to compute the evaluation measure h for all the possible thresholds. This computation has, in principle, time-complexity $O((N-1)*(N+N*k))$, where $N-1$ is the number of thresholds, k is the average number of edges for each node in the network, $O(N)$ is the complexity of the computation of the variance reduction $\widehat{\Delta_Y}(C,\mathcal{P})$ and $O(N*k)$ is the complexity of the computation of autocorrelation $A_Y(G,\eta(\cdot),\mathcal{P})$.

However, it is not necessary to recompute autocorrelation values from scratch for each threshold, since partial sums in both variance reduction computation and in autocorrelation computation can be used. In particular, partial sums can be incrementally updated depending on the examples that are iteratively moved from the right to the left branch. This optimization makes the complexity of the evaluation of the splits for each variable $O(N*k)$. This means that the worst case complexity of creating a splitting node on a continuous attribute is $O(m*(N\log N+N*k))$.

Similarly, for a discrete splitting test (for each variable), the worst case complexity is $O((d-1)*(N+N*k))$, where d is the maximum number of distinct values of a discrete variable ($d \leq N$). This complexity takes the same optimization as proposed for continuous splits into account. Therefore, finding the best splitting node (either continuous or discrete) has a complexity of $O(m*(N\log N+N*k))$+$O(m*d*(N+N*k))$, that is $O(m*N*(\log N+d*k))$.

The cost of passing all instances and calculating the needed statistics for all descriptive attributes over a single target variable is $O(d*N)$. Once the best test is selected, splitting the instances into the respective nodes costs $O(N)$. To sum up, the computational complexity of creating a node in the tree is $O(m*N*(\log N+d*k))+O(d*N)+O(N)$. Furthermore, if we take into consideration that $m=O(d)$, then the computational complexity of the process of creating a node in the tree is $O(d*N*(\log N+d*k))+O(d*N)+O(N)$.

We assume, as in (Witten and Frank, 2005), that the tree is balanced and bushy. This means that the number of nodes is in the order of $\log N$, i.e., $O(\log N)$. Having this in mind, the total computational cost of tree construction is $O(d*N*(log^2 N+d*k))+O(d*N*\log N)+O(N*\log N)$. The upper bound of this cost is determined by the first term of the overall computational complexity, i.e., $O(d*N*(log^2 N+d*k))$, where N is the number of examples, m is the descriptive variables and k is the average number of edges for each example (for each node in the network).

7.3 Empirical Evaluation

In this section, we present an empirical evaluation of the system NCLUS. After we provide a description of the used datasets and experimental settings, we proceed to presenting empirical results. The results target single and multi-target regression tasks.

The predictive performance of NCLUS is compared to competitive existing approaches. Empirical results show that our algorithm performs better than PCTs learned by completely disregarding network information, PCTs that are tailored for spatial data, but does not take into autocorrelation into account, and a variety of other existing approaches.

7.3.1 Datasets

In this experimental evaluation, we use four real network datasets obtained from social domains and six network datasets obtained from spatial data. They are described below.

Social network data

The **VideoL** dataset contains the ECML PKDD 2011 Discovery Challenge data (Antulov-Fantulin et al, 2011). The data are related to the content of VideoLectures.net, a free and open access multimedia repository of video lectures, mainly of research and educational character. The response is the total number of views of lectures published online, where pairs of lectures are viewed together (not necessarily consecutively) with at least two distinct cookie-identified browsers. The predictor variables include several properties of a lecture such as the type, category, author and language of the lecture, as well as the recorded and published dates of the lecture. Here we use the complete (training) data from the Challenge for 2009. The network structure has 754 nodes and 14398 edges. The nodes contain the lectures along with their properties, whereas the dissimilarities are the inverse of frequency (the number of distinct cookie-identified browsers) with which the respective pair of lectures was viewed together.

The **Books** dataset contains cross-rating book data from different users (Ziegler et al, 2005). For each node (book), the ISBN code, author, year of publication and publisher information are given, as well as the users' rating. The response is the average rating of all users. The network structure has 500 nodes (books) and 1167 edges. The nodes represent the books (described with their properties), whereas the weighted edges represent the dissimilarity (scalar distance) of the ratings given by the users to the respective pair of books.

The **Movies** datasets contains movie ratings given to movies by users of the online movie recommender service Movielens, collected during the period 1997-1998 [1]. Specifically, for each movie, it contains the IMDB movie identifier, genre, country, movie director and filming location, as well as all/top/audience critics' ratings: average scores, numbers of reviews/fresh scores/rotten scores from the Rotten Tomatoes film review aggregator. The response variable is the *all critics ratings*: all other ratings data are not included in the analysis. We are interested in pairs of movies that are ranked together by a single user, where the selected users had rated at least 20 movies. The network structure has 500 nodes and 202440 edges for the Movies1 dataset (*snapshot1*) and 500 nodes and 122748 edges for the Movies2 dataset (*snapshot2*). The nodes represent the movies (labeled with their properties), whereas the weighted edges represent the dissimilarity (scalar distance) of the ratings given by the users to the respective pair of movies.

The **Twitter** Maternal Health dataset contains the top Twitter users who recently mentioned maternal mortality. This dataset is obtained by a query performed on August 25, 2010 and it is sorted by betweenness centrality [2]. Specifically, it contains the number of posts (tweets) of a user (response variable), user's registration date on Twitter and its time zone, as well as the number of tweets (posted by other users) that the user marked as "favorites" and the number of "following" and "followed" on Twitter. The relationships "following" and "followed" simply reflect the number of users that are subscribed to receive news information from a specific user and the number of users that a specific user is subscribed to receive news information from. Here, we address only the pairs of users that are in the relationship "following". The network structure has 109 nodes and 198 edges. The nodes are the users (along with their properties), whereas the weighted edges are the "following" relation between the Twitter users. Note that this relationship is binary ("1" if there is a relation "following" between two Twitter users and "0" otherwise) and we consider only edges contained in this relation.

The Twitter network differs from other communities networks (e.g., Facebook, MySpace and MSN), because it is an information and a community network, where a user connects to the latest news information about what he/she finds interesting. Note that Twitter relationships are not symmetric (i.e., networks are of directed nature), thus a path from a user to another user may follow different hops for which the inverse direction does not exist. For example, while in MSN a link represents a mutual agreement of a

[1] `http://www.grouplens.org/node/12`

[2] `http://casci.umd.edu/NodeXL_Teaching/`

relationship, on Twitter a user is not obligated to reciprocate followers by following them.

Spatial network data

In the spatial datasets, the nodes are the spatial units of analysis considered (wards, sampling points, counties and forest fire sites). They are described by some attributes that will be discussed in detail below. The spatial units of analysis are at some distance apart in space.

The **NWE** (North-West England) dataset contains census data collected in the European project SPIN!. The data concerns North West England, an area that is decomposed into censual sections (wards). Census data provided by the 1998 Census are available at ward level. We consider the percentage of mortality (response variable) and measures of deprivation level in the ward according to index scores such as the Jarman Underprivileged Area Score, Townsend score, Carstairs score and the Department of the Environment Index, as well as the coordinates of the ward centroids. The nodes in the network structure are the 970 wards.

The datasets **SIGMEA_MS** and **SIGMEA_MF** (MS and MF) (Demšar et al, 2005) are derived from one multi-target dataset containing measurements of pollen dispersal (crossover) rates from two lines of plants (response variables), that is, the transgenic male-fertile (MF) and the non-transgenic male-sterile (MS) line of oilseed rape. The coordinates of each sample point are collected. The predictor variables are the cardinal direction and distance of the sample point from the center of the donor field, the visual angle between the sample plot and the donor field, and the shortest distance between the plot and the nearest edge of the donor field. The nodes in the network structures of both SIGMEA_MS and SIGMEA_MF are the 817 sampling points.

The **FOIXA** dataset (Debeljak et al, 2012) contains measurements of the rates of outcrossing at sample points located within a conventional field that comes from the surrounding genetically modified (GM) fields within a 400 *ha* large maize production area in the Foixa region in Spain. The measurements include the coordinates of the sampling points (units) and several predictor variables, that is, the number and size of the surrounding GM fields, the ratio of the size of the surrounding GM fields and the size of conventional fields and the average distance between the conventional and the GM fields. The nodes in the network structures of FOIXA represent 420 sampling points.

The **GASD** (USA Geographical Analysis Spatial Dataset) (Pace and Barry, 1997) contains 3106 observations on USA votes cast in 1980 presidential election per county. Besides the number of votes (response variable), the coordinates of the centroid of each county as well as the number of owner-occupied housing units, the aggregate income and the population over 18 years of age are reported, for the respective county. The 3106 counties represent the nodes in the network structure.

The **Forest Fires** (FF) dataset (Cortez and Morais, 2007) is publicly available for research purposes from the UCI Machine Learning Repository [1]. It collects 517 forest fire observations from the Montesinho park in Portugal. The data, collected from January 2000 to December 2003, includes the coordinates of the forest fire sites, the burned area of the forest given in *ha* (response variable), the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), the Drought Code (DC), the Initial Spread Index (ISI), the temperature in degrees Celsius, the relative humidity, the wind speed in km/h and the outside rain in mm within the Montesinho park map. The nodes in the network structure represent the individual forest fires sites.

In the networks obtained from these spatial data, edges are defined for each pair of nodes and dissimilarities are computed according to the Euclidean distance.

[1] `http://archive.ics.uci.edu/ml/`

7.3.2 Experimental setup

In the following subsection, we present the experimental setup that we use in order to evaluate the results obtained by using the proposed NCLUS algorithm. First, we describe the evaluation metrics used for the evaluation. Next, we present the state-of-the-art algorithms that we use for this comparison. A the end, we explain the statistical comparison performed in order to evaluate the empirically obtained results.

Evaluation metrics

We evaluate the performance of several variants of NCLUS and compare it to the performance of the original CLUS algorithms, as well as to the performance of several other algorithms. The evaluation is performed on the two collections of datasets described above.

The evaluation is performed in terms of several metrics, which include accuracy, model complexity, and learning time. In addition, we also measure autocorrelation of the errors of the learned models on the testing set, evaluating the ability of the different algorithms to capture the effect of autocorrelation within the learned models, as well as the correlation of the model predictions with the true target values on the test set.

All of these performance measures are estimated by using 10-fold cross validation (always on the same folds). In particular, the accuracy is measured in terms of the Root Mean Squared Error (RMSE), while the model complexity is measured in terms of the number of leaves in the learned trees. The computation time is measured in seconds. The experiments were run on an Intel Xeon CPU @2.00GHz server running the Linux Operating System.

Algorithms compared

NCLUS is run with the automatic bandwidth determination, with the three different autocorrelation measures, that is, Moran's I (NCLUS_I), Relational Autocorrelation (NCLUS_RA) and Connectivity Index (NCLUS_CI), and with $\alpha \in \{0, 0.5\}$. NCLUS, with the above experimental configurations, is compared to the original CLUS algorithm (Blockeel et al, 1998).

Only for the spatial networks, where the spatial coordinates of each node are available in the data, NCLUS is also compared to a version of CLUS algorithm, where the spatial coordinates are treated as additional response variables. This is done only for the computation of the evaluation measures. In this way, we are able to implicitly take spatial autocorrelation into account. We refer to this configuration of CLUS as CLUS*.

Moreover, the empirical evaluation includes the well-known tree-based method M5' Regression Trees (Quinlan, 1993), as well as non tree-based methods such as Support Vector Regression (SVR) (Basak et al, 2007) and the k-Nearest Neighbors (k-NN) (Aha and Kibler, 1991), which do not consider autocorrelation. The WEKA (Witten and Frank, 2005) implementation of these algorithms was used with their default settings, with the number of examples in a leaf in M5', as well as the number of neighbors in k-NN, set to $\sqrt{N}$. Furthermore, we also compare NCLUS to the Iterative Transductive Learning (ITL) algorithm (Appice et al, 2009) that addresses the network regression problem for spatial data. ITL works in the transductive learning setting and considers autocorrelation. Finally, as a baseline (Base) we also use the default method that always predicts the mean (over the training set) of the response variable.

Statistical comparison

In order to compare the predictive capabilities of the learned models, we use the non-parametric Wilcoxon two-sample paired signed rank test (Orkin and Drogin, 1990). To perform the test, we assume that the experimental results of the two methods compared are independent pairs

Table 7.1: *Comparison of the errors made on social network data.* The RMSEs (estimated by 10-fold CV) of the models obtained by different learning approaches: NCLUS_I, NCLUS_RA, NCLUS_CI, CLUS, SVR, *k*-NN and M5', as well as the default Base model. For each network dataset, the best results are highlighted in bold.

Method/Network dataset		VideoL	MOVIES1	MOVIES2	BOOKS	TWITTER
NCLUS_I	$\alpha = 0.0$	686.32	1.08	1.17	**2.53**	**10170.92**
NCLUS_I	$\alpha = 0.5$	653.2	**1.06**	**1.02**	**2.53**	**10170.92**
NCLUS_RA	$\alpha = 0.0$	686.32	1.08	1.17	**2.53**	**10170.92**
NCLUS_RA	$\alpha = 0.5$	653.2	**1.06**	**1.02**	**2.53**	**10170.92**
NCLUS_CI	$\alpha = 0.0$	686.32	1.62	2.11	**2.53**	**10170.92**
NCLUS_CI	$\alpha = 0.5$	653.2	1.51	1.31	**2.53**	10712.23
CLUS		660.69	1.53	2.42	2.83	10641.03
SVR		721.43	1.77	2.52	2.65	13875.73
k-NN		937.68	1.70	2.58	2.65	11007.23
M5'		**574.17**	2.09	2.2	2.67	12253.34
Base		722.39	2.10	2.70	**2.53**	13255.27

$\{(q_1,r_1),(q_2,r_2),\ldots,(q_n,r_n)\}$ of sample data. We then rank the absolute value of the differences $q_i - r_i$. The Wilcoxon test statistics WT^+ and WT^- are the sum of the ranks from the positive and negative differences, respectively. We test the null hypothesis H_0: "no difference in distributions" against the two-sided alternative H_1: "there is a difference in distributions". Intuitively, when $WT^+ \gg WT^-$ and vice versa, H_0 is rejected. Whether WT^+ should be considered "much greater than" WT^- depends on the considered significance level. The null hypothesis of the statistical test is that the two populations have the same continuous distribution. Since, in our experiments, q_i and r_i are average MRSEs, $WT^+ \gg WT^-$ implies that the second method (R) is better than the first (Q). In all experiments reported in this empirical study, the significance level used in the test is set at 0.05.

7.3.3 Results and Discussion

In this subsection, we present the results from the empirical evaluation of the NCLUS algorithm. The evaluation is performed using the datasets described in Section 7.3.1 and the experimental setup described in Section 7.3.2.

The experiment presented in this chapter are divided in two parts according to the type of the data that they refer to. The first part are experiments performed on social network data, whereas the second part are experiments performed on networks that come from spatial data.

We discuss the evaluation results in terms of their accuracy, as well as in terms of the properties of the predictive models, by analyzing the models sizes, the autocorrelation of the errors of the predictive models and their learning times. Moreover, we compare the results obtained using single and multi-target predictive models and discuss their properties.

Social network data

We present the results of the predictive models obtained by the NCLUS algorithm when evaluating social network data. Table 7.1 reports the average errors of the PCTs by NCLUS_I, NCLUS_RA, NCLUS_CI and CLUS, as well as the errors of the SVR, *k*-NN and M5' approaches, on the social network data. The last row in Table 7.1 gives the errors of the Base model that always predicts the mean. For each network dataset, the best results are highlighted in bold.

We can observe that the NCLUS error depends on the network autocorrelation measure and on the relative importance given to the variance reduction and autocorrelation when building the PCT (according to the α parameter). The use of different autocorrelation measures does not change the obtained results much. However, the best results, in line with considerations reported in Section 7.2.2, are obtained by using the Global Moran's I and the Relational Autocorrelation. In contrast, the α parameter significantly

Table 7.2: *Wilcoxon tests on social network data.* The p-values of the Wilcoxon test comparing different NCLUS variants and CLUS on social network data, at the 0.05 significance level. (+) means that NCLUS is better than CLUS, ($-$) means that CLUS is better than NCLUS, (=) means that both algorithms perform equally well.

Network /Method	NCLUS_I		NCLUS_RA		NCLUS_CI	
dataset	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$
VideoL	($-$)0.33	(+)0.28	($-$)0.33	(+)0.28	($-$)0.33	(+)0.28
MOVIES1	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**	($-$)0.33	(+)0.58
MOVIES2	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**
BOOKS	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**	(+)**0.01**
TWITTER	(+)0.96	(+)0.96	(+)0.96	(+)0.96	(+)0.96	($-$)0.58

affects the errors. The best results are always obtained with $\alpha = 0.5$ (when $\alpha = 0$ NCLUS uses only autocorrelation, when $\alpha = 0.5$ NCLUS equally weights variance reduction and autocorrelation, and when $\alpha = 1$ NCLUS works such as the original CLUS algorithm).

The results of NCLUS_I and NCLUS_RA are very similar and reflect the covariance between pairs of observations. The only difference between the two measures is in the existence of weights in the definition of the Global Moran's I. The weights reflect the strength of the relations between the nodes in the network and are associated to the edges in the network. Thus, the difference in the two measures of autocorrelation is in the explicit (Euclidean similarity measure vs. binary) strength of the relation between the nodes in the network, which usually comes with the definition of the network. On the other hand, NCLUS_CI performs worse than NCLUS_I and NCLUS_RA. From the results presented in Table 7.1, we can see that NCLUS compares very well to mainstream methods that do not consider autocorrelation (SVR, *k-NN*, M5', Base), by providing a remarkable reduction of the error for the most of the network datasets.

Table 7.2 presents the p-values of the Wilcoxon test comparing the errors obtained by NCLUS_I, NCLUS_RA and NCLUS_CI with the errors of the original CLUS algorithm. The analysis of these results reveals that NCLUS (and in particular NCLUS_I and NCLUS_RA) gives statistically better results than CLUS for three out of the five datasets (Movies1, Movies2 and Books). For the other two datasets NCLUS results are better than those obtained by CLUS, but not significantly. This empirical study confirms our hypothesis that the explicit consideration of the non-stationary autocorrelation when building regression models from network data can increase the accuracy (decrease the error) of the obtained PCTs when autocorrelation is present in the data. Section 7.3.3 discusses some further characteristics of the learned PCTs.

Network data obtained from spatial datasets

We present the results of the predictive models obtained by the NCLUS algorithm when evaluating network data obtained from spatial datasets. Table 7.3 reports the average errors achieved by NCLUS, CLUS, CLUS*, ITL, SVR, k-NN and M5' on spatial datasets, as well as the errors of the Base model that always predicts the mean, on the spatial network datasets. Note that ITL builds model trees that consider spatial autocorrelation in a transductive network setting [1]. The best results are highlighted in bold.

As for the social network datasets, NCLUS errors only slightly depend on the adapted network autocorrelation measure. Indeed, the observed errors of PCTs learned by NCLUS do not change too much by varying this measure. However, once again, the best results are obtained with Moran's I.

[1]Strictly speaking a comparison to the latter approach is not fair since model trees are recognized to be more accurate than regression trees. Moreover, ITL, according to the transductive learning settings, exploits both training and testing data during learning. We primarily use these results as a motivation for our further work and present the ITL improvements over the other algorithms in italic.

Table 7.3: *Comparison of the errors made by on spatial network data.* The RMSEs (estimated by 10-fold CV) of the models obtained by different learning approaches: NCLUS_I, NCLUS_RA, NCLUS_CI, CLUS, SVR, *k*-NN and M5', as well as the default Base model. For each network dataset, the best results are highlighted in bold. Results for NWE are multiplied by 10^3.

