This is to certify that the
dissertation entitled

Spatial and Temporal Data Mining with
Applications to Earth Science Data

presented by

Haibin Cheng

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science

Yan Pay Ning

Major Professor's Signature

11/20/2008

Date

SPATIAL AND TEMPORAL DATA MINING WITH APPLICATIONS TO EARTH SCIENCE DATA

By

Haibin Cheng

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2008

ABSTRACT

SPATIAL AND TEMPORAL DATA MINING WITH APPLICATIONS TO EARTH SCIENCE DATA

By

Haibin Cheng

Recent progress on computer and sensor technology has generated huge amounts of spatial and temporal data in the Earth Science domain including climate observations, land cover time series records, ground level pollution measurements, etc. Combined with historical climate records and predictions from ecosystem models, it offers new opportunities for understanding how the earth is changing, for determining what factors cause these changes and for forecasting future changes. Spatial and temporal data mining provides innovative solutions for mining Earth Science data by incorporating spatial and temporal dependencies into standard data mining techniques. Although there has been substantial research in spatial and temporal data mining, there are still many technical issues that need to be addressed. This includes issues such as processing massive high resolution data, reducing the effect of noise, fusing data from heterogeneous sources, etc. We develop a class of efficient and robust spatial and temporal data mining algorithms in this thesis to overcome these challenges. First, we develop an integrated localized prediction framework based on Support Vector Machine to incorporate spatial and temporal dependencies. Efficient algorithms are also proposed to reduce its computational overhead. Second, we study the error accumulation problem in multi-step time series prediction and develop a novel semi-supervised multivariate time series prediction algorithm for long term forecasting. A covariance alignment method is also proposed to reduce the inconsistencies between historical and future climate data when applying the algorithm to future climate projection problems. Third, we propose a graph-based framework to detect and categorize different types of anomalies in multivariate time series data. We applied

the framework to the problem of detecting and characterizing ecosystem disturbances in Earth Science data. While the spatial and temporal data mining techniques proposed in this thesis have been applied to many problems in the Earth Science domain, they are also applicable to spatial and temporal data in other application domains such as traffic analysis, image processing, network monitoring, etc.

To

My Family

For their love and support

ACKNOWLEDGMENTS

It is my pleasure to thank all the wonderful people that contributed to this thesis in many aspects.

I wish to express my warm and sincere thanks to my advisor Dr. Pang-Ning Tan. This dissertation would not have been possible without his continuous support, inspiration and valuable suggestions. He is the one that I can always count on to discuss the details of a problem. I truly appreciate his patience and encouragement that helped me go through my Ph.D. study step by step and develop my research skills.

I am grateful to my thesis committee members - Dr. Rong Jin, Dr. Li Xiao and Dr. Ashton Shortridge. Dr. Rong Jin advised me and helped me greatly in my research. His extensive discussions abount machine learning theory and algorithms have been very helpful for this study. Dr. Li Xiao and Dr. Ashton Shortridge provided me with many important suggestions on the application of my study in different domains.

I wish to thank all the faculties and scientists that has helped me in numerous aspects of my research work - Dr. Christopher Potter and Dr. Steven Klooster from NASA research center, Dr. Jon Sticklen and Dr. William F. Punch from Department of Computer Science and Engineering at Michigan State University, Dr. Haifeng Chen and Dr. Guofei Jiang from NEC research lab, Dr. Jianchan Mao and Dr. Ruofei Zhang from Yahoo Lab.

I am grateful to all the staff in the Department of Computer Science and Engineering at Michigan State University, for helping the department to run smoothly and for assisting me in many different ways. Ms. Linda Moore and Ms. Norma Teague deserve special mention. Many thanks to Ms. Kim Thompson for proof-reading this entire document.

I would like to thank many current or previous student colleagues in Data Mining Research Group - Jerry Scripps, Kapila Moonesinghe, Samah Fodeh, Feilong Chen, Hamed Valizadegan, Jing Gao, and Jun Wang for creating a fun and stimulating research environment. Many thanks to all the other members of the Links Lab too.

I wish to thank all my friends in the U.S. and China. Although I cannot name them all here, I appreciate the support they provided and the wonderful time we had during this period.