Method /Spatial Dataset		FF	NWE	FOIXA	GASD	MS	MF
NCLUS_I	$\alpha = 0.0$	**42.82**	**2.16**	2.53	0.18	2.47	5.44
NCLUS_I	$\alpha = 0.5$	56.55	2.48	**2.34**	0.17	**2.29**	5.81
NCLUS_RA	$\alpha = 0.0$	**42.82**	2.45	2.65	0.18	2.47	6.60
NCLUS_RA	$\alpha = 0.5$	53.27	2.46	2.66	0.17	2.35	5.92
NCLUS_CI	$\alpha = 0.0$	**42.82**	2.47	2.61	0.17	2.49	6.72
NCLUS_CI	$\alpha = 0.5$	52.79	2.45	2.66	0.16	2.35	5.93
CLUS		49.21	2.46	2.65	0.16	2.35	5.64
CLUS*		47.22	2.47	2.52	0.16	2.54	6.68
ITL		58.25	2.54	3.55	*0.14*	*1.92*	*3.52*
SVR		64.58	2.50	2.95	**0.14**	2.80	8.60
kNN		65.44	2.40	2.73	0.16	2.37	**4.56**
M5'		47.22	2.47	2.66	0.16	2.47	5.92
Base		63.66	2.50	2.93	0.20	3.23	8.58

This result is not surprising, as Moran's I was specifically designed for modeling autocorrelation in spatial domains. Unlike for social network data, we have no clear evidence of the best accuracy been achieved with $\alpha = 0.5$. In general, considering autocorrelation is beneficial, but we cannot decide apriori how much the consideration of autocorrelation should influence the PCT construction (i.e., the value of $alpha$). In general, there is always a configuration of NCLUS that outperforms CLUS. Moreover, NCLUS outperforms CLUS*, except for the GASD dataset where they are comparable in performance.

Both, NCLUS and CLUS* are designed to improve (if possible) the accuracy of the CLUS PCTs by modifying/enhancing the heuristic (variance reduction for regression) used to evaluate each split in the process of tree construction. Whereas NCLUS accounts for autocorrelation that is often present in network data, CLUS* takes the spatial coordinates (usually presented in pairs (x, y) or (latitude, longitude)) from spatial datasets and considers them as response variables in addition to the actual response(s). This means that CLUS* aims at generating PCTs that will maximize the inter-cluster variance reduction of both the responses and the coordinates. Moreover, much higher importance is given to the spatial information than to the actual response, as they all are normalized at the beginning of the modeling process and equal importance is given the single target (response) and two coordinates x and y (as additional targets). This makes the predictions of the models more coherent in space than those of the CLUS models, mostly increases the accuracy of the models and shows some other characteristics of the models that will be discussed in Section 7.3.3.

However, in contrast to NCLUS models, CLUS* models cannot deal with non-stationary autocorrelation appropriately. In NCLUS, two different geographical regions that have the same distribution of attribute and target values including autocorrelation, can be covered by one leaf of the tree. In CLUS*, the data will need to be split into different regions due to the strong preference for spatial homogeneity. Moreover, CLUS* cannot handle different definitions of the regression problem that can arise from different definitions of the network, e.g., using different similarity measures. As for the social network datasets, NCLUS compares very well to mainstream methods that do not consider autocorrelation (SVR, *k-NN*, M5', Base), by providing a remarkable reduction of the error in most of spatial network datasets.

Table 7.4 presents the results of the Wilcoxon test when comparing NCLUS with the original CLUS algorithm in terms of the RMSE of the obtained PCTs. The errors obtained with NCLUS are statistically lower than those obtained with CLUS for the FF (using Global Moran's I/Relational Autocorrelation and $\alpha = 0$) and the FOIXA (using Global Moran's I) datasets and worse for the GASD dataset (using Global Moran's I/Relational Autocorrelation and $\alpha = 0$). In the other cases, there is no statistical difference, at the significance level of 0.05.

Table 7.4: *Wilcoxon tests on spatial network data.* Statistical comparison of the performance of different NCLUS variants and CLUS on spatial network data. The p-values of the Wilcoxon test comparing NCLUS and CLUS, at the 0.05 significance level. (+) means that NCLUS is better than CLUS, (−) means that CLUS is better than NCLUS, (=) means that both algorithms perform equally well.

Spatial /Method	NCLUS_I		NCLUS_RA		NCLUS_CI	
dataset	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$
FF	(+)**0.01**	(−)0.56	(+)**0.01**	(−)0.56	(+)**0.01**	(−)0.56
NWE	(+)0.20	(−)0.96	(+)0.80	(=)1.00	(−)0.88	(+)0.80
FOIXA	(+)**0.01**	(+)**0.01**	(+)0.88	(+)0.88	(+)0.72	(−)0.80
GASD	(−)**0.01**	(−)0.28	(−)**0.01**	(−)0.28	(−)0.28	(=)1.00
MF	(−)0.88	(+)0.58	(−)0.88	(+)0.96	(−)0.88	(+)0.96
MS	(+)0.58	(−)0.88	(−)0.20	(−)0.58	(−)0.22	(−)0.58

Properties of the models

We present some properties of the models obtained by the SCLUS algorithm, for single and multi-target classification and regression tasks. Specifically, we present the model sizes, the autocorrelation of the errors of the predictive models and their learning times.

For completeness, we also include Table 7.5 that gives the mean and variance of the target variable calculated on the entire dataset and the correlation of the NCLUS_I, NCLUS_RA, NCLUS_CI, CLUS, CLUS*, SVR, k-NN and M5' predictions with the true target values on the test set. In most cases, NCLUS returns predictions which are more correlated with the true values than other competitive approaches. For the Twitter dataset, NCLUS models result in negative correlation values due to over-pruning. Note that this is the smallest of all datasets considered.

Table 7.6 shows the average size of the PCTs (number of leaves) learned with NCLUS, CLUS and CLUS*. In most cases, NCLUS learns smaller trees, whereas CLUS* learns larger trees than CLUS. This comes as a result of the consideration of the autocorrelation phenomenon in NCLUS models which, in most cases, makes the learned PCTs not only more accurate, but also smaller and consequently simpler to be displayed and interpreted. The predictions of the CLUS* PCTs are more coherent in space in comparison to CLUS PCTS, but differently from NCLUS, this happens at the price of increasing the size of the trees. While NCLUS can consider two different geographical regions that have the same distribution of attribute and target values (including autocorrelation) in one leaf of the tree, CLUS* will split these due to the emphasis on spatial homogeneity. This is the reason for the increase of the tree size.

Moreover, the PCTs learned with NCLUS by considering only the measures of network autocorrelation in the process of tree construction ($\alpha = 0$) are smaller than the models obtained with NCLUS using both autocorrelation and variance reduction in the tree construction ($\alpha = 0.5$). This comes as a result of the reduction in the number of relations/connections in the network with the introduction of additional splitting nodes, which then directly affects the calculation of the measures of network autocorrelation. This kind of situation is most notable in the models obtained using $NCLUS\_I$ and $NCLUS\_RA$, where the network autocorrelation is the only splitting criterion for the tree construction process. On the other hand, models that use only the *CI* index as a splitting criterion are less affected by this phenomenon as CI is only a function of the degree $Deg(v_i)$ of a node v.

In Table 7.7, we present autocorrelation of the prediction errors of the PCTs learned with NCLUS, CLUS, CLUS*, SVR, k-NN and M5'. Autocorrelation is computed by means of the Moran's I on the errors committed on the testing set. We analyze the obtained models in terms of this measure in order to show that PCTs learned with NCLUS can capture autocorrelation, when present in the network, and generate predictions that exhibit small (absolute) autocorrelation in the errors. The analysis of the results reveals that NCLUS handles autocorrelation better than CLUS. In fact, coherently with the analysis reported in Section 7.2.2, NCLUS is able to correctly remove the effect of autocorrelation when

Table 7.5: *Main characteristics of the target variable.* Mean and variance of the target variable calculated on the entire dataset and the correlation of the NCLUS_I, NCLUS_RA, NCLUS_CI, CLUS, CLUS*, SVR, *k*-NN and M5' model predictions with the true target values on the test set.

Dataset	Mean	Variance	NCLUS_I		NCLUS_RA		NCLUS_CI		CLUS	CLUS*	SVR	*k*-NN	M5'
			α										
			0.0	0.5	0.0	0.5	0.0	0.5					
VideoL	359.20	$522*10^3$	0.34	0.42	0.34	0.42	0.34	0.42	0.35	-	0.29	0.52	0.44
MOVIES1	1993.63	53.23	0.85	0.86	0.85	0.86	0.85	0.85	0.59	-	0.56	0.27	0.52
MOVIES2	1985.12	385.77	0.88	0.91	0.88	0.91	0.87	0.83	0.67	-	0.46	0.53	0.34
BOOKS	5.66	6.40	0.12	0.12	0.12	0.12	0.12	0.12	0.12	-	0.12	0.00	0.12
TWITTER	5987.29	$173*10^6$	-0.32	-0.32	-0.32	-0.32	-0.32	0.28	-0.32	-	0.10	0.27	0.33
FF	12.95	4052.09	0.46	0.44	0.46	0.44	0.46	0.44	0.44	0.45	0.00	0.02	-0.12
NWE	10.00	9.00	0.14	0.08	0.10	0.10	0.09	0.10	0.08	0.06	0.10	0.02	0.11
FOIXA	0.93	8.58	0.34	0.34	0.33	0.33	0.33	0.33	0.26	0.34	0.28	0.37	0.30
GASD	9.76	1.74	0.46	0.57	0.46	0.57	0.57	0.59	0.59	0.59	0.70	0.68	0.58
MF	0.52	10.43	0.73	0.88	0.73	0.88	0.73	0.88	0.73	0.36	0.70	0.83	0.83
MS	0.05	3.24	0.87	0.81	0.83	0.80	0.80	0.81	0.70	0.45	0.77	0.79	0.77

Table 7.6: *The average size of the PCTs learned with NCLUS, CLUS, CLUS* and M5'.*

Dataset /Method	NCLUS_I		NCLUS_RA		NCLUS_CI		CLUS	CLUS*	M5'
	α								
	0.0	0.5	0.0	0.5	0.0	0.5			
VideoL	3.0	6.9	3.0	4.9	3	6.9	6.9	-	11.0
MOVIES1	11.1	13.2	11.1	13.2	12.9	13.5	7.7	-	19.0
MOVIES2	7.9	10.8	7.9	10.8	7.9	11.9	7.5	-	11.0
BOOKS	1.0	1.0	1.0	1.0	1.0	1.0	4.8	-	9.0
TWITTER	1.7	1.7	1.7	1.7	1.0	1.7	2.2	-	4.0
FF	1.0	1.4	1.0	1.8	1.1	1.8	1.8	1.0	1.0
NWE	1.4	3.6	2.0	3.8	1.2	3.9	5.6	2.3	4.0
FOIXA	1.8	2.3	1.0	3.8	2.1	4.0	4.7	6.1	3.0
GASD	8.4	31.3	8.4	30.7	23.1	30.6	27.7	23.8	49.0
MF	1.0	4.4	1.0	4.2	3.4	4.1	5.1	19.7	6.0
MS	1.5	6.0	1.0	5.1	2.9	5.9	6.0	19.2	6.0

making predictions. Thus, it is able to obtain network-consistent predictions. This analysis also reveals that CLUS* is able to capture autocorrelation better than CLUS, but worse than NCLUS. This is expected according to the differences between NCLUS and CLUS*, already been discussed in this section. Moreover, as expected, autocorrelation on the errors is lower when $\alpha = 0$.

Table 7.8 reports the average learning times for NCLUS, CLUS, CLUS*, ITL, SVR, *k*-NN and M5' models. Results for CLUS* and ITL are available only for the spatial datasets. The shortest learning times are obtained by using the CLUS algorithm. The learning times for CLUS* are similar (slightly larger) to the times of CLUS, as in this configuration CLUS is run by considering the spatial coordinates as responses, while the time complexity of the PCT induction remains the same. The learning times for NCLUS are much longer than the learning times for CLUS because the consideration of autocorrelation introduces additional computations and increases the complexity of building a PCT. This is coherent with the time complexity analysis reported in Section 7.2.4. The learning times for ITL are significantly longer than the times of CLUS, CLUS* and NCLUS because of its iterative co-training implementation.

NCLUS for Multi-Target Regression

In this section, we investigate the capability of NCLUS to adequately combine the effects of autocorrelation over several response variables when solving multi-target regression tasks.

The results, presented in Table 7.9, demonstrate that there is no statistically significant difference between the accuracy of the multi-target PCTs and the accuracy of the corresponding single-target PCT. This behavior is observed independently on the autocorrelation measure (NCLUS_I, NCLUS_RA and NCLUS_CI) used to learn the PCTs. In any case, we observe that the size of a single Multi-Target

Table 7.7: *Autocorrelation of the prediction errors on network data.* Average autocorrelation of the prediction errors on the testing sets, made by PCTs learned with NCLUS, CLUS, CLUS*, SVR, *k*-NN and M5'. For each dataset, the best results (the smallest in absolute value) are given in bold.

Dataset /Method	NCLUS_I		NCLUS_RA		NCLUS_CI		CLUS	CLUS*	SVR	*k*-NN	M5'
	α										
	0.0	0.5	0.0	0.5	0.0	0.5					
VideoL	0.30	**0.22**	0.35	**0.22**	0.35	**0.22**	1.00	-	**0.22**	**0.22**	**0.22**
MOVIES1	-0.02	-0.02	-0.02	-0.02	**-0.01**	-0.02	-0.02	-	-0.02	-0.02	-0.02
MOVIES2	**-0.01**	-0.02	**-0.01**	**0.01**	**0.01**	-0.02	-0.02	-	**-0.01**	**-0.01**	**-0.01**
BOOKS	0.04	0.04	0.04	0.04	0.04	0.04	0.04	-	0.04	0.04	0.04
TWITTER	-0.50	0.50	-0.50	0.50	-0.50	**0.35**	0.76	-	-0.50	0.50	0.50
FF	**-0.02**	**-0.02**	**-0.02**	**-0.02**	**-0.02**	**-0.02**	1.00	**-0.02**	**-0.02**	**-0.02**	0.98
NWE	**0.00**	-0.01	-0.03	-0.02	-0.03	-0.02	0.84	-0.01	-0.01	-0.02	-0.01
FOIXA	**-0.02**	**-0.02**	**-0.02**	**-0.02**	**-0.02**	**-0.02**	0.96	**-0.02**	-0.03	-0.03	-0.06
GASD	0.19	0.19	0.11	0.19	0.07	0.05	1.00	0.08	**0.01**	0.03	0.37
MF	**-0.01**	0.15	**0.01**	0.07	0.08	0.06	0.88	0.15	**-0.01**	**0.01**	0.14
MS	0.13	0.24	0.03	0.055	**0.01**	0.04	0.66	0.13	**-0.01**	**-0.01**	0.34

Table 7.8: *Learning times of the obtained models.* The models are obtained with NCLUS_I, CLUS, CLUS*, ITL, SVR, *k*-NN and M5' and the times are given in seconds.

Dataset /Method	NCLUS_I		CLUS	CLUS*	ITL	SVR	*k*-NN	M5'
	$\alpha = 0$	$\alpha = 0.5$						
VideoL	8.66	7.71	0.04	-	-	0.03	0.03	0.07
MOVIES1	130.21	141.17	0.07	-	-	0.04	0.04	1.02
MOVIES2	45.79	48.89	0.07	-	-	0.04	0.08	0.09
BOOKS	0.07	0.09	0.09	-	-	0.02	0.04	0.06
TWITTER	0.23	0.97	0.01	-	-	0.05	0.01	0.01
FF	1.50	1.03	0.04	0.04	422.10	1.03	0.45	1.23
NWE	2.31	1.62	0.11	0.11	857.30	1.22	1.02	1.94
FOIXA	0.69	0.49	0.02	0.02	162.90	0.49	0.56	0.88
GASD	27.86	20.79	0.04	0.04	30462.50	30.45	20.54	20.43
MF	2.68	1.39	0.04	0.03	593.60	1.00	2.00	2.25
MS	3.39	1.41	0.04	0.03	528.20	0.59	2.04	3.55

PCT is always significantly lower than the combined (by sum) sizes of the two single-target trees (see Table 7.10). This means that the multi-target PCTs learned with NCLUS adequately combine the effect of autocorrelation on several response variables by resulting in a predictive model that is accurate enough and simpler to be interpreted than several distinct trees. Moreover, the learning time spent to construct a multi-target PCT is lower than the learning times spent to learn several distinct single-target PCTs (see Table 7.11).

Table 7.9: *Wilcoxon tests of the Single and Multi-Target PCTs.* The p-values of the Wilcoxon tests comparing the average RMSE of the Single and Multi-Target PCTs learned by NCLUS. $(-)$ means that Single-Target PCTs are more accurate than Multi-Target PCTs; $(+)$ means that Multi-Target PCTs are more accurate than Single-Target PCTs; (=) means that they perform equally well.

Dataset /Method	NCLUS_I		NCLUS_RA		NCLUS_CI	
	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$
MF	(=)1.00	$(-)$0.24	$(-)$0.96	$(-)$0.88	$(+)$0.51	$(-)$0.80
MS	$(-)$0.11	$(-)$0.51	$(-)$0.88	$(-)$0.65	$(+)$0.39	$(-)$0.33

Table 7.10: *Average size of the Single and Multi-Target PCTs.* The models are learned by NCLUS.

Dataset /Method	NCLUS_I		NCLUS_RA		NCLUS_CI	
	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$
MS-MF	1.0	6.4	1.5	6.3	4.8	4.8
MF	1.0	4.2	1.0	4.2	3.4	4.1
MS	1.4	6.1	1.0	5.1	2.9	5.9

Table 7.11: *Learning times for Single and Multi-Target PCTs.* The times are given in seconds and the models are learned by NCLUS_I.

Dataset /Method	NCLUS_I	
	$\alpha = 0$	$\alpha = 0.5$
MS-MF	4.73	2.7
MF	2.68	1.39
MS	3.39	1.41

7.4 Summary

In this chapter, we address the task of network regression. This is an important task as demonstrated by the presented related work and the use of real-world datasets. While many approaches for network classification exist, there are very few approaches to the network regression task.

The network setting that we address uses both the descriptive information (node attributes) and the network structure during training and uses only the descriptive information in the testing phase, where the network structure around the (new) testing instances may be unknown. This is quite different from existing approaches to network classification and regression, where the descriptive information used during both training and testing phase is typically closely related to the network structure and the connections between the nodes.

In this setting, we develop a data mining method that explicitly considers autocorrelation when building regression models from network data. The resulting models adapt to local properties of the data, providing, at the same time, smoothed predictions. The novelty of our approach is that, due to the generality of PCTs, it can work for different predictive modeling tasks, including regression and Multi-Target regression, as well as some clustering tasks.

We use well-known measures of (spatial and relational) autocorrelation, since we deal with a range of different data networks. The heuristic we use in the construction of PCTs is a weighted combination of variance reduction (related to predictive performance) and autocorrelation of the response variable(s). Our approach can consider different sizes of neighborhoods (bandwidth) and different weighting schemes (degrees of smoothing) when calculating autocorrelation. We identify suitable combinations of autocorrelation metrics and weighting schemes and automatically determine the appropriate bandwidth.

We evaluate our approach on an extensive set of real-world problems of network regression, coming from the areas of spatial and social networks. Empirical results show that the proposed extension of PCTs (NCLUS) performs better than both the original PCTs, which completely disregard network information, and PCTs that capture local spatial network regularities (CLUS*), but do not take autocorrelation into account and compares very well to mainstream methods that do not consider autocorrelation (SVR, *k-NN* and M5').

Our approach performs better along several dimensions. First, it is better in terms of predictive performance (as measured by RMSE, estimated by cross-validation). Second, autocorrelation of the errors made by our approach is smaller. Finally, the models produced by our approach are (on average) smaller.

Several directions for further work remain to be explored. The automated determination of the parameter α that sets the relative importance of variance reduction and autocorrelation during tree construction deserves immediate attention. In a similar fashion, one might consider selecting an appropriate autocorrelation measure. Moreover, it would be interesting to define novel autocorrelation measures that take indirect connections into account when computing autocorrelation. Finally, while our approach completely ignores the network information in the testing phase, we would like to explore developments in the direction of using this information if and when available.

8 Learning PCTs for HMC from Network Data

In this chapter, we propose an extension of the predictive clustering framework for predicting structured targets, i.e., for the case when the output is a structured object. The extensions that we have proposed to deal with the typical classification and regression task were described in Chapters 6 and 7, where we deal with a similar extension for the Hierarchial Multi-Label Classification (HMC) task.

The system that we propose is denoted by **NHMC**, which stands for Network Hierarchical Multi-label Classification. It is implemented in the CLUS framework. It learns models for predicting structured outputs where relations between the classes exist, i.e., the classes are organized into a tree-shaped hierarchy or a directed acyclic graph (DAG). Besides relationships among classes, the algorithm takes into account relationships between the examples, described by a network.

First, we motivate the research presented in this chapter, which is a combination of hierarchical multi-label classification and network inference. Next, we describe the proposed extension in detail, focusing on the algorithm and its time complexity. Then, we empirically evaluate it on a chosen functional genomics problem. In particular, we learn to predict the protein functional classes from a proposed approach on a hierarchy or tree-shaped or DAG of annotations, taking into account protein-to-protein interaction (PPI) networks. Finally, we compare the predictive performance of the NHMC algorithm to the performance of already existing methods, on a the task of predicting gene function in yeast, using different yeast data and different yeast PPI networks.

8.1 Motivation

In the era of high-throughput computational biology, discovering the biological functions of the genes/proteins within an organism is a central goal. Several studies have applied machine learning to infer functional properties of proteins, or directly predict one or more functions for unknown proteins (Clare and King, 2003; Qi and Noble, 2011). Indeed, the prediction of multiple biological functions with a single model, by using learning methods which exploit multi-label prediction, has made considerable progress (Barutcuoglu et al, 2006) in recent years.

A step forward is represented by models considering possible structural relationships among functional class definitions (Jiang et al, 2008; Vens et al, 2008). This is motivated by the presence of ontologies and catalogs such as Gene Ontology (GO) (Ashburner et al, 2000) and MIPS-FUN (FUN henceforth) (Mewes et al, 1999) which are organized hierarchically (and, possibly, in the form of Directed Acyclic Graphs (DAGs), where classes may have multiple parents), where general functions include other more specific functions (see Fig. 5.4). In this context, the hierarchial constraint must be considered: A gene annotated with a function must be annotated with all the ancestor functions from the hierarchy. In order to tackle this problem, hierarchical multi-label classifiers, that are able to take the hierarchical organization of the classes into account, during both the learning and the prediction phase, have been recently used (Barutcuoglu et al, 2006).

Increasing attention in recent years has been attracted by the topic of considering protein-protein interaction (PPI) networks in the identification and prediction of protein functions. This stream of research is mainly motivated by the consideration that "when two proteins are found to interact in a high throughput assay, we also tend to use this as evidence of functional linkage"(Jiang et al, 2008). As a confirmation, numerous studies have demonstrated that proteins sharing similar functional annotations tend to interact more frequently than proteins which do not share them (*guilt-by-association* principle). Interactions reflect the relation or dependence between proteins. In their context, gene functions show some form of *autocorrelation*.

Protein-protein interactions occur when two or more proteins bind together, often in order to carry out their biological function. Many of the most important molecular processes in the cell, such as DNA replication, are carried out by large molecular machines that are built from a large number of protein components organized by protein-protein interactions. Protein interactions have been studied from the perspectives of biochemistry, quantum chemistry, molecular dynamics, chemical biology, signal transduction and other metabolic or genetic/epigenetic networks. Indeed, protein-protein interactions are at the core of the entire interactomics system of any living cell. Interactions between proteins are important for the majority of biological functions. For example, signals from the exterior of a cell are mediated to the inside of that cell by protein-protein interactions of the signaling molecules.

The use of such relationships among proteins introduces the autocorrelation phenomenon into the problem of gene function prediction and violates the assumption that instances are independently and identically distributed (i.i.d.), adopted by most machine learning algorithms. Recall that while *correlation* denotes any statistical relationship between different variables (properties) of the same objects (in a collection of independently selected objects), *autocorrelation* (Cressie, 1993) denotes the statistical relationships between the same variable (e.g., protein function) on different but related (dependent) objects (e.g., interacting proteins). As described in the introductory chapters of this thesis, autocorrelation has been mainly studied in the context of regression analysis of temporal (Mitsa, 2010) and spatial (LeSage and Pace, 2001) data.

Although autocorrelation has not yet been studied in the context of Hierarchical Multi-label Classification (HMC), it is not a new phenomenon in protein studies. For example, it has been used for predicting protein properties using sequence-derived structural and physicochemical features of protein sequences (Horne, 1988). In this work, we propose a definition of autocorrelation for the case of HMC and propose a method that considers such autocorrelation in gene function prediction.

The consideration of PPI network data for HMC has received limited attention so far. One of the works that faces this problem is presented by Jiang et al (2008), where a probabilistic approach that classifies a protein on the basis of the conditional probability that a protein belongs to the child function class is presented, assuming that it belongs to the parent function class. The computation of this conditional probability takes the PPI network into account in terms of the number of the neighbors of a node. Although similar to the approach we propose latter in this chapter, this approach has problems with sparse networks and does not properly exploit the autocorrelation phenomenon. Second, in order to deal with DAGs, it transforms a DAG into a hierarchy by removing hierarchical relationships present at lower levels, thus ignoring possibly useful information. Third, this work reports experiments only for GO annotations.

The work presented in this chapter is intended to be a further step toward the investigation of methods which originate from the intersection of these two promising research areas, namely hierarchical multi-label classification and learning in presence of autocorrelation. In particular, we propose an algorithm for hierarchical multi-label prediction of protein functional classes. The algorithm exploits the non-stationary autocorrelation phenomenon which comes from protein-protein interaction by means of tree-based prediction models. In this way, we are able to:

- improve the predictive capabilities of learned models
- obtain predictions consistent with the network structure
- consistently combine two sources of information (hierarchical collections of functional class defi-

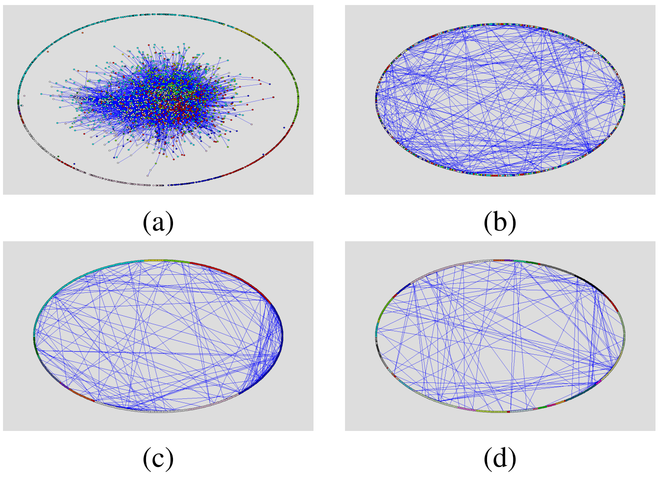


Figure 8.1: *An example of DIP Yeast network.* Different colors correspond to different classes of the FUN hierarchy. (a) Examples that are not connected are arranged along the ellipse's border; (b) Examples are arranged along the ellipse's border to show the density of the PPI interactions; (c) Examples are arranged along the ellipse's border and grouped according to the first level of the FUN hierarchy (not considering the root); d) Examples are arranged along the ellipse's border and grouped according to the second level of FUN. The networks are drawn using the Pajek software by Batagelj and Mrvar (1998).

nitions and PPI networks)

- capture the non-stationary effect of autocorrelation at different levels of the hierarchy
- also work with DAG (directed acyclic graph) structures, where a class may have multiple parents.

We first introduce new measures of autocorrelation for HMC tasks. We then describe the learning of PCTs for the HMC task in more detail. We focus on the proposed algorithm and its computational complexity.

8.2 Measures of autocorrelation in a HMC setting

We also consider autocorrelation in the context of hierarchial multi-label classification (HMC) setting. We first illustrate the notion of network autocorrelation for HMC on the problem of gene function prediction ion the context of PPI networks. In the following subsections, we introduce two new measures of network autocorrelation within the HMC setting. To the best of our knowledge, this has not been done before.

8.2.1 Network Autocorrelation for HMC

Network autocorrelation for HMC is a special case of network autocorrelation (Doreian, 1990). It can be defined as the statistical relationship between observations of a variable that takes values in the form of a hierarchy (e.g., protein function) on distinct but related (connected) nodes in a network (e.g., interacting proteins). In HMC, the domain values of the target variable form a hierarchy, such as the GO hierarchy for protein functions. Therefore, it is possible to define network autocorrelation at various levels of the hierarchy.

In predictive modeling, network autocorrelation can be a problem, since the i.i.d. assumption is violated, but also an opportunity, if it is properly considered here. This is particularly true for the task of hierarchical multi-label classification considered in this work. Indeed, due to non-stationary autocorrelation, network data can provide useful (and diversified) information for each single class at each level of the hierarchy. This information may concern gene protein functions at different levels of the hierarchy.

To better explain this concept, we refer to Figure 8.1 which shows the DIP Yeast network. Figure 8.1 (a) represents the network so that the examples that are not connected are randomly arranged along the ellipse border. To show the density of the PPI interactions, Figure 8.1 (b) represents the same network so that all examples are arranged along the ellipse border. Figure 8.1 (c) and Figure 8.1 (d) provide us a different view of Figure 8.1 (b), where examples are grouped according to the first (Figure 8.1 (c)) and second level (Figure 8.1 (d)) of the FUN hierarchy. Keeping in mind that all these graphs represent the same number of edges, from the comparison of Figure 8.1 (b) and Figure 8.1 (c) we can see that the edges "move" from the center of the ellipse towards the border. This is clearly due to autocorrelation, since the number of interactions between genes of the same class at the same level of the hierarchy is much larger than the number of interactions between genes of different classes. Moreover, by comparing Figure 8.1 (c) and Figure 8.1 (d) we notice that the autocorrelation effect is more localized in the second level of the hierarchy then in the first. Indeed, in Figure 2.5 (d), we observe a reduction of the density of edges in the center of the ellipse (most of the edges overlap with (are hidden by) the examples arranged along the ellipse border).

8.2.2 Global Moran's I

In order to consider network autocorrelation within the HMC setting, we first adapt the Global Moran's I measure of spatial autocorrelation (Legendre, 1993) to fit the general structure of network data and satisfy the hierarchial constraint. The new adapted Moran's I for HMC is created by replacing the residuals in the standard Global Moran's I formula with the distance function $d(\cdot,\cdot)$ (Vens et al, 2008) which is used for HMC tasks within the PCTs framework.

The distance function $d(\cdot,\cdot)$ is a weighted Euclidean distance, where labels at the higher levels of the hierarchy have larger weights. In the context of HMC, this means that the similarity of class labels at higher levels of the hierarchy is more important than the similarity of class labels at lower levels. Moreover, this allows us to take the hierarchy into account and to also deal with classes which are structured according to a directed acyclic graph (DAG).

The weighted Euclidean distance is given as:

$$d(L_1,L_2) = \sqrt{\sum_{l=1}^{|L|} \omega(c_l) \cdot (L_{1,l} - L_{2,l})^2} \tag{8.1}$$

where $L_{i,l}$ is the i-th component of the class vector L_i. The class weights $\omega(c)$ decrease with the depth of the classes in the hierarchy. More precisely, $\omega(c) = \omega_0 \cdot \text{avg}_j \{\omega p_j(c))\}$, where $p_j(c)$ denotes the j-th parent of class c and $0 < \omega_0 < 1$). For instance, consider a small hierarchy, given in Figure 8.2 and two data examples: (X_1,S_1) and (X_2,S_2) that belong to the classes $S_1 = \{c_1,c_2,c_3\}$ and $S_2 = \{c_2,c_4\}$, respectively. The class vectors would be $L_1 = [1,1,1,0]$ and $L_2 = [0,1,0,1]$. The class c_0 represents the root node and is ommited.

The new adapted Moran's I for HMC, $I(U)$, for a given set of examples U, is given as:

$$I(U) = 1/2 + \frac{(|\,U\,|) \cdot \sum_{L_i \in U} \sum_{L_j \in U} w_{ij} \cdot d(L_i,\overline{L}) d(L_j,\overline{L})}{2 \quad \cdot \quad \sum_{L_i \in U} \sum_{L_j \in U} w_{ij} \quad \cdot \quad \sum_{L_l \in U} d(L_l,\overline{L})^2} \tag{8.2}$$

where L_i is the vector representation of the class of the i-th example, $(\overline{L})$ is the vector representation of the set's mean class and $d(\cdot,\cdot)$ is the above distance function. In the vector L_i, the j-th component is 1 if the example belongs to class c_j ($j = 1,\ldots,m$) and 0 otherwise. In this definition m is the number

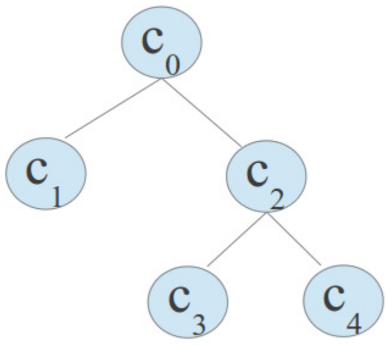


Figure 8.2: *An example of a hierarchy of class labels.* Class label names are consecutive and contain information about the position in the hierarchy.

of classes and each L_i has to satisfy the hierarchy constraint. By using $I(U)$, it is possible to consider different effects of the autocorrelation phenomenon at different levels of the tree model, as well as at different levels of the hierarchy (according to the weights $\omega(c_j)$, higher levels of the tree will likely capture the regularities at higher levels of the hierarchy, i.e., non-stationary autocorrelation).

The weights reflect the relative position of the data points in the context space. In a spatial context, these weights can be defined from the spatial coordinates of the data points, as discussed in Section 2.2. In a network context, these weights are given explicitly with the specification of the edges in the network.

Just as the spatial Global Moran's I, the Moran's I for HMC, $I(U)$, has values in the interval $[0,1]$, where 1 (0) means strong positive (negative) autocorrelation and 0.5 means no autocorrelation.

8.2.3 Global Geary's C

In addition to the adaptation of the Global Moran's I for the HMC task, we have also adapted the Global Geary's C (Legendre, 1993) measure as well. It has also been originally designed as a measure of spatial autocorrelation. As for the new Global Moran's I for the HMC task, the new Global Geary's C for HMC is created by replacing the residuals in the standard Global Geary's C formula with the distance function $d(\cdot,\cdot)$, used for HMC tasks within the PCTs framework.

Let $(x_i, y_i) \in U \subseteq \mathbf{X} \times 2^C$ be an example pair in a training set U of N examples. Let K be the number of classes in C, possibly defining a hierarchy. We represent y_i as a binary vector L_i of size K, such that $L_{i,k} = 1$ if $c_k \in y_i$, $L_{i,k} = 0$ otherwise. Obviously, each L_i has to satisfy the hierarchical constraint (8.4). Let $d(L_i, L_j)$ be a distance measure defined for two binary vectors associated to two example pairs (x_i, y_i), (x_j, y_j). It can be any distance which can take the class-label hierarchy into account. Here we will take the definition from $d(\cdot,\cdot)$ given by formula 8.1.

For a given generic set of examples U, the new adapted Global Geary's C for HMC, labeled as $C(U)$, is given as:

$$C(U) = 1 - \frac{(\mid U \mid -1) \cdot \sum_{L_i \in U} \sum_{L_j \in U} w_{ij} \cdot d(L_i, L_j)^2}{4 \cdot \sum_{L_i \in U} \sum_{L_j \in U} w_{ij} \quad \cdot \quad \sum_{L_l \in U} d(L_l, \overline{L})^2} \tag{8.3}$$

where L_i is the vector representation of the class of the i-th example, $(\overline{L})$ is the vector representation of the set's mean class and $d(\cdot,\cdot)$ is the distance function discussed above. In the vector L_i, the j-th component is 1 if the example belongs to class c_j ($j = 1, \ldots, m$) and 0 otherwise. In this definition, m is the number of classes and L_i has to satisfy the hierarchy constraint. The constant 4 in the denominator is included for scaling purposes. Just as the spatial Global Geary's C, $C(U)$ has values in the interval $[0,1]$, where 1 (0) means strong positive (negative) autocorrelation and 0.5 means no autocorrelation.

The Global Geary's C is a more suitable measure for autocorrelation in a HMC task. It can better fit the general structure of network data in comparison to the Global Moran's I. This is because Global Geary's C relies on the distance function (relationships) between pairs of examples, whereas Global Moran's I is based on the distance function between the class of an example and the set's mean class.

8.3 Learning PCTs for HMC by Taking Network Autocorrelation into Account

The task of hierarchical multi-label classification (HMC). We next discuss the network setting that we consider and use the network autocorrelation measure for HMC tasks defined above. Subsequently, we describe the CLUS-HMC algorithm for HMC and introduce its extension NHMC (i.e., Network CLUS-HMC), which takes into account network autocorrelation for HMC.

For the HMC task, the input is a dataset consisting of example pairs $(x_i, y_i) \in \mathbf{X} \times 2^C$, where $\mathbf{X} = X_1 \times X_2 \ldots \times X_m$ is the space spanned by m attributes or features, while 2^C is the power set of $C = \{c_1, \ldots, c_K\}$, the set of all possible class labels. C is hierarchically organized with respect to a partial order $\preceq$ which represents the superclass relationship. Note that each y_i satisfies the *hierarchical constraint*:

$$c \in y_i \Rightarrow \forall c' \preceq c : c' \in y_i. \tag{8.4}$$

The method we propose (NHMC) is based on decision trees and is set in the PCTs framework (Blockeel et al, 1998). Recall (from Chapter 5 that this framework views a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all the data, that is recursively partitioned into smaller clusters when moving down the tree. PCTs combine elements from both prediction and clustering. As in clustering, clusters of data that are similar to each other are identified, but, in addition, a predictive model is also associated to each cluster. This predictive model provides a prediction for the target property of new examples that are recognized to belong to the cluster. The benefit of using PCTs is that, besides the clusters themselves, they also provide symbolic descriptions of the constructed (hierarchically organized) clusters.

The original PCTs framework is implemented in the CLUS system (Blockeel et al, 1998) (http://sourceforge.net/projects/clus/). The induction of PCTs is not very different from that of standard decision trees. The algorithm takes as input a set of training instances and searches for the best acceptable test to partition the instances and reduce the variance of the target. If such a test can be found, then the algorithm creates a new internal node and calls itself recursively to construct a subtree for each subset (cluster) in the partition induced by the test on the training instances.

8.3.1 The network setting for NHMC

Some uses of network data in learning predictive models include: *i)* treating the interactions between pairs of nodes as descriptive attributes (e.g., binary attributes, (Schietgat et al, 2010)); *ii)* generating new features as combinations of network data with other descriptive attributes. Both approaches demand that the data are pre-processed before applying a network oblivious learning method (e.g., CLUS-HMC). However, the applicability of predictive models built in this way strongly depends on network information availability for the test data.

In order to learn general models, which can be used to make predictions for any test set, here we use interactions as a form of background knowledge to exploit only in the learning phase. More specifically, in the training phase information on both node properties and network structure is considered, while in the testing phase the network structure is disregarded. This key feature of the proposed solution is especially attractive when prediction concerns new nodes for which interactions with other nodes are not known or are still to be confirmed.