Finally, I would like to extend my long loving thanks to my parents and my wife for their continuous care, support, and encouragement. Their love accompanies me wherever I go. My thanks also go to my grandma, my brother, my aunts, and my uncles.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Advances in sensor technology, online mapping services (such as Google Earth, Yahoo Maps, and Microsoft Virtual Earth), and location-aware computing have generated massive amounts of spatial and temporal data that can be potentially mined to uncover valuable knowledge about a particular domain. Spatial data refers to a set of observations characterized by a spatial location in addition to other attributes. Examples of spatial data includes crime data, housing price data, and vegetation cover data collected for various locations. Temporal data corresponds to an ordered series of observations, where each observation is associated with a time stamp. Examples of temporal data include climate time series, sensor data streams, and telecommunication alarm sequences. Spatio-temporal data refers to data with both spatial and temporal components. Examples of spatio-temporal data include cyclone trajectories, mobile user tracking data, and dynamic brain images, etc. The thesis will investigate some of the outstanding issues in mining spatial and temporal data with applications to the Earth Science domain.

1.1 Earth Science Applications

With the growing concerns about the impact of climate change, climate study has become an increasingly important area of research. Figure 1.1 shows a substantial increasing trend of global temperature since the mid-twentieth century [181], which is believed to be partly caused by a steady increase of the global atmospheric concentrations of CO_2, CH_4, and N_2O due to the fast growth of industrial activities [113]. According to a recent assessment by the Intergovernmental

Panel on Climate Change (IPCC) the projected temperature increase by the end of the century is expected to range between 1.1°C and 6.4°C. Due to global warming, numerous long term changes in climate have been observed at continental, regional, and ocean basin scales. Some of these include widespread changes in temperature, precipitation amounts, ocean salinity, wind patterns, and the frequency and severity of extreme weather conditions, which lead to adverse consequences such as the melting of polar ice caps, rising of sea level, diminishing of fresh water supplies, decreasing of rainfall, increasing number of ecosystem disturbance events, and so on. Most of these events have a direct or indirect impact on human life, e.g. cities close to the coast may disappear due to rising sea level, growth of crops is strongly affected by the diminishing of fresh water. Thus developing new techniques to explore the cause and impact of climate change has become an important research topic for scientists in various disciplines.

To help better understand and project future climate changes, scientists have collected a huge amount of data by utilizing the latest sensor technology and computing systems to measure various aspects of land surface, atmosphere, oceans, and other components of the Earth system, which provides new opportunities for answering some important questions in Earth Science, e.g. (1) how

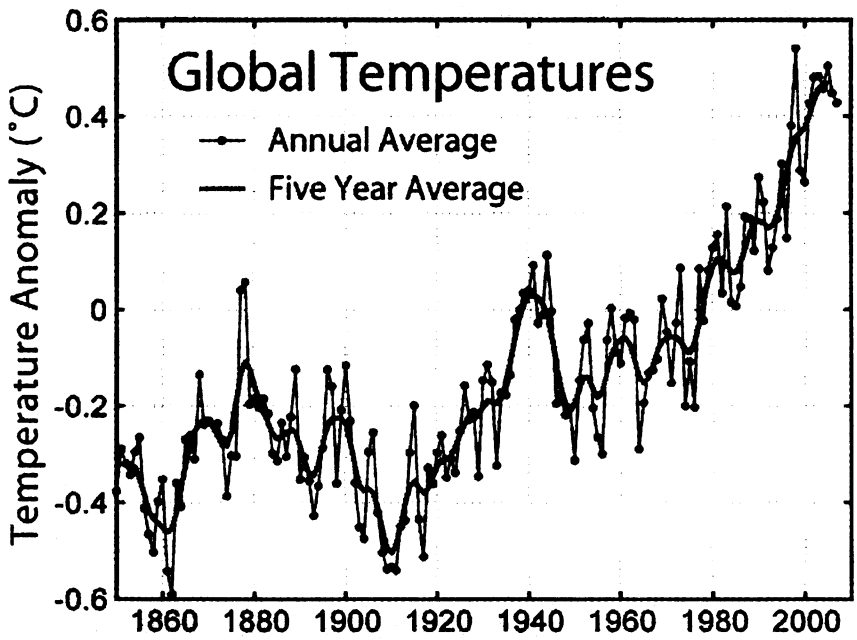


Figure 1.1. An illustration of increasing global mean temperature from 1850 to 2000 (relative to mean temperature from 1961 – 1990 (from Global Warming Art [181]).