Following Steinhaeuser et al (2011), we view a training set as a single network of labeled nodes. Formally, the network is defined as an undirected edge-weighted graph *G=(V,E)*, where V is the set of labeled *nodes*, while $E \subseteq \{\langle u,v,w\rangle \mid u,v \in V, w \in \mathbb{R}^+\}$ is the set of *edges* (i.e., the context space D), such that each edge $u \leftrightarrow v$ is assigned a nonnegative real number w, called the *weight* of the edge. It can be represented by a symmetric adjacency matrix $\mathbf{W}$, whose entries are positive ($w_{ij} > 0$) if there is an edge connecting i to j in G, and zero ($w_{ij} = 0$) otherwise. Here, edge weights express the strength of the interactions between network nodes and can be any non-negative weight values: in practice, binary $(0/1)$ weights could also be used, due to the limitations of available data in specific application domains. The edges represent the context space D which in this case in defined only using the distance function $d(;\dot{)}$.

Each node of the network is associated with an example pair $(x_i,y_i) \in \mathbf{X} \times 2^C$, where $y_i = \{y_{i_1}, y_{i_2}, ..., y_{i_q}\}, q \leq K$, is subject to the hierarchical constraint. Given a network $G = (V,E)$ and a function $\eta : V \longmapsto (\mathbf{X} \times 2^C)$ which associates each node with the corresponding example pair, we interpret the task of hierarchical multi-label classification as the task of building a PCT which represents a multi-dimensional predictive function $f : \mathbf{X} \longmapsto 2^C$ that satisfies the hierarchical constraint (8.4), maximizes the network autocorrelation of the observed class values y_i, for the examples that fall in the same leaves, and minimizes the predictive error on the training data $\eta(V)$.

8.3.2 Trees for Network HMC

In this section, we introduce the method NHMC (Network CLUS-HMC) which builds autocorrelation-aware HMC models. We shall start with a brief description of the algorithm CLUS-HMC which is the basis for the development of NHMC.

CLUS-HMC

The CLUS-HMC (Vens et al, 2008) algorithm builds HMC trees. These are very similar to classification trees, but each leaf predicts a hierarchy of class labels rather than a single class label. CLUS-HMC builds the trees in a top-down fashion as in classical tree induction algorithms, with key differences both in the search heuristics and in making predictions.

Search heuristics. To select the best test in an internal node of the tree, the algorithm hierarchically scores the tests according to the reduction in variance induced on the set U of examples associated to the node. Recall that in Chapter 5 the variance in CLUS-HMC is defined as follows:

$$Var(U) = \frac{1}{\mid U \mid} \cdot \sum_{u_i \in U} d(L_i, \overline{L})^2, \tag{8.5}$$

where $d(\cdot,\cdot)$ is the distance function on vectors associated to set of class labels for examples in U, as defined by Equation 8.1.

At each node of the tree, the test that maximizes the variance reduction is selected. This is expected to result in maximizing cluster homogeneity with respect to the target variable, as well as in improving the predictive performance of the tree. If no test can be found that significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a prediction, which can consist of multiple hierarchically organized labels.

Predictions. A classification tree typically associates a leaf with the "majority class", i.e., the class most frequent in the training examples at the leaf, which is later used for prediction purposes when a test case reaches that leaf. However, in the case of HMC, where an example may have multiple classes, the notion of "majority class" cannot be straightforwardly applied. In fact, CLUS-HMC associates the leaf with the mean $\bar{L}$ of the class vectors of the examples in the leaf. The value at the k-th component of $\bar{L}$ is interpreted as the membership score, i.e., the probability of the example belonging to class c_k for an example arriving at the leaf.

The prediction for an example arriving at a leaf can be obtained by applying a user defined threshold τ on this probability. If the i-th component of $\bar{L}$ is above τ ($> \tau$), then the leaf predicts the class c_i. To ensure that the predictions fulfill the hierarchical constraint, i.e., whenever a class is predicted its super-classes are also predicted, it suffices to choose $\tau_i \leq \tau_j$ whenever c_j is ancestor of c_i.

The threshold τ is selected depending on the context. The user may set the threshold such that the resulting classifier has high precision at the cost of lower recall or vice versa, to maximize the F-score, to maximize the interpretability or plausibility of the resulting model, etc. It is also possible to use threshold-independent measures (such as precision-recall curves and the area under the ROC curve) to evaluate the performance of HMC trees.

Search space. In CLUS-HMC, for each internal node of the tree, the best split is evaluated by considering all available attributes. Let $X_i \in \{X_1, \ldots, X_m\}$ be an attribute and Dom_{X_i} its domain: A split partitions the current sample space D according to test $X_i \in B$, where $B \subseteq Dom_{X_i}$. This means that D is partitioned into two sets, D_1 and D_2, on the basis of the value of X_i.

For continuous attributes, possible tests are of the form $X \leq \beta$ while, for discrete attributes, they are in the form $X \in \{a_{i_1}, a_{i_2}, \ldots, a_{i_o}\}$ (where $\{a_{i_1}, a_{i_2}, \ldots, a_{i_o}\}$ is a non-empty subset of the domain Dom_X of X). In the former case, possible values of β are determined by sorting the distinct values of X in D, then considering the midpoints between pairs of consecutive values. For b distinct values, $b-1$ thresholds are considered. When selecting a subset of values for a discrete attribute, CLUS-HMC relies on the non-optimal greedy strategy proposed by Mehta et al (1996).

NHMC

We can now proceed to describe the top-down induction algorithm for building Network HMC trees. The search space is exactly the same as for CLUS-HMC, while the heuristic used are different. The network is considered as background knowledge to exploit only in the learning phase. Below, we first outline the algorithm and then give the details on the new search heuristics which take autocorrelation into account. A discussion follows on how the new search heuristics can be computed efficiently.

Outline of the algorithm. The top-down induction algorithm for building PCTs for HMC from network data is given below (Algorithm 4). It takes as input the network $G = (V, E)$ and the corresponding HMC dataset U defined by applying $\eta : V \mapsto \mathbf{X} \times 2^C$ to the vertices of the network. It then recursively partitions U until a stopping criterion is satisfied (Algorithm 4 line 2). Since the implementation of this algorithm is based on the implementation of the CLUS-HMC algorithm, we call this algorithm NHMC (Network CLUS-HMC).

Heuristics. The major difference between NHMC and CLUS-HMC is in the heuristic we use for the evaluation of each possible split. The variance reduction heuristics employed in CLUS-HMC aims at finding accurate models, since it considers the homogeneity in the values of the target variables and reduces the error on the training data. However, it does not consider the dependencies of the target variables values between related examples and therefore neglects the possible presence of autocorrelation in the training data. To address this issue, we introduce network autocorrelation as a search heuristic and combine it with the variance reduction in a new heuristics.

More formally, the NHMC heuristic is a linear combination of the average autocorrelation measure

Algorithm 4 *Top-down induction of NHMC.*

```
1: procedure NHMC($G,U$) returns tree
2: if stop($U$) then
3:     return leaf(Prototype($U$))
4: else
5:     $(t^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6:     for each possible Boolean test $t$ according to the values of $X$ in $U$ do
7:         $\mathcal{P} = \{U_1, U_2\}$ partition induced by $t$ on $U$
8:         $h = \alpha \cdot \left( \frac{|U_1| \cdot A_Y(U_1) + |U_2| \cdot A_Y(U_2)}{|U|} \right) +$
9:             $+(1-\alpha) \cdot \left( Var'(U) - \frac{|U_1| \cdot Var'(U_1) + |U_2| \cdot Var'(U_2)}{|U|} \right)$
10:        if $(h > h^*)$ then
11:            $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
12:        end if
13:    end for
14:    $tree_1$ = NHMC($G, U_1$)
15:    $tree_2$ = NHMC($G, U_2$)
16:    return node($t^*, tree_1, tree_2$)
17: end if
```

$A_Y(\cdot)$ (first term) and the variance reduction $Var(\cdot)$ (second term):

$$h = \alpha \cdot \left(\frac{|U_1| \cdot A_Y(U_1) + |U_2| \cdot A_Y(U_2)}{|U|} \right) + (1-\alpha) \cdot \left(Var'(U) - \frac{|U_1| \cdot Var'(U_1) + |U_2| \cdot Var'(U_2)}{|U|} \right) \tag{8.6}$$

where $Var'(U)$ is the min-max normalization of $Var(U)$ required to keep the values of the linear combination in the unit interval $[0,1]$, that is:

$$Var'(U) = \frac{Var(U) - \delta_{min}}{\delta_{max} - \delta_{min}}, \tag{8.7}$$

with δ_{max} and δ_{min} being the maximum and the minimum values of $Var(U)$ over all tests.

We point out that the heuristic in NHMC combines information on both the network structure, which affects $A_Y(\cdot)$, and the hierarchical structure of the class, which is embedded in the computation of the distance $d(\cdot,\cdot)$ used in formulae (8.3) and (8.5) to calculate the variance and its reduction.

We also note that the tree structure of the NHMC model makes it possible to consider different effects of the autocorrelation phenomenon at different levels of the tree model, as well as at different levels of the hierarchy (non-stationary autocorrelation). In fact, the effect of the class weights $\omega(c_j)$ in the formula 8.1 is that higher levels of the tree will likely capture the regularities at higher levels of the hierarchy.

However, the efficient computation of distances according to formula (8.1) is not straightforward. The difficulty comes from the need of computing $A(U_1)$ and $A(U_2)$ *incrementally*, i.e., based on statistics already computed for other partitions. Indeed, the computation of $A(U_1)$ and $A(U_2)$ from scratch for each partitioning would increase the time complexity of the algorithm by an order of magnitude and would make the learning process too inefficient for large datasets.

Efficient computation of the heuristics. In our implementation, different tests are generated by progressively moving examples from one partition to the other, specifically from U_2 to U_1). Consequently $Var(\cdot)$ can be computed according to classical methods for the incremental computation of variance. As to equation (8.3), the numerator only requires distances, which can be computed in advance. Therefore, the problem remains only for the denominator of equation (8.3). To compute it incrementally, we consider the following algebraic transformations:

$$\begin{aligned}
&\textstyle\sum_{u_i \in U} d(L_i, \overline{L_U})^2 \\
&= \sum_{u_i \in U} \sum_{k=1}^{K} \omega(c_k)(L_{i,k} - \overline{L_U}_k)^2 \\
&= \sum_{k=1}^{K} \omega(c_k) \sum_{u_i \in U} (L_{i,k} - \overline{L_U}_k)^2 \\
&= \sum_{k=1}^{K} \omega(c_k) \sum_{u_i \in U'} (L_{i,k} - \overline{L_{U'}}_k)^2 + (L_{t,k} - \overline{L_{U'}}_k)(L_{t,k} - \overline{L_U}_k) \\
&= \sum_{u_i \in U'} d(L_i, \overline{L_{U'}})^2 + (L_{t,k} - \overline{L_{U'}}_k)(L_{t,k} - \overline{L_U}_k)
\end{aligned}$$

where $U = U' \cup \{u_t\}$ and $\overline{L_U}$ ($\overline{L_{U'}}$) is the average class vector computed on U (U').

Time complexity. In NHMC, the time complexity of selecting a splitting test represents the main cost of the algorithm. In the case of a continuous split, a threshold β has to be selected for a continuous variable. If N is the number of examples in the training set, then the number of distinct thresholds can be N-1 at worst. The determination of candidate thresholds requires an ordering (sorting) of the examples, with time complexity of $O(m \cdot N \cdot \log N)$, where m is the number of descriptive variables. For each variable, the system has to compute the heuristic h for all possible thresholds. In general, this computation has time-complexity $O((N-1) \cdot (N + N \cdot s) \cdot K)$, where $N-1$ is the number of thresholds, s is the average number of edges for each node in the network, K is the number of classes, $O(N)$ is the complexity of the computation of the variance reduction and $O(N \cdot s)$ is the complexity of the computation of autocorrelation. However, according to the analysis reported before, it is not necessary to recompute the autocorrelation values from scratch for each threshold. This optimization makes the complexity of the evaluation of the splits for each variable $O(N \cdot s \cdot K)$. This means that the worst case complexity of creating a splitting node on a continuous attribute is $O(m \cdot (N \cdot \log N + N \cdot s) \cdot K)$.

In the case of a split by a discrete variable, the worst case complexity (for each variable and in case of optimization) is $O((d-1) \cdot (N + N \cdot s) \cdot K)$, where d is the maximum number of distinct values of a discrete variable ($d \leq N$).

Overall, the identification of the best splitting test (either for a continuous or a discrete variable) has a complexity of $O(m \cdot (N \cdot \log N + N \cdot s) \cdot K)$+$O(m \cdot d \cdot (N + N \cdot s) \cdot K)$, that is $O(m \cdot N \cdot (\log N + d \cdot s) \cdot K)$. This complexity is similar to that of CLUS-HMC, except for the s factor which may equal N in the worst case, although such a worst-case behavior is unlikely.

Additional remarks. The relative influence of the two parts of the linear combination in Formula (8.6) is determined by a user-defined coefficient α that falls in the interval [0,1]. When $\alpha = 0$, NHMC uses only autocorrelation, when $\alpha = 0.5$, it weights equally variance reduction and autocorrelation, and when $\alpha = 1$ NHMC works just as the original CLUS-HMC algorithm. If autocorrelation is present, examples with high autocorrelation will fall in the same cluster and will have similar values of the response variable (gene function annotation). In this way, we are able to keep together connected examples without forcing splits on the network structure (which can result in losing generality of the induced models).

At the end, note that the linear combination that we use here (Formula (8.6)) was selected as a results of our previous work deling with autocorrelated data (Stojanova et al, 2012). The variance and autocorrelation can also be combined in some other way (e.g., in a product). Investigating different ways of combining them is one of the directions for our future work.

8.4 Empirical Evaluation

In this section, we present the evaluation of the system NHMC on several datasets related to predicting gene function in yeast. Before we proceed to presenting the empirical results, we provide a description of the datasets used and of the experimental settings.

8.4.1 Data Sources

We use 12 yeast (*Saccharomyces cerevisiae*) datasets as considered by Clare and King (2003), but with new and updated class labels (Vens et al, 2008). The datasets describe different aspects of the genes in the yeast genome. They include five types of bioinformatics data: sequence statistics, phenotype, secondary structure, homology and expression. The different sources of data highlight different aspects related to gene function.

The **seq** dataset records sequence statistics that depend on the amino acid sequence of the protein for which the gene codes. These include amino acid frequency ratios. They also include label properties such as sequence length, molecular weight and hydrophobicity.

The **pheno** dataset contains phenotype data. These data represent the growth or lack of growth of knock-out mutants that are missing the gene in question. The gene is removed or disabled and the resulting organism is grown with a variety of media to determine what the modified organism might be sensitive or resistant to.

The **struc** dataset includes features computed from the secondary structure of the yeast proteins. The secondary structure is not known for all yeast genes, but can be predicted from the protein sequence with reasonable accuracy, using software such as Prof (Ouali and King, 2000). Due to the relational nature of secondary structure data, Clare and King (2003) performed a preprocessing step of relational frequent pattern mining. The **struc** dataset includes the constructed patterns as binary attributes.

The **hom** dataset includes, for each yeast gene, information from other, homologous genes. Homology is usually determined by sequence similarity; here, PSI-BLAST (Altschul et al, 1997) was used to compare yeast genes both with other yeast genes and with all genes indexed in SwissProt v39. This provided (for each yeast gene) a list of homologous genes. For each of these, various properties were extracted (keywords, sequence length, names of databases they are listed in, ...). Clare and King (2003) preprocessed these data in a similar way as they preprocess the secondary structure data to produce binary attributes.

The **cellcycle, church, derisi, eisen, gasch1, gasch2, spo, exp** datasets are expression datasets. They include microarray yeast data (Clare and King, 2003). The attributes for these datasets are real valued, representing fold changes in expression levels.

We construct two versions of each dataset. The values of the descriptive attributes are identical in both versions, but the classes are taken from two different classification schemes. In the first version, they are from FUN (http://www.helmholtz-muenchen.de/en/mips/projects/funcat), a scheme for classifying the functions of gene products, developed by MIPS (Ruepp et al, 2004). FUN is a tree-structured class hierarchy; a small part is shown in Figure 8.2(a). In the second version of the data sets, the genes are annotated with terms from the Gene Ontology (GO) (Ashburner et al, 2000) (http://www.geneontology.org), which forms a directed acyclic graph instead of a tree: each term can have multiple parents (we use GO's "is-a" relationship between terms). Only annotations from the first six levels are used. Note that GO has an order of magnitude more classes than FUN for our datasets. The 24 resulting datasets can be found at the webpage http://kt.ijs.si/daniela_stojanova/NHMC/.

In addition, we use several protein-protein interaction networks (PPIs) for yeast genes as in (Rahmani et al, 2010). In particular, the networks DIP (Deane et al, 2002), VM (von Mering et al, 2002) and MIPS (Mewes et al, 1999) are used, which contain 51233, 65982 and 38845 interactions among 7716, 2399 and 40374 proteins, respectively. DIP (Database of Interacting Proteins) stores and organizes information on binary protein-protein interactions that are retrieved from individual research articles. VM stores protein-protein interactions that are retrieved from numerous sources, including experimental data, computational prediction methods and public text collections. Finally, MIPS represents interactions between proteins on the basis of their signal transduction.

The basic properties of the datasets and the networks are given in Table 8.1.

8.4.2 Experimental Setup

In the experiments, we deal with several dimensions: different descriptions of the genes, different descriptions of gene functions, and different gene interaction networks. We have 12 different descriptions of the genes from the Clare and King' datasets (Clare and King, 2003) and 2 class hierarchies (FUN and GO), resulting in 24 datasets with several hundreds of classes each. Furthermore, we use 3 different PPI networks (DIP, VM and MIPS) for each of those. Moreover, for each dataset, we extracted the subset containing only the genes that are most connected, i.e., have at least 15 interactions in the PPI network (*highly connected genes*).

As suggested by Vens et al (2008), we build models trained on 2/3 of each data set and test on the remaining 1/3. The subset of highly connected genes may include genes that belong to the testing or the training set. To prevent over-fitting, we used two pre-pruning methods: minimal number of examples in a leaf (set to 5) and F-test pruning. The latter uses the F-test to check whether the variance reduction is statistically significant at a given level (0.001, 0.005, 0.01, 0.05, 0.1, 0.125).

Following Vens et al (2008), we evaluate the proposed algorithm by using the Average Area Under the Precision-Recall Curve ($\overline{AUPRC}$), i.e., the (weighted) average of the areas under the individual (per class) Precision-Recall (PR) curves, where all weights are set to $1/\mid C\mid$, with C the set of classes. The closer the $\overline{AUPRC}$ is to 1.0, the better the model is. A PR curve plots the precision of a classifier as a function of its recall. The points in the *PR* space are obtained by varying the value for the threshold τ from 0 to 1 with a step of 0.02. In the considered datasets, the positive examples for a given class are rare as compared to the negative ones. The evaluation by using *PR* curves (and the area under them), is the most suitable in this context, because we are more interested in correctly predicting the positive instances (i.e., that a gene has a given function), rather than correctly predicting the negative ones.

In order to evaluate the performance of the proposed NHMC algorithm, we compare it to CLUS-HMC (NHMC works just as CLUS-HMC when $\alpha = 1$). Moreover, we report the results of NHMC with $\alpha = 0$, when it uses only autocorrelation, and with $\alpha = 0.5$, when it equally weights variance reduction and autocorrelation.

8.4.3 Comparison between CLUS-HMC and NHMC

For each of the datasets, the $\overline{AUPRC}$ of CLUS-HMC (which does not consider network information) and NHMC, which uses the DIP PPI networks is shown in Table 8.2. For each algorithm and dataset, both FUN and GO annotations are considered. Finally, two variants of each dataset and network are considered, one with all genes and the other with the subset of highly connected genes (with at least 15 connections).

When using all genes, the best results are obtained by using CLUS-HMC for FUN annotations and NHMC with $\alpha = 0.5$ for GO annotations (even if, in this case, the average good results are mainly due to the results obtained on the cellcycle dataset). This can be explained by the fact that only a half of the genes have at least one connection to other genes in the PPI networks and this is not enough to improve the predictive accuracy of the global predictive HMC model that is constructed using NHMC over the model constructed by using CLUS-HMC.

NHMC shows competitive results with respect to CLUS-HMC when using GO annotations because of the larger percentage of function-relevant connections (see Table 8.1) which indicates that DIP network presents some form of autocorrelation on the GO labels. Moreover, the results obtained on GO annotated datasets by using NHMC with $\alpha = 0.5$ and $\alpha = 0$ are similar, indicating that the combination of variance reduction and autocorrelation does not lead to significant accuracy improvement.

As expected, the introduction of the PPIs is beneficial in the case of using the subset of highly connected genes, for both FUN and GO annotations. In this case, the best results are obtained when using NHMC with $\alpha = 0$, i.e., by using only the autocorrelation in gene function as a search heuristic. The average gain is 0.07 (which is an improvement of 54 % on average) in the case of the FUN hierarchy and 0.04 (43 %) in the case of the GO hierarchy. The advantage of NHMC over CLUS-HMC comes from the simultaneous use of the hierarchy of classes and the PPI information in protein function prediction and confirms the benefits coming from the consideration of autocorrelation during the learning phase. Indeed, this is also possible because of the tree structure of the learned models which consider different effects of autocorrelation at different levels of granularity.

In addition, in Figure 8.3, we present the $\overline{AUPRC}$ results obtained by using CLUS-HMC and NHMC (with $\alpha = 0.5$ and $\alpha = 0$) for GO annotations of the gasch2 and cellcycle datasets. These results are obtained by varying the portions of considered testing data on the basis of the minimum number of connections of a node in the DIP PPI networks. In this way, it is possible to see the effect on the $\overline{AUPRC}$ results, as we concentrate on highly connected nodes. Indeed, in this case, the results show a clear advantage of NHMC (using both $\alpha = 0.5$ and $\alpha = 0$) over CLUS-HMC and that this advantage increases when considering datasets with more connected genes. These results are particularly interesting if we consider that we are not including network information on the testing set. This means that network information extracted from the training set is profitably used in the classification phase.

8.4.4 Comparison with Other Methods

We also compare the results of NHMC to the results of recent bio-inspired strategies which work in the HMC setting, but do not consider network information, such as Artificial Neural Networks (HMC-LMLP), Ant Colony Optimization (hmAnt-Miner), as well as a genetic algorithm for HMC (HMC-GA) (Cerri et al, 2012). While the first algorithm is a 1-vs-all (it solves several binary classification problems) method based on artificial neural networks trained with the Back-propagation algorithm, the latter two are methods that discover HMC rules. The algorithms are evaluated on 7 yeast FUN annotated datasets (Clare and King, 2003) using the same experimental setup as for CLUS-HMC and NHMC.

In Table 8.3, we present the $AU\overline{PRC}$ results obtained by using HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC ($\alpha = 0.5$) on several FUN annotated datasets. NHMC outperforms all other methods with a great margin. An exception is only the church dataset, for which NHMC performs worse than hmAnt-Miner. Note that $AU\overline{PRC}$ (Cerri et al, 2012) is similar to $\overline{AUPRC}$, but uses weights that consider the number of examples in each class - $AU\overline{PRC}$ is used here to make results easily comparable with results obtained with the other competitive methods.

8.4.5 Comparison of Different PPI Networks in the Context of Gene Function Prediction by NHMC

Although all PPI networks are frequently updated and maintained, many works have pointed out that the PPI networks are also very noisy (e.g., (Shi et al, 2011)). In this paper, we argue that NHMC can be a valid tool for assessing the quality of network data in the perspective of exploiting information coming from PPI networks for gene function prediction. Before we compare the results obtained by NHMC using different PPI networks, we discuss some functional and topological properties of the 3 considered yeast PPI networks: DIP, VM and MIPS.

Table 8.1 shows the percentage of proteins that are covered by the PPI networks. On average, only a half of the proteins are known to interact with other proteins. DIP covers the highest percentage of proteins. However, this percentage is not much different from that of the other two networks, especially from MIPS.

In addition, Table 8.1 shows the percentage of function-relevant interactions. An interaction is considered to be function-relevant if the two proteins involved in the interaction have at least one function in common (with respect to a given hierarchy). As it is possible to see, 6 %-23 % observed interactions are relevant. However, a closer look at the statistics reveals that connections are more function-relevant with respect to GO annotations than with respect to FUN annotations. This is expected, as GO contains a much larger number of functions.

The modular nature of PPI networks is reflected by a high degree of clustering, measured by the clustering coefficient. The clustering coefficient measures the local cohesiveness around a node, and it is defined, for any node i, as the fraction of neighbors of i that are connected to each other:

$$C_i = \frac{2n_i}{k_i(k_i-1)} \tag{8.8}$$

where n_i denotes the number of direct links between the k_i neighbors of node i. The clustering coefficient of a network is the average of the clustering coefficients of all the nodes in the network. Simply stated, the clustering coefficient C_i measures the presence of 'triangles' which have a corner at i. The high degree of clustering is based on local sub-graphs with a high density of internal connections, while being less tightly connected to the rest of the network. Table 8.1 presents the clustering coefficients of the considered PPI networks. As it is possible to see, the network with the highest clustering coefficient value is MIPS, followed by VM and DIP.

Having described some of the characteristics of the different PPI networks used in this study, we can now proceed with the comparison of the obtained results using these network data. In Figure 8.4, we present the $\overline{AUPRC}$ results of CLUS-HMC and NHMC for each dataset, for both FUN and GO annotations, obtained by using three different PPI networks (DIP, VM and MIPS), trained on the subset of the highly connected genes. The results show a clear advantage of NHMC against CLUS-HMC in most of the cases. Exceptions are the *struc* and *hom* datasets, where the high number of attributes leads NHMC to encounter the curse of dimensionality phenomenon (Hughes, 1968). A detailed analysis of the results reveals that $\overline{AUPRC}$ values obtained with the DIP network are better than values obtained by using the other two PPI networks. By pairwise comparing results obtained with DIP and MIPS networks, it is possible to see that, although they present the same *% of connected genes*, the *% of functional related genes* is higher for DIP. This explains the better predictive capabilities of models learned with the DIP network. Moreover, by comparing results obtained with DIP and VM networks, it is possible to see that, although the *% of function related genes* is higher for VM (this means that VM is *potentially* more autocorrelated on the label than DIP), the *% of connected genes* is lower. This inhibits our algorithm from obtaining competitive methods in the case of VM, if compared to those obtained with DIP. These considerations make the DIP network the most informative network to be exploited in learning autocorrelation-aware predictive models.

Apparently, the clustering coefficient does not seem to influence predictive accuracy. Indeed, clustering coefficient only explains (very) local autocorrelation. On the contrary, tree structured model learned by NHMC allow us to catch different effects of autocorrelation (non-stationarity of the autocorrelation phenomenon (Angin and Neville, 2008)) at different levels of the tree, and thus, at different granularity levels (both global and local).

Finally, results in Figure 8.4 show that, although for GO we have higher *% of function related genes* in all the PPI networks, due to the significantly higher number of classes, the learning task for GO is

Table 8.1: *Basic properties of the datasets and the PPI networks when predicting gene function in yeast.* We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by (Clare and King, 2003)) grouped by their functional (FUN and GO) annotation and 3 different PPI networks (DIP (Deane et al, 2002), VM (von Mering et al, 2002) and MIPS (Mewes et al, 1999)). In addition, the percentage of connected genes, the percentage of function related genes and the clustering coefficients are presented for each of the networks.

Annotation	Dataset	#Instan-ces	#Attri-butes	#Classes	% of connected genes			% of function related genes			Clustering coefficient		
					DIP	VM	MIPS	DIP	VM	MIPS	DIP	VM	MIPS
FUN	seq	3932	476	499	46	43	46	8	11	7	0.46	0.64	0.66
	pheno	1592	67	455	46	34	46	6	6	7	0.48	0.74	0.66
	struc	3838	19629	499	13	43	13	7	10	6	0.45	0.64	0.66
	hom	3848	47035	499	45	43	45	7	10	6	0.33	0.59	0.66
	cellcycle	3757	77	499	72	44	72	2	10	6	0.63	0.63	0.66
	church	3779	550	499	46	44	46	15	9	5	0.66	0.63	0.66
	derisi	2424	63	499	72	69	72	7	10	6	0.49	0.65	0.66
	eisen	3725	79	461	35	31	35	9	10	7	0.46	0.64	0.66
	gasch1	3764	172	499	47	44	47	9	10	6	0.49	0.63	0.66
	gasch2	3779	51	499	47	44	47	7	10	6	0.34	0.59	0.66
	spo	3703	79	499	48	44	48	3	4	3	0.46	0.63	0.66
	exp	3782	550	499	46	44	46	15	9	5	0.66	0.63	0.66
GO	seq	3900	476	4133	46	43	46	15	22	13	0.46	0.64	0.66
	pheno	1587	67	3127	46	34	46	16	18	3	0.58	0.74	0.66
	struc	3822	19629	4132	59	42	59	14	19	12	0.33	0.66	0.66
	hom	3567	47035	4126	48	47	48	14	22	12	0.33	0.66	0.66
	cellcycle	3751	77	4125	47	44	47	17	26	14	0.45	0.63	0.66
	church	3774	550	4131	46	44	46	13	23	13	0.66	0.63	0.66
	derisi	2418	63	3573	73	69	73	11	18	9	0.45	0.65	0.65
	eisen	3719	79	4119	35	31	35	19	25	15	0.49	0.65	0.66
	gasch1	3758	172	4125	47	44	47	19	26	14	0.42	0.61	0.66
	gasch2	3758	51	4131	47	44	47	17	26	14	0.40	0.40	0.66
	spo	3698	79	4119	48	44	48	17	25	14	0.46	0.63	0.66
	exp	3773	550	4131	46	44	46	39	23	13	0.66	0.63	0.66

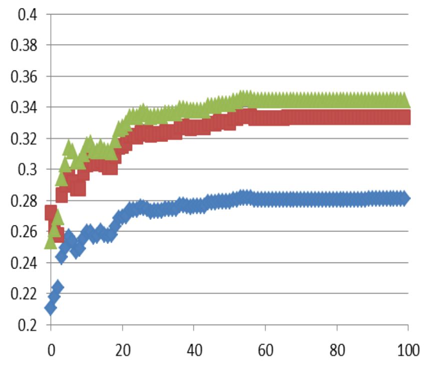
more complex than that for FUN. This explains the better $\overline{AUPRC}$ results for FUN in comparison to GO, when comparing the results of NHMC on the same datasets and using the same network.

8.5 Summary

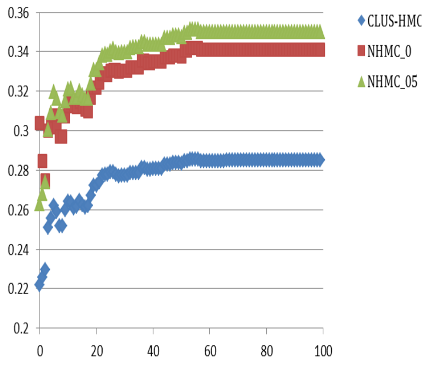
In this work, we tackle the problem of multi-label prediction of gene functional classes when relationships among the classes (instances may belong to multiple classes and classes are organized into a hierarchy), as well as relationships among the instances (instances may be connected in PPI networks) exist. The use of the latter relationships between the instances introduces autocorrelation and violates the assumption that instances are independently and identically distributed (i.i.d.), which underlines most machine learning algorithms. While this consideration introduces additional complexity to the learning process, it also carries substantial benefits.

The main contributions of this work are in the consideration of network autocorrelation in gene function prediction. Specifically, in this work, we presented a definition of network autocorrelation in the hierarchical multi-label classification setting, introduced an appropriate autocorrelation measure for autocorrelation in such setting and developed a method NHMC for hierarchial gene function prediction in a PPI network context.

Given a set of genes with known functions, NHMC learns to predict multiple gene functions when gene classes are hierarchically organized (and, possibly, in the form of DAGs), according to a hierarchial classification scheme, such as the MIPS-FUN and the Gene Ontology. During the learning process, NHMC takes into account PPI networks and the network autocorrelation of gene function that arises in this context. Due to the tree structure of the learned models, it is also possible to consider non-stationary

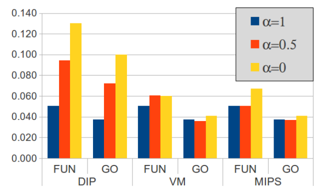


(a) gasch2

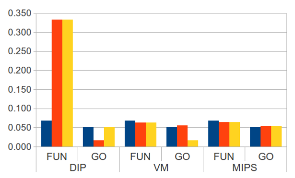


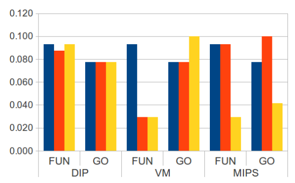
(b) cellcycle

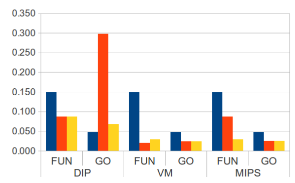
Figure 8.3: $\overline{AUPRC}$ *distribution* Comparison of the predictive models ($\overline{AUPRC}$ distribution) obtained by using CLUS-HMC and NHMC ($\alpha = 0.5$ and $\alpha = 0$) for the most connected subsets of (a) gasch2 and (b) cellcycle datasets for GO hierarchy. The horizontal axis gives the number of interactions from the DIP PPI network of the genes in the testing data, whereas the vertical axis gives the $\overline{AUPRC}$ values.

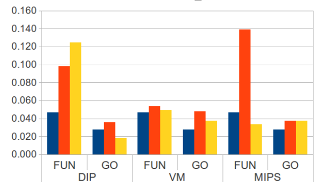


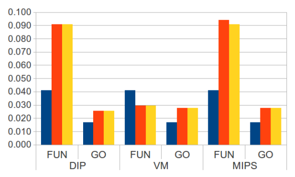
(a) seq

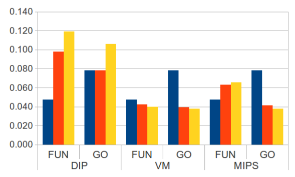

(b) pheno

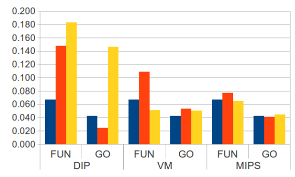

(c) struc

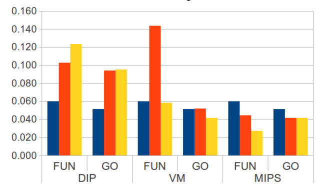

(d) hom

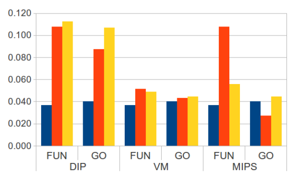

(e) cellcycle

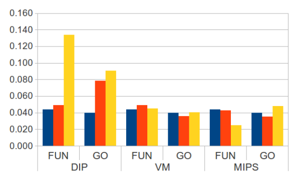

(f) church


(g) derisi


(h) eisen


(i) gasch1


(j) gasch2


(k) spo

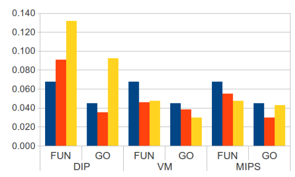

(l) exp

Figure 8.4: $\overline{AUPRC}$ *results* Comparison of the predictive models ($\overline{AUPRC}$ results) obtained by using CLUS-HMC and NHMC ($\alpha = 0.5$ and $\alpha = 0$) for each dataset using DIP, VM and MIPS PPI networks, for both FUN and GO annotations.

Table 8.2: *Comparison of $\overline{AUPRC}$ results.* The $\overline{AUPRC}$ of CLUS-HMC ($\alpha = 1$) and NHMC ($\alpha = 0.5$ and $\alpha = 0$) when predicting gene function in yeast. We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by (Clare and King, 2003)) grouped by their functional (FUN and GO) annotation, using all genes and the subset of highly connected genes and the DIP PPI network for yeast genes.

Dataset	All genes			Highly connected genes		
	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$
FUN annotated datasets						
seq	0.059	0.054	0.053	0.051	0.094	0.130
pheno	0.036	0.035	0.028	0.068	0.333	0.333
struc	0.030	0.020	0.020	0.093	0.088	0.093
hom	0.073	0.020	0.023	0.149	0.088	0.088
cellcycle	0.032	0.030	0.037	0.047	0.098	0.125
church	0.029	0.020	0.020	0.041	0.091	0.091
derisi	0.027	0.028	0.025	0.048	0.098	0.119
eisen	0.047	0.042	0.025	0.067	0.147	0.183
gasch1	0.036	0.040	0.032	0.060	0.103	0.124
gasch2	0.034	0.034	0.027	0.037	0.108	0.112
spo	0.030	0.029	0.025	0.044	0.049	0.134
exp	0.040	0.030	0.025	0.067	0.091	0.132
Average:	0.039	0.032	0.028	0.064	**0.116**	**0.139**
GO annotated datasets						
seq	0.034	0.032	0.030	0.037	0.072	0.100
pheno	0.019	0.016	0.016	0.051	0.016	0.051
struc	0.018	0.012	0.012	0.078	0.078	0.078
hom	0.040	0.013	0.013	0.047	0.068	0.068
cellcycle	0.019	0.287	0.288	0.027	0.036	0.018
church	0.014	0.015	0.012	0.017	0.025	0.025
derisi	0.017	0.015	0.017	0.078	0.078	0.106
eisen	0.030	0.024	0.024	0.043	0.061	0.146
gasch1	0.024	0.018	0.019	0.051	0.094	0.095
gasch2	0.020	0.021	0.021	0.040	0.088	0.107
spo	0.019	0.018	0.015	0.040	0.078	0.090
exp	0.023	0.017	0.016	0.045	0.036	0.092
Average:	0.022	**0.041**	**0.040**	0.046	**0.058**	**0.081**

Table 8.3: *Comparison with other methods* The $AU\overline{PRC}$ of HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC (using $\alpha = 0.5$ and the DIP PPI network), for 7 FUN annotated yeast datasets, as used in (Cerri et al, 2012).

Dataset	HMC-GA	HMC-LMLP	hmAnt-Miner	NHMC_05
pheno	0.148	0.085	0.162	0.241
cellcycle	0.150	0.144	0.154	0.173
church	0.149	0.140	0.168	0.152
derisi	0.152	0.138	0.161	0.172
eisen	0.165	0.173	0.180	0.196
gasch2	0.151	0.132	0.163	0.186
spo	0.151	0.139	0.174	0.181

autocorrelation, i.e., different effects of network autocorrelation at different levels of granularity. However, NHMC does not need the PPI network in the prediction phase, which is beneficial, especially in cases where the prediction needs to be made for new examples (genes) for which connections/interactions to other examples (genes) are not known or still need to be confirmed.

Empirical evidence shows that explicitly taking network autocorrelation into account increases the predictive capability of the models, especially when the considered PPI networks are dense. Furthermore, NHMC can be used as a tool to assess network data and the information it provides with respect to the hierarchy of functions used to classify the genes. This would give the opportunity of discarding less promising hypotheses, whose (wet-lab) validation is expensive and, to concentrate on the verification of potentially valid ones.

In future work, we intend to evaluate our approach by using additional datasets and networks. This will include datasets for organisms other that yeast. This will also include networks based on sequence similarities (usually implying homology) among genes.

9 Conclusions and Further Work

In this dissertation, we proposed several extensions of the predictive clustering approach for handling non-stationary autocorrelation in spatial and network data. We developed new heuristic functions that take into account both the variance of the target variables and its spatial/network autocorrelation. The advantages of the proposed extensions arise from the fact that they inherit the characteristics of the predictive clustering approach and at the same time deal with the global and local effects of the autocorrelation phenomenon, if it exists in the analyzed data.

We evaluated the proposed extensions on a wide range of real-world datasets with spatial and network data from a variety of domains, for single target classification and regression problems, as well as multi-target classification and regression problems. We also considered the hierarchical multi-label classification problem of gene function prediction in a network context. In this case protein-protein interaction networks were taken into account.