Table 1.1. Some Examples of Earth Science Data and their Applications.

Example of Earth Science data	Applications
Climate observations (e.g. precipitation)	Projection of future climate scenarios
Remote sensing data (e.g. FPAR)	Detection of ecosystem disturbance
Earth environment measurements (e.g. Ozone)	Prediction of the ozone alarm
Trajectory data (e.g. cyclone tracks)	Clustering of extratropical cyclones

the earth is changing, (2) what factors cause these changes, and (3) how to forecast future changes. Data mining and knowledge discovery techniques can aid this effort by discovering patterns that capture complex interactions among ocean temperature, air pressure, surface meteorology, and terrestrial carbon flux. Table 1.1 shows several examples of Earth Science data and their potential applications.

- Projection of future climate scenarios such as forecasting the frequency and amount of rainfall or the number of frost days is beneficial to climate-dependent industries such as agriculture, tourism, recreational and commercial fishing, etc [105].

- Discovery of anomalous patterns from historical vegetation cover data can be used to detect ecosystem disturbances such as forest fires, droughts, and hurricanes [170].

- Localized prediction with spatial and temporal dependencies could be used to predict the burned area of forests for an unknown region [54], or to classify the days in which the ground-level ozone reaches a dangerous level [224].

- Clustering of the trajectories of tropical cyclone tracks is important for improving the ability to forecast the location and timing of hurricane landfall [86], which in turn, may help to mitigate the hurricane damage.

1.2 Challenges

Despite its wide range of applications, mining Earth Science data is not a trivial problem because of its intrinsic spatial and temporal nature, which makes the direct application of standard data mining methods and algorithms problematic. As an example, the prediction of climate for a region depends strongly on the local topographic conditions of the field. Traditional data mining

algorithms fail to give special consideration to inherent spatial and temporal dependencies. This has led to the development of numerous spatial and temporal data mining algorithms for different tasks such as prediction, clustering, anomaly detection, change detection, etc. Here we formally define spatial and temporal data mining as:

Definition 1. (Spatial and Temporal Data mining) *Spatial and temporal data mining is a task of discovering useful knowledge in data set with spatial, temporal, or spatio-temporal dependencies.*

In addition to the spatial and temporal dependencies, there are still many other technical challenges that need to be addressed, particularly in the context of mining Earth Science data. First, the deployment of high-resolution earth imaging satellites has generated huge amounts of Earth Science data in very high spatial resolution. For instance, satellite-derived vegetation cover data such as FPAR (Fraction of Photosynthetically Active Radiation Absorbed by Vegetation), are available at a spatial resolution of 500m $\times$ 500m, which makes it impractical to perform exploratory data analysis in real-time fashion. Second, Earth Science data may involve many variables, among which strong correlations may exist. For example, the change of global vegetation cover could be caused by the disturbance events in climate variables such as temperature, precipitation, sea level pressure, and so on. Third, Earth Science data is mostly noisy. For example, satellite observations are distorted because of sensor noise or cloud blocking. Fourth, Earth Science data may come from different sources. For instance, there is climate simulation data from GCM (Global Climate Modeling) or RCM (Regional Climate Modeling) in addition to the true observation data from sensors. Fifth, acquiring early historical data in Earth Science is not only difficulty but also impossible due to the unavailability of the sensor technology in previous years. Last but not least, Earth Science data is not evenly distributed across different time periods or spatial locations. For example, forest fires happen more frequently in regions with dry weather or in years with less rainfall. Solving these issues by inventing new spatial and temporal data mining techniques and algorithms is critical to fully utilize the Earth Science data in helping understand or predict the potential impact of climate change.

1.3 Contributions

In this thesis, we investigate these problems mainly in three aspects:

- Localized prediction with spatial and temporal dependencies.
- Long term time series forecasting.
- Anomaly detection, anomaly characterization, and visualized exploration.

1.3.1 Localized Prediction with Spatial and Temporal Dependencies

Prediction is one of the most popular tasks in data mining and machine learning. It typically consists of two aspects: classification and regression. The task of classification is concerned with the prediction of predefined discrete labels, while the task of regression aims to predict continuous target values. Prediction with spatial and temporal dependencies is an important problem in mining Earth Science data with many applications. For example, environmental scientists are interested in predicting the the burned area of a potential forest fire [54] for a given region such that prevention measures can be better organized to reduce the damages. The spatial and temporal dependencies in Earth Science data may indicate a non-even distribution of the data across different spatial locations and temporal timestamps, which has posed a great challenge for building an accurate prediction function. As an example, Figure 1.2 shows the forest fire distribution by month in Cuba from 1981 – 1996 [93]. Apparently, forest fires happen more frequently from February to May in Cuba.

To address the challenges, one must first decide whether to use global learning or local learning approaches. Global learning tends to build a sophisticated single model using the global characteristics of the training data. Auto-regression and Kriging are two popular global learning techniques widely used for spatial and time series prediction [29][96][103]. However, they are both least square methods for linear prediction. Recently a large number of algorithms have been proposed to solve the prediction problems in data mining and machine learning literature, which consist of various objectives and are capable of modeling complex nonlinear decision surfaces [221][32][179][130]. Among them, one of the most effective algorithms is the nonlinear Support Vector Machine (**SVM**), which learns a prediction function by maximizing the margin in classification or enforcing function flatness in regression, and employs a kernel to model nonlinear decision functions. One of the main difficulties found in global learning methodologies is the model selection problem. More precisely, one needs to select a suitable model and its parameters

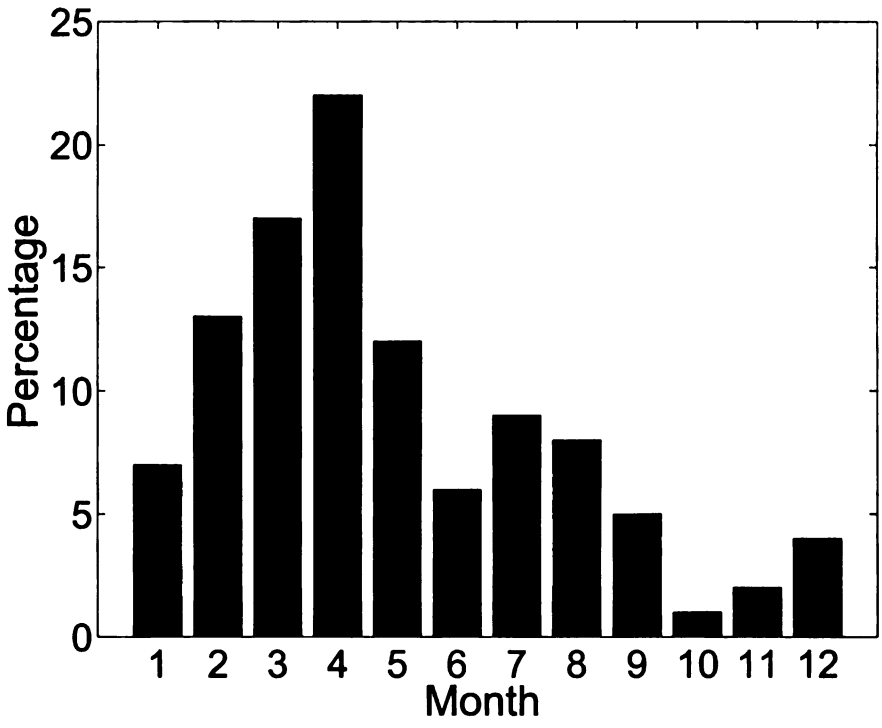


Figure 1.2. Forest Fire distribution by month in Cuba 1981 – 1996 [93].

in order to represent the observed data. For example, a nonlinear SVM must employ sophisticated kernel functions to predict data sets with complex decision surfaces. Determining the right parameters for such kernel functions is not only challenging [84], the resulting models are also susceptible to overfitting due to their large VC dimensions [32]. This problem is even more severe when applying nonlinear SVM directly to Earth Science prediction problems due to its disregard of the spatial and temporal dependencies. Instead of fitting a global model to the entire training data, another strategy is to use local learning by constructing multiple models using the local characteristics of the data. Recent research has demonstrated that local learning is superior to global learning especially on data sets that are not evenly distributed [28][223][89][132]. A well-known classification technique that employs such a strategy is the K-nearest neighbor (KNN) classifier [55]. The advantage of using KNN is that it does not require making a biased assumption about characteristics of the data nor about the decision surfaces [11]. Nevertheless, KNN performs not very well in high dimensional data and because of its lazy learning strategy, classifying test examples can be computationally expensive.