In the remainder of this chapter, we first summarize the scientific contributions of the dissertation. We then discuss how the proposed methods can be further improved and applied.

9.1 Contribution to science

The methods we propose in this dissertation extend the predictive clustering framework towards learning predictive models from autocorrelated (spatial and network) data. They also include new autocorrelation measures designed for use on existing and new predictive modeling tasks, rather than classification and regression. We summarize the contributions as follows:

- We defined the problem of considering autocorrelation in a general predictive modeling context, as supported by the predictive clustering paradigm and predictive clustering trees (PCTs) induction. Besides considering the classical predictive modeling tasks of classification and regression, we also consider the task of milti-target classification and regression, as well as Hierarchial Multi-Label Classification (Section 2.1). To the best of our knowledge, this is the first time that autocorrelation has been considered in a structured-output prediction setting. We focus on:

  – Autocorrelated data, i.e., data for which the independent and identically distributed (i.i.d.) assumption (Clauset, 2001), which typically underlies machine learning methods and multivariate statistics, is no longer valid. The assumption is important in the classical form of the central limit theorem, which states that the probability distribution of the sum (or average) of i.i.d. variables with finite variance approaches a normal distribution and tends to simplify the underlying mathematics of many statistical and machine learning methods.

  – A context space D is introduced, as a result of this, into the definition of the predictive modeling task that we consider. This dimension generalizes the explicitly given temporal and spatial dimension and the implicit space defined by a network. It serves as background knowledge and introduces information related to the modeling space, in the form of relations between the training examples.

- The tree structure of PCTs enables us to represent and capture for non-stationary autocorrelation, in contrast to the many machine leaning algorithms assuming that autocorrelation dependencies are stationary (Angin and Neville, 2008) (i.e., do not change) throughout the considered dimension (time, space, or network).

- We presented a broad overview of different autocorrelation measures introduced in the literature and their use in the case of non i.i.d. data. To our knowledge, such a broad overview has not been completed before (chapter 3)
  - The autocorrelation measures are organized according to the different forms of autocorrelation that they consider (spatial, temporal, spatio-temporal and network autocorrelation), as well as to the predictive (classification and regression) task that they are used for. In addition, we investigate the use of these autocorrelation measures in the induction of PCTs by taking into account variations in the global/ local data distribution across the network.
  - We defined new autocorrelation measures by generalizing the already existing ones, in order to be able to deal with the autocorrelation phenomenon within a variety of predictive modeling tasks.
    - New autocorrelation measure for spatial and network autocorrelation in for the classification task (Section 3.2.2, Section 3.2.2, Section 3.4.2).
    - New autocorrelation measures for network autocorrelation in the HMC setting (Section 8.2).
  - Moreover, we provided a theoretical discussion based on specific properties of autocorrelation that, when exploited in the PCTs induction, allow the discovery of clusters or dense zones of the network with similar values in the response variable.

- We presented a comprehensive review of state-of-the-art methods, proposed both in statistics and data mining, which explicitly consider the autocorrelation phenomenon (chapter 4). The review is very broad and systematic. The relevant methods are organized according to the different forms of autocorrelation that they consider (spatial, temporal, spatio-temporal and network autocorrelation), as well as to the predictive (classification and regression) task that they concern. Again, such a broad review, organized along these two dimensions together, has not been done before.

- The major contributions of this dissertation are three extensions of the predictive clustering approach for handling non-stationary (spatial and network) autocorrelated data for different predictive modeling tasks.
  - These include:
    - **SCLUS** (Spatial Predictive Clustering System) (Stojanova et al, 2011) (chapter 6), that explicitly considers spatial autocorrelation in regression (and classification),
    - **NCLUS** (Network Predictive Clustering System) (Stojanova et al, 2011, 2012) (chapter 7), that explicitly considers network autocorrelation in regression (and classification), and
    - **NHMC** (Network Hierarchical Multi-label Classification) (Stojanova et al, 2012) (chapter 8), that explicitly considers network autocorrelation in hierarchical multi-label classification.
  - The algorithms are heuristic: we define new heuristic functions that take into account both the variance of the target variables and its spatial/network autocorrelation. Different combinations of these two components enable us to investigate their influence in the heuristic function and on the final predictions.

  - To the best of our knowledge, this is the first approach that uses predictive clustering trees that consider the autocorrelation phenomenon, in particular the autocorrelation that comes from the spatial arrangements of objects in space and the network homophily. The advantages of the proposed methods arise from the fact that they inherit the characteristics of the predictive clustering approach and at the same time deal with the global and local effects of the autocorrelation phenomenon, if it exists in the analyzed data.
  - Within these extensions, we have developed techniques to automatically estimate the size of the neighborhood where the effects of autocorrelation are the highest.

- We have performed extensive empirical evaluation of the newly developed methods on single target classification and regression problems, as well as multi-target classification and regression problems. We consider 9 datasets for the regression task and 4 datasets for the classification task for the evaluation of SCLUS, 11 datasets for the evaluation of NCLUS and 24 datasets for the evaluation of NHMC.
  - We have compared the performance of the proposed predictive models for classification and regression tasks, when predicting single and multiple targets simultaneously to current state-of-the-art methods. Our approaches compare very well to mainstream methods that do not consider autocorrelation (support vector regression (SVR), *k*-Nearest Neighbor and M5' Model Trees), as well as to well-known methods that consider autocorrelation (Geographically Weighted Regression (GWR) that considers spatial autocorrelation). Furthermore, our approach can more successfully remove the autocorrelation of the errors of the obtained models. Finally, the obtained predictions are more coherent in space (or in the network context).
  - We have also applied the proposed predictive models to real-word problems, such as the prediction of outcrossing rates from genetically modified crops to conventional crops in ecology (Stojanova et al, 2012), prediction of the number of views of online lectures (Stojanova et al, 2011, 2012) and protein function prediction in functional genomics (Stojanova et al, 2012).

9.2 Further Work

In this dissertation, we presented several extensions of the methods for learning predictive clustering trees in the direction of taking into account autocorrelation in the training data. The methods developed can take into account non-stationary spatial and network autocorrelation on neighborhoods of varying sizes, where the appropriate size of the neighborhood (bandwidth) is determined automatically. They can be used to solve three different tasks of predicting structured outputs: multi-target regression, multi-target classification and hierarchical multi-label classification. The approaches developed were empirically evaluated on a variety of real-world datasets, including datasets from the area of ecological and bioinformatics.

Due to the generality of our approach and the broad range of methods developed and problems addressed, many directions for further work have opened up during the research presented in this thesis. On one hand, many improvements and extensions of the proposed approaches can be conceived. On the other hand, the already developed approaches and their possible improvements/extensions can be applied to many new practically relevant problems.

At least two improvements of the developed methods come to mind immediately. First, other ways of determining an appropriate bandwidth or neighborhood size should be explored. Second, better ways to combine variance reduction and autocorrelation in the heuristic functions for learning PCTs are needed. For example, one might consider automated tuning of the parameter α that determines the relative importance of the two in the combined heuristic. Nonlinear methods for combining the two might be considered as well.

In terms of extensions of the developed methods, we should consider also the types of autocorrelation not addressed in this thesis, such as temporal and spatio-temporal autocorrelation. Other types of structured targets should be considered as well, such as short time series of real numbers. In this context, new autocorrelation will need to be defined, implemented and included within the context of learning predictive clustering trees from autocorrelated data.

Moreover, the methods we presented can also be extended to the transduction setting. In this way, the methods can benefit from the advantages of the transductive approach over the basic inductive approach when few labeled examples are available and manual annotation is fairly expensive ((Appice et al, 2010; Ceci et al, 2010)). Extensions in the direction of transfer learning can also be envisaged.

In terms of applications, many possible directions can be explored for gene function prediction in the context of protein-protein interaction networks. Different organisms can be considered as well as different types of input data. Also, different types of networks can be considered, such as networks derived from homology data.

In the area of ecological modeling, different task of habitat modeling can be considered in a spatial context. Different individual species may be considered, as well as entire communities: Predicting community structure is a task of multi-target prediction. Finally, if we consider the taxonomy of living organisms, this becomes a task of hierarchical multi-label classification.

10 Acknowledgments

I would like to thank my advisor, Prof. Dr. Sašo Džeroski for his collaboration and support of this research. His ideas and suggestions have guided my work and improved the quality of my research. Also, I would like to thank the Department of Knowledge Technologies of the Jožef Štefan Institute for supporting my research.

I owe a debt of gratitude to my committee members, Prof. Dr. Marko Bohanec, Assoc. Prof. Janez Demšar and Asst. Prof. Michelangelo Ceci, who each have helped me to expand the breadth of my research by providing me insights into their areas of expertise.

I would like to thank my colleagues from the Department of Knowledge Technologies of the Jožef Štefan Institute, especially Aleksandar Pečkov, Elena Ikonomovska, Valentin Gjorgjioski and Katerina Taškova for many interesting, insightful, and thought provoking conversations related to my research topic. I would like to express my deepest appreciations for the time and the effort they put in our friendship and their support during the difficult times of our doctoral studies. I would also like to thank my office mates Darko Aleksovski and Vladimir Kuzmanovski for the many inspirational moments during the period of my doctoral studies. I am thankful to my friends Jožica Gričar, Ljupka Stojčevska, Violeta Mirčevska, Marina Katanič and Jelena Gajevič who know how to make me laugh and always stand by my side.

I thank the colleagues from the LACAM (Knowledge Acquisition and Machine Learning Lab) research laboratory of the Department of Computer Science at the Università degli Studi di Bari "Aldo Moro" in Bari, Italy that I visited during my doctoral studies. Especially, I would like to thank Asst. Prof. Michelangelo Ceci and Asst. Prof. Annalisa Appice for their collaboration, insightful suggestions, enthusiasm, support and research directions.

Last, but not least, I am deeply grateful to my family for their continuous moral support and acceptance of my long hours away from them.

11 References

Aha, D.; Kibler, D. Instance-based learning algorithms. *Machine Learning* **6**, 37–66 (1991).

Altschul, S.; Madden, T.; Schaffer, A.; Zhang, J.; Zhang, Z.; Miller, W.; Lipman, D. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research* **25**, 3389–3402 (1997).

Angin, P.; Neville, J. A shrinkage approach for modeling non-stationary relational autocorrelation. In: *Proc. 8th IEEE Intl. Conf. on Data Mining*. 707–712 (IEEE Computer Society, Washington, DC, 2008).

Anselin, L. *Spatial Econometrics: Methods and Models* (Kluwer, Dordrecht, 1988).

Anselin, L.; Bera, A. Spatial dependence in linear regression models with an application to spatial econometrics. In: Ullah, A.; Giles, D. (ed.) *Handbook of Applied Economics Statistics*. 21–74 (CRC Press, Boca Raton, FL, 1998).

Antulov-Fantulin, N.; Bošnjak, M.; Žnidaršič, M.; Grčar, M.; Morzy, M.; Šmuc, T. Discovery challenge overview. In: *Proc. ECML-PKDD Discovery Challenge Workshop*. 7–20 (IEEE Computer Society, Washington, DC, 2011).

Appice, A.; Ceci, M.; Malerba, D. An iterative learning algorithm for within-network regression in the transductive setting. In: *Proc. 12th Intl. Conf. on Discovery Science*. 36–50 (Springer, Berlin, 2009).

Appice, A.; Ceci, M.; Malerba, D. Transductive learning for spatial regression with co-training. In: *Proc. of 12th Annual ACM Symposium on Applied Computing*. 1065–1070 (ACM, New York, NY, 2010).

Arthur, G. A history of the concept of spatial autocorrelation: A geographer's perspective. *Geographical Analysis* **40**, 297–309 (2008).

Ashburner, M.; Ball, CA.; Blake, JA.; Botstein, D.; Butler, H.; Cherry, JM.; Davis, AP.; Dolinski, K.; Dwight, SS.; Eppig, JT.; Harris, MA.; Hill, DP.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, JC.; Richardson, JE.; Ringwald, M.; Rubin, GM.; Sherlock, G. Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics* **25**, 25–29 (2000).

Bakır, G. H.; Hofmann, T.; Schölkopf, B.; Smola, A. J.; Taskar, B.; Vishwanathan, S. V. N. *Predicting Structured Data* (MIT Press, Cambridge, MA, 2007).

Baltagi, B. *Econometric Analysis of Panel Data* (Wiley, San Francisco, CA, 2005).

Barutcuoglu, Z.; Schapire, R. E.; Troyanskaya, O. G. Hierarchical multi-label prediction of gene function. *Bioinformatics* **22**, 830–836 (2006).

Basak, D.; Pal, S.; Patranabis, D. Support vector regression. *Neural Information Processing - Letters and Reviews* **11**, 203–224 (2007).

Batagelj, V.; Mrvar, A. PAJEK – Program for large network analysis. http://vlado.fmf.uni-lj.si/pub/networks/pajek/ (accessed: July, 2012).

Bel, D.; Allard, L.; Laurent, J.; Cheddadi, R.; Bar-Hen, A. CART algorithm for spatial data: Application to environmental and ecological data. *Computational Statistics and Data Analysis* **53**, 3082–3093 (2009).

Besag, J. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society* **36**, 192–236 (1974).

Bishop, C. *Pattern Recognition and Machine Learning* (Springer, Berlin, 2007).

Blockeel, H. *Top-down Induction of First Order Logical Decision Trees*. Ph.D. thesis (Department of Computer Science, Katholieke Universitet, Leuven, Belgium, 1998).

Blockeel, H.; De Raedt, L.; Ramon, J. Top-down induction of clustering trees. In: *Proc. 15th Intl. Conf. on Machine Learning*. 55–63 (Morgan Kaufmann, San Mateo, CA, 1998).

Blockeel, H.; Struyf, J. Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research* **3**, 621–650 (2002).

Bogorny, V.; Valiati, J. F.; da Silva Camargo, S.; Engel, P. M.; Kuijpers, B.; Alvares, L. O. Mining maximal generalized frequent geographic patterns with knowledge constraints. In: *Proc. 6th IEEE Intl. Conf. on Data Mining*. 813–817 (IEEE Computer Society, Washington, DC, 2006).

Boriah, S.; Chandola, V.; Kumar, V. Similarity Measures for Categorical Data: A Comparative Evaluation. In: *Proc. 6th Int. SIAM Conf. on Data Mining*. 243–254 (SIAM, 2008).

Box, G.; Jenkins, G.; Reinsel, G. *Time Series Analysis, Forecasting, and Control* (Prentice-Hall, Newark, New Jersey, 1994).

Bratko, I. *Prolog Programming for Artificial Intelligence* (Addison Wesley, Boston, MA, 2000).

Breiman, L.; Friedman, J.; Olshen, R.; Stone, J. *Classification and Regression Trees* (Wadsworth & Brooks, Monterey, CA, 1984).

Brent, R. *Algorithms for Minimization without Derivatives* (Prentice-Hall, Newark, New Jersey, 1973).

Bullmore, E.; et al. Statistical methods of estimation and inference for functional MR image analysis. *Magnet Resonance Medical* **35**, 261–277 (1996).

Ceci, M.; Appice, A. Spatial associative classification: Propositional vs structural approach. *Journal of Intelligent Information Systems* **27**, 191–213 (2006).

Ceci, M.; Appice, A.; Malerba, D. Transductive learning for spatial data classification. In: *Proc. Advances in Machine Learning*. 189–207 (Springer, Berlin, 2010).

Cerri, R.; Barros, R. C.; de Carvalho, A. C. P. L. F. A genetic algorithm for hierarchical multi-label classification. In: *Proc. 27th Annual ACM Symposium on Applied Computing*. 250–255 (ACM, New York, NY, 2012).

Cheng, T.; Haworth, J.; Wang, J. Spatio-temporal autocorrelation of road network data. *Journal of Geographical Systems* **4**, 1–25 (2011).

Chuhay, R. Marketing via friends: Strategic diffusion of information in social networks with homophily. *Technical Report 2010.118* (Fondazione Eni Enrico Mattei, Higher School of Economics, Milan, Italy, 2010).

Clare, A. *Machine Learning and Data Mining for Yeast Functional Genomics*. Ph.D. thesis, (University of Wales at Aberystwyth, UK, 2003).

Clare, A.; King, R. D. Predicting gene function in S. Cerevisiae. In: *Proc. 8th European Conf. on Computational Biology*. 42–49 (ECCB, 2003).

Clauset, A. *A Brief Primer on Probability Distributions* (Santa Fe Institute, Santa Fe, NM, 2001).

Cliff, A.; Ord, J. Spatial and temporal analysis: autocorrelation in space and time. In: *Wrigley, N. and Bennett, R. (eds.) Quantitative Geography: A British View*. 104–110 (Routledge, London, 1981).

Cortez, P.; Morais, A. A data mining approach to predict forest fires using meteorological data. In: *Proc. 13th Portuguese Conf. on Artificial Intelligence*. 512–523 (Springer, Berlin, 2007).

Cressie, N. *Statistics for Spatial Data* (Wiley, San Francisco, CA, 1993).

Cui, R.; Lund, Y. A new look at time series of counts. *Biometrika* **96**, 781–792 (2009).

Davis, J. (ed.) *Statistics and Data Analysis in Geology*, 2nd edition (Wiley, San Francisco, 1986).

De Iaco, S.; Myers, S.; Posa, D. Nonseparable space-time covariance models: Some parametric families. *Journal of Mathematical Geology* **34**, 4577–4596 (2002).

de Jong, P.; Sprenger, C.; van Veen, F. On extreme values of Moran's I and Geary's C. *Geographical Analysis* **16**, 17–24 (1984).

Deane, C. M.; Salwiński, Ł.; Xenarios, I.; Eisenberg, D. Protein interactions. *Molecular and Cellular Proteomics* **1**, 349–356 (2002).

Debeljak, M.; Trajanov, A.; Stojanova, D.; Leprince, F.; Džeroski, S. Using relational decision trees to model out-crossing rates in a multi-field setting. *Ecological Modelling* **245**, 75–83 (2012).

Demšar, D.; Debeljak, M.; Lavigne, C.; Džeroski, S. Modelling pollen dispersal of genetically modified oilseed rape within the field. In: *Abstracts of the 90th ESA Annual Meeting*, 152 (The Ecological Society of America, Montreal, 2005).

Deza, M. *Encyclopedia of Distances* (Springer, Berlin, 2009).

Domingos, P. Prospects and challenges for multi-relational data mining. *SIGKDD Explorations Newsletter* **5**, 80–83 (2003).

Doreian, P. Network autocorrelation models: Problems and prospects. In: *Griffith, D. (eds.) Spatial Statistics: Past, Present, and Future* (Institute of Mathematical Geography, Ann Arbor, 1990).

Dubin, R. A. Spatial autocorrelation: A primer. *Journal of Housing Economics* **7**, 304–327 (1998).

Durbin, J.; Watson, G. S. Testing for serial correlation in least squares regression. *Biometrika* **37**, 409–428 (1950).

Džeroski, S.; Gjorgjioski, V.; Slavkov, I.; Struyf, J. Analysis of time series data with predictive clustering trees. In: *Proc. 5th Intl. Wshp. on Knowledge Discovery in Inductive Databases*. 63–80 (Springer, Berlin, 2007).

Engle, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom infation. *Econometrica* **50**, 987–1007 (1982).

Epperson, B. Spatial and space-time correlations in ecological models. *Ecological Modeling* **132**, 63–76 (2000).

Ester, M.; Kriegel, H.-P.; Sander, J. Spatial data mining: A database approach. In: *Proc. 5th Intl. Symp. on Spatial Databases* **1262**, 47–66 (Springer, Berlin, 1997).

Fortin, M.; Dale, M. R. T. *Spatial Analysis: A Guide for Ecologists* (Cambridge University Press, 2005).

Fotheringham, A. S.; Brunsdon, C.; Charlton, M. *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships* (Wiley, San Francisco, CA, 2002).

Gallagher, B.; Tong, H.; Eliassi-Rad, T.; Faloutsos, C. Using ghost edges for classification in sparsely labeled networks. In: *Proc. 14th SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 256–264 (ACM, New York, NY, 2008).

Getis, A.; Ord, J. K. The analysis of spatial association by use of distance statistics. *Geographical Analysis* **24**, 189–199 (1992).

Glotsos, D.; Tohka, J.; Soukka, J.; Ruotsalainen, U. A new approach to robust clustering by density estimation in an autocorrelation derived feature space. In: *Proc. 6th Nordic Symposium on Signal Processing*. 296–299 (IEEE Computer Society, Washington, DC, 2004).

Godfrey, L. G. Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica* **46**, 320–324 (1978).

Gora, G.; Wojna, A. RIONA: A classifier combining rule induction and k-NN method with automated selection of optimal neighbourhood. In: *Proc. 13th European Conf. on Machine Learning*. 111–123 (IOS Press, Amsterdam, 2002).

Griffith, D. Interdependence of space and time: numerical and interpretative considerations. In: *Grifit, D. (eds.) Dynamic Spatial Models*. 258–287 (Sijthoff & Noordhoff, Alphen aan den Rijn, Nederlands, 1981).

Griffith, D. *Spatial autocorrelation and spatial filtering: gaining understanding through theory and scientific visualization. Grifit, D. (eds.) Advances in Spatial Science* (Springer, Berlin, 2003).

Griffith, D.; Heuvelink, G. Deriving space-time variograms from space-time autoregressive (star) model specifications. In: *Proc. of StatGIS Conf.* 1–15 (International Society for Photogrammetry and Remote Sensing, Hong Kong, 2009).

Grčar, M.; Lavrač, N. A methodology for mining document-enriched heterogeneous information networks. In: *Proc. 14 Intl. Conf. on Discovery Science*, **6926** 107–121 (Springer, Berlin, 2011).

Gujarati, D. N. *Basic Econometrics* (McGraw-Hill, West Point, NY, 2002).

Haining, R. The moving average model for spatial interaction. *Transactions of the Institute of British Geographers* **3**, 202–25 (1978).

Hardisty, F.; Klippel, A. Analysing spatio-temporal autocorrelation with LISTA-VI. *International Journal of Geographic Information Science* **24**, 1515–1526 (2010).

Hasan, M. A.; Chaoji, V.; Salem, S.; Zaki, M. Link prediction using supervised learning. In: *Proc. SDM Wshp. on Link Analysis, Counterterrorism and Security*. 51–77 (SDM, Minneapolis, 2006).

Horne, D. Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers* **27**, 451–477 (1988).

Huang, B.; Wu, B.; Barry, M. Geographically and temporally weighted regression for modeling spatio-temporal variation in house prices. *International Journal of Geographical Information Sciences* **24**, 383–401 (2010).

Huang, H.; Martinez, F.; Mateu, J.; Nontes, F. Model comparison and selection for stationary space-time models. *Computational Statistical Data Analysis* **51**, 4577–4596 (2007).

Huang, Y.; Shekhar, S.; Xiong, H. Discovering colocation patterns from spatial data sets: A general approach. *IEEE Transactions on Knowledge Data Engineering* **16**, 1472–1485 (2004).

Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory* **14**, 55–63 (1968).

Hyndman, R. *Nonparametric Additive Regression Models for Binary Time Series* (Norwood Institute, Melbourne, Australia, 1999).

Jahani, S.; Bagherpour, M. A clustering algorithm for mobile ad hoc networks based on spatial autocorrelation. In: *Proc. Intl. 8th Symposium on Computer Networks and Distributed Systems*, 136–141 (CNDS, Tehran, 2011).

Jensen, D.; Neville, J. Linkage and autocorrelation cause feature selection bias in relational learning. In: *Proc. 9th Intl. Conf. on Machine Learning*. 259–266 (Morgan Kaufmann, 2002).

Jensen, D.; Neville, J.; Gallagher, B. Why collective inference improves relational classification. In: *Proc. 10th Intl. SIGKDD Conf. on Knowledge Discovery and Data Mining*. 593–598 (ACM, New York, NY, 2004).

Jiang, X.; Nariai, N.; Steffen, M.; Kasif, S.; Kolaczyk, E. Integration of relational and hierarchical network information for protein function prediction. *BMC Bioinformatics* **9**, 364–368 (2008).

Jin, F. Exploring Spatial Dependence - Starting from the Moran's I and the APLE Statistics. In: *Proc. 20th Annual Meetings of the Midwest Econometrics Group*. 367 (MEG, 2010).

Kamarianakis, Y.; Prastacos, P. Space-time modeling of traffic flow. *Journal of Computational Geoscience* **31**, 119–133 (2005).

Kwak, H.; Lee, C.; Park, H.; Moon, S. What is twitter, a social network or a news media? In: *Proc. 19th Intl. Conf. on World Wide Web*. 591–600 (ACM, New York, NY, 2010).

Langley, P. *Elements of Machine Learning* (Morgan Kaufmann, San Francisco, CA, 1996).

Lee, S. Developing a bivariate spatial association measure: An integration of Pearson's R and Moran's I. *Journal of Geographical Systems* **3**, 369–385 (2001).

Legendre, P. Spatial autocorrelation: Trouble or new paradigm? *Ecology* **74**, 1659–1673 (1993).

Legendre, P.; Dale, M. R. T.; Fortin, M.-J.; Gurevitch, J.; Hohn, M.; Myers, D. The consequences of spatial structure for the design and analysis of ecological field surveys. *Ecography* **25**, 601–615 (2002).

LeSage, J.; Pace, K. Spatial dependence in data mining. In: *Grossman, R.; Kamath, C.; Kegelmeyer, P.; Kumar, V.; Namburu, R. (eds.) Data Mining for Scientific and Engineering Applications*, 439–460 (Kluwer, Dordrecht, 2001).

Li, H.; Calder, C. A.; Cressie, N. Beyond Moran's I: Testing for spatial dependence based on the spatial autoregressive model. *Geographical Analysis* **39**, 357–375 (2007).

Li, X.; Claramunt, C. A spatial entropy-based decision tree for classification of geographical information. *Transactions in GIS* **10**, 451–467 (2006).

Li, X.; Kang, H.; Cao, J. Coordinated workload scheduling in hierarchical sensor networks for data fusion applications. In: *Proc. 4th Intl. Conf. on Mobile Adhoc and Sensor Systems*. 1–9 (IEEE Computer Society, Washington, DC, 2007).

Ma, C. Spatio-temporal variograms and covariance models. *Advanced Applied Probabilities* **3**, 706–725 (2005).

Macchia, L.; Ceci, M.; Malerba, D. Learning to rank by transferring knowledge across different time windows. In: *Proc. of Mining Complex Entities from Network and Biomedical Data, Intl. Workshop, MIND-2011*. (Springer, Berlin, 2011).

Macskassy, S.; Provost, F. Classification in networked data: A toolkit and a univariate case study. *Machine Learning* **8**, 935–983 (2007).

Macskassy, S. A. Improving learning in networked data by combining explicit and mined links. In: *Proc. 22th Intl. Conf. on Artificial Intelligence*. 590–595 (AAAI Press, 2007).

Malerba, D.; Appice, A.; Varlaro, A.; Lanza, A. Spatial clustering of structured objects. In: *Proc. 15th Intl. Conf. on Inductive Logic Programming*. 227–245 (Springer, Berlin, 2005).

Malerba, D.; Ceci, M.; Appice, A. Mining model trees from spatial data. In: *Proc. 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases*. 169–180 (Springer, Berlin, 2005).

Mantel, N. The detection of disease clustering and a generalized regression approach. *Cancer Research* **27**, 209–220 (1967).

McCarthy, J.; Minsky, M.; Rochester, N.; Shannon, C. A proposal for the Dartmouth summer research project on artificial intelligence. http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html (accessed: July, 2012).

McGovern, A.; Hiers, N. C.; Collier, M. W.; Gagne, D. J. G.; Brown, R. A. Spatiotemporal relational probability trees: An introduction. In: *Proc. 8th IEEE Intl. Conf. on Data Mining*. 935–940 (IEEE Computer Society, Washington, DC, 2008).

McPherson, M.; Smith-Lovin, L.; Cook, J. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* **27**, 415–444 (2001).

Mehta, M.; Agrawal, R.; Rissanen, J. SLIQ: A fast scalable classifier for data mining. In: *Proc. 5th Intl. Conf. Extending Database Technology*. 18–32 (EDBT, 1996).

Mewes, H.; Heumann, K.; Kaps, A.; Mayer, K.; Pfeiffer, F.; Stocker, S.; Frishman, D. MIPS: A database for protein sequences and complete genomes. *Nuclear Acids Research* **27**, 44–48 (1999).

Michalski, R. S.; Stepp, R. *Learning from Observation: Conceptual Clustering*, *Machine learning: an artificial intelligence approach*, 331–363 (Tioga, Palo Alto, CA, 2003).

Mitchell, T. *Machine learning* (McGraw-Hill, West Point, NY, 1997).

Mitsa, T. *Temporal Data Mining* (Chapman & Hall, London, 2010).

Moran, P. A. P. Notes on continuous dtochastic phenomena. *Biometrika* **37**, 17–23 (1950).

Neville, J.; Jensen, D. Relational dependency networks. *Journal of Machine Learning Research* **8**, 653–692 (2007).

Neville, J.; Simsek, O.; Jensen, D. Autocorrelation and relational learning: Challenges and opportunities. In: *Proc. Wshp. Statistical Relational Learning*. 33–50 (ICML, Montreal, 2004).

Newman, M. E. J. Assortative mixing in networks. *Physical Review Letters* **89**, 208701 (2002).

Newman, M. E. J.; Watts, D. *The structure and dynamics of networks* (Princeton University Press, Princeton, New Jersey, 2006).

Ohashi, O.; Torgo, L.; Ribeiro, R. P. Interval forecast of water quality parameters. In: *Proc. 19th European Conf. on Artificial Intelligence*, *Frontiers in Artificial Intelligence and Applications* **215**, 283–288 (IOS Press, Lisbon, 2010).

Orkin, M.; Drogin, R. *Vital Statistics* (McGraw-Hill, West Point, NY, 1990).

Ouali, M.; King, R. Cascaded multiple classifiers for secondary structure prediction. *Protein Science* **9**, 1162–76 (2000).

Pace, P.; Barry, R. Quick computation of regression with a spatially autoregressive dependent variable. *Geographical Analysis* **29**, 232–247 (1997).

Papoulis, A. *Probability, Random Variables, and Stochastic Processes* (Mc-Graw Hill, New York, 1984).

Park, J.; Barabasi, A. Distribution of node characteristics in complex networks. *Proc. of the National Academy of Sciences of the United States of America* **104**, 17916–17920 (2007).

Pfeifer, P.; Deutsch, S. A three-stage iterative procedure for space-time modelling. *Technometrics* **22**, 35–47 (1980).

Popescul, A.; Ungar, L. H. Statistical relational learning for link prediction. In: *IJCAI Wshp. on Learning Statistical Models from Relational Data* (IJCAI, Acapulco, 2003).

Qi, Y.; Noble, W. Protein interaction networks: Protein domain interaction and protein function prediction. In: Lu, H. H.; Scholkopf, B.; Zhao, H. (eds.) *Handbook of Computational Statistics: Statistical Bioinformatics* (Springer, Berlin, 2011).

Quinlan, R. J. *C4.5: Programs for Machine Learning* (Morgan Kaufmann, San Francisco, 1993).

Rahmani, H.; Blockeel, H.; Bender, A. Predicting the functions of proteins in protein-protein interaction networks from global information. *Journal of Machine Learning Research* **8**, 82–97 (2010).

Randic, M. On characterization of molecular attributes. *Acta Chimica Slovenica* **45**, 239–252 (1998).

Rice, J. A. *Mathematical Statistics and Data Analysis* (Duxbury Press, London, 2001).

Rinzivillo, S.; Turini, F. Classification in geographical information systems. In: *Proc. 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases*. 374–385 (Springer, Berlin, 2004).

Rinzivillo, S.; Turini, F. Knowledge discovery from spatial transactions. *Jornal of Intellegent Information Systems* **28**, 1–22 (2007).

Robinson, W. S. Ecological correlations and the behavior of individuals. *American Sociological Review* **15**, 351–357 (1950).

Ruepp, A.; Zollner, A.; Maier, D.; Albermann, K.; Hani, J.; Mokrejs, M.; Tetko, I.; Güldener, U.; Mannhaupt, G.; Münsterkötter, M.; Mewes, H. W. The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research* **32**, 5539–5545 (2004).

Russell, J.; Norvig, P. *Artificial Intelligence: A Modern Approach* (Pearson Education, London, 2003).

Sampson, P. D.; Guttorp, P. Nonparametric estimation of nonstationary dpatial covariance structure. *Journal of the American Statistical Association* **87**, 108–119 (1992).

Schietgat, L.; Vens, C.; Struyf, J.; Blockeel, H.; Kocev, D.; Džeroski, S. Predicting gene function using hierarchical multi–label decision tree ensembles. *BMC Bioinformatics* **11**, 103–126 (2010).

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; Eliassi-Rad, T. Collective classification in network data. *AI Magazine* **29**, 93–106 (2008).

Sharov, A. On-line lectures. http://home.comcast.net/ sharov/PopEcol/ (accessed: July, 2012).

Shi, L.; Lei, X.; Zhang, A. Protein complex detection with semi-supervised learning in protein interaction networks. *Proteome Science* **9**, 41–42 (2011).

Silla, C.; Freitas, A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* **1**, 1–42 (2011).

Steinhaeuser, K.; Chawla, V., Nitesh; Ganguly, A. R. Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. *Statistical Analysis and Data Mining* **4**, 497–511 (2011).

Stojanova, D. *Estimating Forest Properties from Remotely Sensed Data by using Machine Learning*. (Jožef Stefan Intl. Postgraduate School, Ljubljana, Slovenia 2009).

Stojanova, D.; Ceci, M.; Appice, A.; Džeroski, S. Network regression with predictive clustering trees. In: *ECML/PKDD (3)*. 333–348 (Springer, Berlin, 2011).

Stojanova, D.; Ceci, M.; Appice, A.; Džeroski, S. Network regression with predictive clustering trees. *Data Mining and Knowledge Discovery* **14**, 378–413 (2012).

Stojanova, D.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. Global and local spatial autocorrelation in predictive clustering trees. In: *Proc. 14th Intl. Conf. on Discovery Science*. 307–322 (Springer, Berlin, 2011).

Stojanova, D.; Ceci, M.; Malerba, D.; Džeroski, S. Using PPI networks in hierarchical multi-label classification trees for gene function prediction. In: *In 11th European Conference on Computational Biology: book of abstracts*. 236 (ECCB, Basel, 2012).

Stojanova, D.; Debeljak, M.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. *Models of the ecological hierarchy from molecules to the ecosphere*, *Dealing with spatial autocorrelation in gene flow modeling* (Elsevier, Amsterdam, 2012).

Stojanova, D.; Kobler, A.; Ogrinc, P.; Ženko, B.; Džeroski, S. Estimating the risk of fire outbreaks in the natural environment. *Data Mining and Knowledge Discovery* **2**, 411–442 (2012).

Struyf, J.; Džeroski, S. Constraint based induction of multi-objective regression trees. In: *Proc. 4th Intl. Wshp. on Knowledge Discovery in Inductive Databases*. 222–233 (Springer, Berlin, 2006).

Tobler, W. A computer movie simulating urban growth in the Detroit region. *Economic Geography* **46**, 234–240 (1970).

Vapnik, V. *Statistical Learning Theory* (Wiley, San Francisco, CA, 1998).

Varina, C.; Vidoni, P. Pairwise likelihood inference for ordinal categorical time series. *Computational Statistics and Data Analysis* **51**, 2365–2373 (2006).

Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; Blockeel, H. Decision trees for hierarchical multi-label classification. *Machine Learning* **73**, 185–214 (2008).

von Mering, C.; Krause, R.; Snel, B.; Cornell, M.; Oliver, S. G.; Fields, S.; Bork, P. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* **417**, 399–403 (2002).

Wang, J.; Cheng, T.; Heydecker, B.G.; Haworth, J. STARIMA for journey time prediction in London. In: *Proc. 5th Conf. on Math in Transp*. 93–106 (ISPRS, London, 2010).

Wang, Y.; Witten, I. Inducing model trees for continuous classes. In: Van Someren, M.; Widmer, G. (eds.) *Proc. 9th European Conf. on Machine Learning*. 128–137 (Springer, Prague, 1997).

Weng, J.; Lim, E.; Jiang, J.; He, Q. Twitterrank: finding topic-sensitive influential twitterers. In: *Proc. 3rd Intl. Conf. on Web Search and Data Mining*. 261–270 (ACM, New York, NY, 2010).

Witten, I.; Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques* (Morgan Kaufmann, San Francisco, 2005).

Yang, Q.; Wu, X. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology and Decision Making* **5**, 597–604 (2006).

Yue, Y.; Yeh, A. Spatiotemporal traffic-flow dependency and short-term traffic forecasting. *Environmental Planning* **35**, 762–771 (2008).

Zhang, P.; Huang, Y.; Shekhar, S.; Kumar, V. Exploiting spatial autocorrelation to efficiently process correlation-based similarity queries. In: *Proc. 8th Symp. on Advances in Spatial and Temporal Databases*. 449–468 (Springer, Berlin, 2003).

Zhao, M.; Li, X. An application of spatial decision tree for classification of air pollution index. In: *19th Intl. Conf. on Geoinformatics*. 1–6 (IEEE Computer Society, Washington, DC, 2011).

Zhu, J.; Huang, H.-C.; Wu, J. Modeling spatial-temporal binary data using markov random fields. *Journal of Agricultural Biological and Environmental Statistics* **10**, 212–225 (2005).

Zhu, J.; Zheng, Y.; Carroll, A.; Aukema, B. Autologistic regression analysis of spatial-temporal binary data via monte carlo maximum likelihood. *Journal of Agricultural, Biological, and Environmental Statistics* **13**, 84–98 (2008).

Zhu, S.; Chen, Z.; Chen, D. Bivariate moran spatial correlation analysis between zonal farm-land use and rural population distribution. In: Ju, W.; Zhao, S. (eds.) *Geoinformatics 2007: Remotely Sensed Data and Information* (ACM, New York, NY, 2007).

Zhu, X.; Ghahramani, Z.; Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In: *Proc. 20th Intl. Conf. on Machine Learning*. 912–919 (ICML, Omnipress, 2003).

Ziegler, C.; Mcnee, S.; Konstan, J.; Lausen, G. Improving recommendation lists through topic diversification. In: *Proc. 14th Intl. Conf. on World Wide Web*. 22–32 (ACM, New York, NY, 2005).

List of Figures

List of Tables

List of Algorithms

Appendices

Appendix A: CLUS user manual

The methods presented in this thesis are implemented in the CLUS system. Here, we provide a short user manual for CLUS. We begin by stating some general information about the system and some general settings. Next, we describe the settings that are specific to our method.

A.1 General information

CLUS is an open source machine learning system that implements the predictive clustering framework. It supports learning of predictive clustering trees and predictive clustering rules. The predictive clustering framework unifies unsupervised clustering and predictive modelling and allows for a natural extension to more complex prediction settings, such as predicting multiple target variables, hierarchical multi-label classification and prediction of time series.

CLUS is co-developed by the *Declarative Languages and Artificial Intelligence* group of the *Katholieke Universiteit Leuven*, Belgium, and the *Department of Knowledge Technologies* at the *Jožef Stefan Institute*, Ljubljana, Slovenia. CLUS is a free software (licensed under the *GPL*) and can be downloaded from `http://dtai.cs.kuleuven.be/clus/`.