We first developed a framework known as *Localized Support Vector Machine* (**LSVM**), consist-

ing of *Localized Support Vector Classification* (**LSVC**) and *Localized Support Vector Regression* (**LSVR**), which leverages the strengths of both Support Vector Machine (**SVM**) and K-nearest neighbor (**KNN**). Instead of fitting a sophisticated kernel function to build a global model for the entire training data, LSVM builds multiple linear SVM models by considering the training examples located in the vicinity of each test example. We empirically show that such a strategy would lead to significant performance improvement over global nonlinear SVM and avoids the thorny issue of model selection.

However, since LSVM builds a unique model for each test example, it is impractical when the number of test examples is large. To overcome this limitation, we develop an efficient implementation of the algorithm, known as *Profile Support Vector Machine* (**PSVM**), consisting of both *Profile Support Vector Classification* (**PSVC**) and *Profile Support Vector Regression* (**PSVR**). The intuition behind this algorithm is based on the premise that many test examples with similar neighboring training examples tend to share a common set of support vectors. The PSVM algorithm is therefore designed to exploit this property by employing a supervised clustering algorithm to partition the training data into clusters and training a local SVM model for each cluster. After constructing the local models, a test example will be classified by finding its nearest cluster centroid and invoking the corresponding local SVM model. Our experimental results show that this strategy enables PSVM to maintain the high performance of LSVM without its costly computational overhead.

We also extend the LSVM and PSVM framework to Earth Science data, which consists of strong spatial and temporal dependencies. Recent research on local learning for spatial and temporal prediction has demonstrated the significantly improved performance of building multiple local machine learning models compared with global machine algorithms [89][132]. The neighborhood information of examples is defined in terms of the temporal and spatial attributes in LSVM and PSVM. The experimental results on real-word Earth Science data sets demonstrate a significantly improvement of the performance by the proposed LSVM and PSVM algorithms compared with global nonlinear SVM as well as KNN.

1.3.2 Long Term Time Series Forecasting

Time series prediction aims to predict the future values for a target variable of interest based on its historical observations or other related information. It has been an active area of research for a long time with many applications, e.g. weather forecasting [75] [40], stock market analysis [215], network monitoring [27], and transportation planning [107][158], etc. Most of these problems focused on single step or short term prediction. Recently, there has been an increasing interest on long term time series forecasting for strategic planning and decision making. For example, Earth scientists are interested in projecting the future climate to assess their potential impact on the ecosystem and society. Figure 1.3 shows such an example of projecting the global temperature for the next 100 years by simulating different climate scenarios [182], e.g. CCSR/NIES, CCCma, and NCAR PCM. Despite its wide applications, long term time series forecasting is a challenging problem because of several issues. First, an extensive amount of historical time series data is required in order to build a reliable predictive model for long term time series prediction. However, collecting historic data in the early days may be not only difficult but also impossible due to the unavailability of facilities at that time. Second, most existing long term time series forecasting methods are extended from the single step prediction method by repeatedly invoking a model that makes its prediction one step at a time. However, since the model uses predicted values from the past to infer future values, these approaches may lead to the error accumulation problem, where the errors made from one prediction step will be propagated to the next prediction step. Alternative methods such as independent prediction also have their own problems. Third, potential concept drifting in many domains makes the long term time series forecasting even more difficult. For example, to predict rainfall in the next 100 years, one should certainly consider the increasing trend of global temperature around the world as illustrated in Figure 1.3. Last but not least, using data from different sources to aid the long term time series forecasting is a challenging problem since they may follow different data distributions. For instance, GCM data in climate projection as shown in Figure 1.3 is generated by computer systems driven by a set of emissions scenarios, which may assume greenhouse gas concentrations that are different from those observations in the historical climate record.

To address these issues, we employ a multivariate time series prediction method in this thesis to