CLUS uses (at least) two input files and these are named `filename.s` and `filename.arff`, with `filename` a name chosen by the user. The file `filename.s` contains the parameter settings for CLUS. The file `filename.arff` contains the training data, which need to be in ARRF data format for the tasks of predicting a single or multiple target variables (for HMC and clustering short time series, an extension of the ARFF format is used). For example, a file (that has the spatial coordinates as the first two attributes in the file) that can be used for the SCLUS or NCLUS algorithms is given below.

```
@relation forestfires
@attribute X numeric
@attribute Y numeric
@attribute month {mar,oct,aug,sep,apr,jun,jul,feb,jan,dec,may,nov}
@attribute day {fri,tue,sat,sun,mon,wed,thu}
@attribute FFMC numeric
@attribute DMC numeric
@attribute DC numeric
@attribute ISI numeric
@attribute temp numeric
@attribute RH numeric
@attribute wind numeric
@attribute rain numeric
@attribute area numeric
@data
7,5,mar,fri,86.2,26.2,94.3,5.1,8.2,51,6.7,0,0
```

```
7,4,oct,tue,90.6,35.4,669.1,6.7,18,33,0.9,0,0
7,4,oct,sat,90.6,43.7,686.9,6.7,14.6,33,1.3,0,0
8,6,mar,fri,91.7,33.3,77.5,9,8.3,97,4,0.2,0
```

The results of a CLUS run go to an output file `filename.out`. Figure 11.1 gives an overview of the input and output files of CLUS.

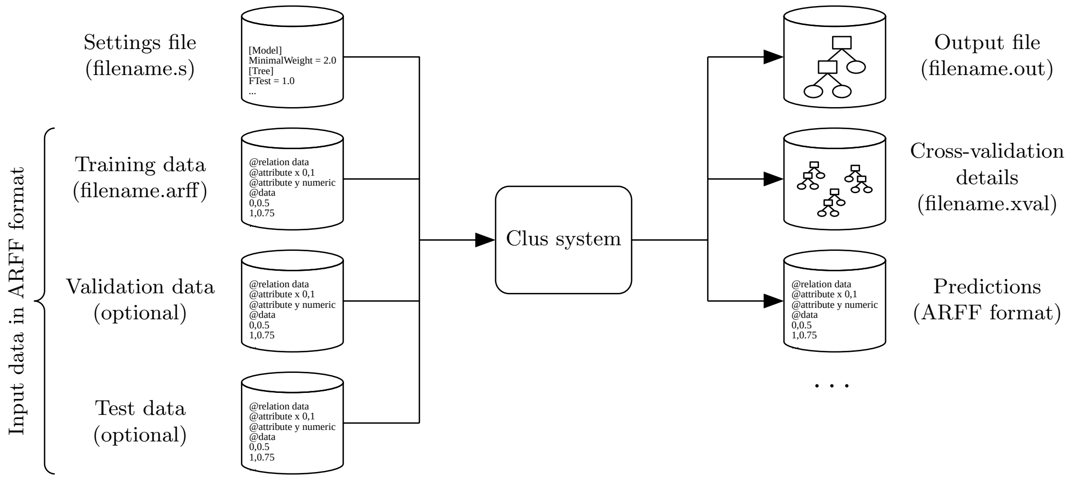


Figure 11.1: Input and output files of CLUS.

The algorithms implemented in the CLUS system are controlled by a number of parameters. These parameters are specified in a settings file. For most of the parameters there are default values that are used, unless a different value is specified by the user. The parameters in the settings file are grouped in the following sections:

- `General`: specifying the training, testing and validation datasets and number of folds for cross-validation,
- `Attributes`: lists the attributes that should be used as decsriptive, target etc.,
- `Tree`: type of heuristic score used for tree learning, tree pruning algorithm, pruning parameters etc.,
- `Rules`: type of heuristic score used for rule learning, coverage, optimization parameters etc.,
- `Output`: which errors and models should be output by CLUS, should the predictions of the model also be provided, etc.,
- `Constraints`: size, depth and error of the tree, syntactic constraints etc.,
- `TimeSeries`: the distance used for clustering time series and thye type of sampling when calculating the ICV heuristic,
- `Hierarchical`: parameters for hierarchical multi-label classification, such as type of hierarchy, weighting, threshold for making a prediction, etc.,
- `Beam`: parameters for beam search, such as number of trees in the beam, soft and hard size constraints, similarity constraint, etc.,

- `Ensembles`: parameters for constructing the ensembles, such as number of base predictive models, type of ensemble learning method, voting scheme, etc.

The sections in a settings file are described in detail in the document *CLUS: User's manual* which is available within the CLUS project documentation files. In the remainder of this section, we shortly present the `Attributes` and `Hierarchical` Sections. In the next section, we describe the relevant setting in the `Tree` section in more detail.

In the following example, we present a scenario for learning a PCT for predicting multiple targets. First, the attributes from the 25-th position in the dataset until the 28-th position are used as target attributes in the PCT construction. Second, the attributes from the 2-nd until 24-th position are used as descriptive attributes (the splits in the internal nodes of the tree will use these attributes). Next, the construction algorithm will ignore (i.e., disable) the attribute in the first position. Finally, the algorithm will perform normalization of the numeric attributes thus they equally contribute to the overall heuristic score.

```
[Attributes]
Descriptive = 2-24           % index of descriptive attributes
Target = 25-28               % index of target attributes
Disable = 1                  % Disables some attributes (e.g., "5,7-8")
Key = None                   % Sets the index of the key attribute
Weights = Normalize          % Normalize numeric attributes
```

The next example considers construction of a PCT for hierarchical multi-label classification. The classes are organized into a DAG hierarchy and the weighting factor (w_0) is set to 0.75. The weights for the classes with multiple parents are calculated by using average value over the weights of all parents. The Ftest pruning of the PCT should be done to optimize the pooled *AUPRC* performance measure.

```
[Hierarchical]
Type = DAG                           % Tree or DAG hierarchy?
WType = ExpAvgParentWeight            % aggregation of class weights
WParam = 0.75                         % parameter w_0
OptimizeErrorMeasure = PooledAUPRC   % FTest optimization strategy
ClassificationThreshold = 40      % threshold for "positive"
```

A.2 Settings for Autocorrelated PCTs

In the CLUS system, the predictive clustering trees can be constructed by using the standard top-down induction of decision trees strategy (a default setting in CLUS). Here, we shortly describe the parameters (discussed in Section 3) that control the induction of Autocorrelated PCTs and give an excerpt of a settings file.

The `Tree` section from the settings file contains the following options:

- `Heuristic=` *VarianceReductionGIS* : sets the correct heuristic for building Autocorrelated PCTs.

- `SpatialMatrix =` o : o is an element of {`Binary, Modified, Euclidean, Gaussian`} and defines the weights in the autocorrelation statistic, by default it is set to `Binary`.

  - `Binary`: Binary weights.
  - `Modified`: Modified weights.

  - `Euclidean`: Euclidean weights.
  - `Gaussian`: Gaussian weights.

- `SpatialMeasure` = o : o is an element of {`GlobalMoran`, `GlobalGeary`, `GlobalGetis`, `LocalMoran`, `LocalGeary`, `LocalGetis`, `StandardizedGetis`, `EquvalentI`, `IwithNeighbours`, `EquvalentIwithNeighbours`, `GlobalMoranDistance`, `GlobalGearyDistance`, `CI`, `MultiVariateMoranI`, `CwithNeighbours`, `Lee`, `MultiIwithNeighbours`, `CIwithNeighbours`, `LeewithNeighbours`, `Pearson`, `CIDistance`, `DH`, `EquvalentIDistance`, `PearsonDistance`, `EquvalentG`, `EquvalentGDistance`, `EquvalentPDistance`} and defines the autocorrelation statistic, by default it is set to `None`.
  - `GlobalMoran`: uses Global Moran (Legendre, 1993).
  - `GlobalGeary`: uses Global Geary (Legendre, 1993).
  - `GlobalGetis`: uses Global Getis (Getis and Ord, 1992).
  - `LocalMoran`: uses Local Moran (Anselin, 1988).
  - `LocalGeary`: uses Local Geary (Anselin, 1988).
  - `LocalGetis`: uses Local Getis (Getis and Ord, 1992).
  - `StandardizedGetis`: uses Local Standardized Getis (Getis and Ord, 1992).
  - `EquvalentI`: uses an incremental formula for Global Moran.
  - `IwithNeighbours`: uses Global Moran statistics by setting the number of neighbors instead of the bandwidth.
  - `EquvalentIwithNeighbours`: uses an incremental formula for Global Moran by setting the number of neighbors instead of the bandwidth.
  - `GlobalMoranDistance`: uses Global Moran (Legendre, 1993) with a separate distance file statistics.
  - `GlobalGearyDistance`: uses Global Geary (Legendre, 1993) with a separate distance file statistics.
  - `CI`: uses Connectivity Index (CI) for Graph Data (Randic, 1998).
  - `MultiVariateMoranI`: uses Cross Moran (Zhu et al, 2007).
  - `CwithNeighbours`: Global Geary by setting the number of neighbors instead of the bandwidth.
  - `Lee`: uses Bivariate Lee's measure (Lee, 2001).
  - `MultiIwithNeighbours`: uses Cross Moran (Zhu et al, 2007) by setting the number of neighbors instead of the bandwidth.
  - `CIwithNeighbours`:uses Connectivity Index (CI) for Graph Data (Randic, 1998) by setting the number of neighbors instead of the bandwidth.
  - `LeewithNeighbours`:uses Bivariate Lee's measure (Lee, 2001) by setting the number of neighbors instead of the bandwidth.
  - `Pearson`:uses Pearson correlation coefficient (Angin and Neville, 2008).
  - `CIDistance`:uses Connectivity Index (CI) for Graph Data (Randic, 1998) with a separate distance file.

  - DH:uses Dyadicity and Heterophilicity (Park and Barabasi, 2007) for Graph Data with a separate distance file.
  - EquvalentIDistance:uses an incremental formula for Global Moran with a separate distance file.
  - PearsonDistance:uses Pearson correlation coefficient (Angin and Neville, 2008) with a separate distance file.
  - EquvalentG:uses an incremental formula for Global Geary.
  - EquvalentGDistance:uses an incremental formula for Global Geary with a separate distance file.
  - EquvalentPDistance:uses an incremental formula for Pearson correlation coefficient (Angin and Neville, 2008) with a separate distance file.

- Bandwidth = n : sets the b parameter that controls the bandwidth used in the calculation of autocorrelation. It ranges in [0, 1], it is expressed in % of the maximum bandwidth calculated from the dataset and its default value is 0.0.
- Alpha = n : sets the α parameter that gives the relative influence of variance reduction and autocorrelation. It ranges in [0, 1] and its default value is 1.0 i.e., do not consider autocorrelation and works just like a PCT.
- Longlat = Yes/No : If Yes, the spatial coordinates are given in longitude-latitude format and need to be converted into a cartesian coordinate system. Since this is rarely the case, the default setting is No.
- NumNeightbours = n : sets the number of neighbors n when used with SpatialMeasures that require it. Its default value is 0.0.

We also need to specify the attributes involved in the calculation of the autocorrelation. This will be done using the following option in the Attributes section:

- GIS = o : o is an element of {None, n}, where n specifies the autocorrelation related attribute(s), by default it is set to None.

At the end, we give an example of a setting file that can be used with the data example given above.

```
[Data]
File = ff.arff

[Attributes]
Target = 3
Descriptive =4-12
GIS=1-2 % Spatial coordinates

[Model]
MinimalWeight = 20.5

[Tree]
Heuristic = VarianceReductionGIS
SpatialMatrix = Euclidian
SpatialMeasure =GlobalMoran
```

```
Bandwidth=0.5
Alpha = 0.5
```

Appendix B: Bibliography

List of publications related to this dissertation.

B.1 Publications related to this thesis

B.1.1 Journal papers:

Stojanova, D.; Ceci, M.; Appice, A.; Džeroski, S. Network regression with predictive clustering trees. *Data Mining and Knowledge Discovery* **14**, 411–442 (2012).

Stojanova, D.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. Dealing with spatial autocorrelation when learning predictive clustering trees. *Ecological Informatics* [in press] http://dx.doi.org/10.1016/j.ecoinf.2012.10.006 (2012).

Stojanova, D.; Ceci, M.; Malerba, D.; Džeroski, S. Using PPI networks in hierarchical multi-label classification trees for gene function prediction. *BMC BioInformatics* [under review] (2012).

B.1.2 Book chapters

Stojanova, D.; Debeljak, M.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. Dealing with spatial autocorrelation in gene flow modeling. In Jordan, F. and Jorgensen, S. (ed.), *Models of the ecological hierarchy from molecules to the ecosphere*, 35–49 (Elsevier, Amsterdam, 2012).

B.1.3 Conference and workshop papers:

Stojanova, D.; Ceci, M.; Malerba, D.; Džeroski, S. Using PPI networks in hierarchical multi-label classification trees for gene function prediction. In: *11th European Conference on Computational Biology: book of abstracts*. 236 (ECCB, Basel, 2012).

Stojanova, D.; Ceci, M.; Appice, A.; Džeroski, S. Network regression with predictive clustering trees. In: *ECML/PKDD (3)*. 333–348 (Springer-Verlag, Berlin, Heidelberg, 2011).

Stojanova, D.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. Global and local spatial autocorrelation in predictive clustering trees. In: *Proc. 14th Intl. Conf. on Discovery Science*. 307–322 (Springer-Verlag, Berlin, Heidelberg, 2011).

Stojanova, D.; Debeljak, M.; Ceci, M.; Appice, A.; Malerba, D.; Džeroski, S. Dealing with spatial autocorrelation in gene flow modeling. In: *7th European Conference on Ecological Modelling. Ecological hierarchy from the genes to the biosphere: book of abstracts*. 97 (Springer-Verlag, Berlin, Heidelberg, 2011).

B.2 Other publications

B.2.1 Journal papers:

Debeljak, M.; Trajanov, A.; Stojanova, D.; Leprince, F.; Džeroski, S. Using relational decision trees to model out-crossing rates in a multi-field setting. *Ecological Modelling* **245**, 75–83 (2012).

Stojanova, D.; Kobler, A.; Ogrinc, P.; Ženko, B.; Džeroski, S. Estimating the risk of Fire outbreaks in the natural environment. *Data Mining and Knowledge Discovery* **2**, 411–442 (2012).

Železnik P.; Stojanova, D.; Kraigher, H. Organization of fine root data obtained from minirhizotrons and ingrowth soil cores (how to construct an operational database using MS Access). *Research Reports of Forestry and Wood Science and Technology* **95**, 37–44 (2011).

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

Taškova, K.; Stojanova, D.; Bohanec, M.; Džeroski, S. A qualitative decision-support model for evaluating researchers. Informatica **31**, 479–486 (2007).

B.2.2 Master Thesis:

Stojanova, D. Estimating Forest Properties from Remotely Sensed Data by using Machine Learning. Master's thesis (Jožef Stefan Intl. Postgraduate School, Ljubljana, 2009).

B.1.3 Other publications:

Debeljak, M.; Trajanov, A.; Stojanova, D.; Leprince, F.; Džeroski, S. Using relational decision trees to model out-crossing rates in a multi-field setting. In: *7th European Conference on Ecological Modelling. Ecological hierarchy from the genes to the biosphere: book of abstracts.* 53 (Springer-Verlag, Berlin, Heidelberg, 2011).

Stojanova, D.; Kobler, A.; Džeroski, S. A qualitative decision support system for evaluating forest models : effect of forest management practice. In: *Proceeding of Small scale forestry in a changing world: opportunities and challenges and the role of extension and technology transfer.* 717–725 (Gozdarski inštitut Slovenije, Ljubljana, 2010).

Stojanova, D.; Kobler, A.; Džeroski, S. A qualitative decision support system for evaluating forest models : effect of forest management practice. In: *Proceeding of Small scale forestry in a changing world: opportunities and challenges and the role of extension and technology transfer : book of abstracts.* 64 (Gozdarski inštitut Slovenije, Ljubljana, 2010).

Stojanova, D.; Levanic, Tom. Web-based GIS system : a case study from Slovenia. In: *Proceedings of International scientific Conference on Spatial Data Infrastructures.* 88–95 (National Infrastructures, Skopje, 2010).

Železnik, P.; Stojanova, D.; Kraigher, H. Handling of 4 years data from minirhizotrons and ingrowth soil cores with MS Access : how to construct a usable database. In: *Proceedings of 5th International Symposium on physiological processes in roots of woody plants.* 7–14 (Victoria University, Victoria, 2010).

Levanic, T.; Stojanova, D. Uporaba spletnega geografsko informacijskega sistema za spremljanje populacij divjadi v Sloveniji. In: *2nd Meeting of Slovenian Wildlife Community*. 56 (ERICo, Velenje, 2010).

Janža, M.; Komac, M.; Kobler, A.; Stojanova, D.; Oštir, K.; Marsetič, A.; Džeroski, S.; Gosar, A. Metodologija ocene višine in gostote vegetacijskega pokrova z daljinsko zaznanimi podatki in možnosti njene uporabe v geologiji. In: *Proceedings of 19th Meeting of Slovenian Geologists*. 58–61 (Univerza v Ljubljani, Ljubljana, 2009).

Levanic, T.; Stojanova, D. Osrednji slovenski lovsko informacijski sistem. *Javna gozdarska služba v letu 2008*. 15 (Gozdarski inštitut Slovenije, Ljubljana, 2008).

Stojanova, D.; Taškova, K.; Bohanec, M.; Džeroski, S. In: *Proceedings of the 9th International Multi-Conference Information Society*. 60–63 (Institut "Jožef Stefan", Ljubljana, 2007).

Stojanova, D.; Panov P.; Kobler, A.; Džeroski, S.; Taškova, K. Learning to Predict Forest Fires with Different Data Mining Techniques. In: *Proceedings of the 9th International Multi-Conference Information Society*. 255–258 (Institut "Jožef Stefan", Ljubljana, 2006).

Taškova, K.; Panov P.; Kobler, A.; Džeroski, S.; Stojanova, D. Predicting Forest Stand Properties from Satellite Images using Different Data Mining Techniques. In: *Proceedings of the 9th International Multi-Conference Information Society*. 259–262 (Institut "Jožef Stefan", Ljubljana, 2006).

Kocev, D.; Stojanova, D. Optic Flow Estimation Using Lucas-Kanade Algorithm. In: *Proceedings of VII National Conference With International Participation*. 146–151 (Elektrotehnički fakultet, Skopje, 2005).

Appendix C: Biography

Daniela Stojanova was born on 22 June 1982 in Strumica, Macedonia. She demonstrated interest in math and computer science and was a member of the math section at Jane Sandanski High School in Strumica. As a high school student she won several awards at regional and state math competitions. In 2000, she enrolled in the Faculty of Electrical Engineering and Information Technologies, “Ss. Cyril and Methodius” University in Skopje, Macedonia. In 2005, she was finished the BSc program in the area of Computer Engineering, Information Science and Automatics. As a student she published some of her research work at the ETAI Macedonian National Conference.

She received a Master’s Degree in New media and E-science at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia in 2009. She defended her MSc thesis titled “Estimating forest properties from remotely sensed data by using machine learning”, under the supervision of Prof. Dr. Sašo Džeroski and co-supervision of MSc. Andrej Kobler. In 2009, she enrolled in the Doctoral Studies under the third-level PhD program entitled “Information and Communication Technologies” at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Her research interests focus on the learning of predictive models from non i.i.d (when samples are drawn independently and identically from an unknown data generating distribution) data i.e., when data samples are correlated between each other, as it is ubiquitous in many real-world problems. In particular, she explores the consideration of the autocorrelation phenomenon when learning predictive models, as well as their applications to ecological, environmental, network and bioinformatics problems. Her work is covered within the projects of the Department of Knowledge Technologies at the Jožef Stefan Institute, Ljubljana, Slovenia where she works as research assistant.

She has applied the developed algorithms to problems in ecological modeling, environmental studies, spatial-temporal analysis, remote sensing applications and bioinformatics. She has published her work in several premier international journals including “Data Mining and Knowledge Discovery” and “Ecological informatics”. She has attended and given talks at several conferences and workshops: MLSB2012, ECML/PKDD2011, DS2011, ECEM2011, IS2007, ETAI 2005, both in the areas of data mining and the respective domains of the case studies.