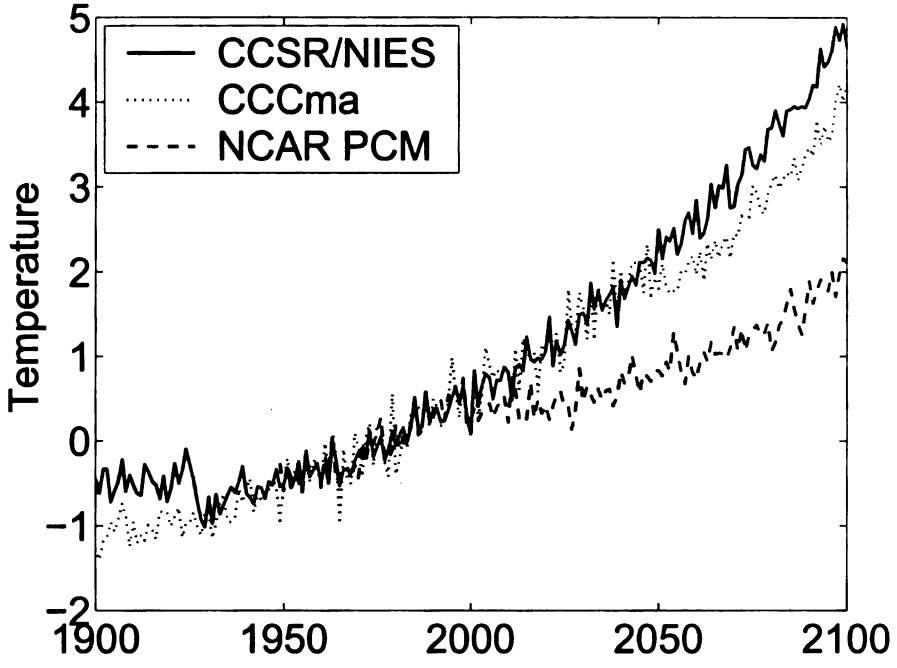


Figure 1.3. Climate model predictions for future global temperature (^oC) until 2100 under the SRES A2 emissions scenario relative to global average temperatures in 2000 (from Global Warming Art [182]).

preform long term time series forecasting instead of predicting the future values of a time series based on its historical values alone. The time series for a set of predictor variables are fitted against the time series for the response variable. This strategy avoids the requirement for long periods of historical data; however, it is contingent upon the availability of future values of the predictor variables. Such values can be obtained from computer-driven simulation models in many domains. For example, in climate modeling, outputs from global climate models (GCMs) in future periods [75] are often used as predictor variables to determine the climate of a local region. Potential concept drifting in the future has been incorporated into the GCM data by simulating different climate scenarios. However, current approaches utilize the simulation data mostly in a supervised learning setting, where a predictive model trained on past observations is used to estimate the future values. Such an approach fails to take advantage of information about the future data during model building. A semi-supervised learning framework is proposed for long term time series forecasting based on Hidden Markov Model Regression (HMMR). Our approach builds an initial HMMR model from past observations and incorporates future data to iteratively refine the model. Since the initial predictions for some of the future data may not be reliable, we need to

ensure that they do not adversely affect the revised model. We developed an approach to overcome this problem by assigning weights to instances of the future data based on the consistency between their global and local predictions. This approach also helps to ensure smoothness of the target function [227]. We demonstrated the efficacy of our approach using data sets from a variety of application domains.

When applying the semi-supervised HMMR to climate modeling problems, we must consider the potential inconsistencies between the training and future data since they come from different sources. Previous work on semi-supervised classification has shown that combining labeled and unlabeled data with different distributions may degrade the performance of a classifier [199]. We encountered a similar problem when applying the semi-supervised HMMR method to climate projection. A data calibration technique is developed to transform the data sets in a way that aligns their covariance structure while preserving most of the neighborhood information. Experimental results for modeling climate at a number of locations in Canada showed that semi-supervised HMMR with data calibration outperforms the conventional (supervised) HMMR method.

1.3.3 Anomaly Detection, Characterization, and Visualization

Anomaly detection is a valuable data mining tool for discovering patterns significantly different than the rest of a data set. Recently, there has been an increasing interest on time series anomaly detection in many applications, e.g. to detect unusual events such as ecosystem disturbances in Earth Science data. While numerous algorithms have been developed for detecting anomalies [121, 141, 16, 122, 212, 220] in time series data, most of them are applicable only to univariate time series.

The detection of anomalies in multivariate time series is more challenging due to several reasons. First, it is difficult to establish a concise definition of an anomaly. Analogous to univariate time series, some anomalies may correspond to abnormally high (or low) values or unusual subsequences (discord [121]) in one or more time series. In addition, the multivariate anomalies may correspond to unexpected changes in the relationships among a set of variables [42]. For example, the time series for vegetation cover at mid-latitude locations in the United States typically varies in a 12-month cycle, peaking during the warm summer months and dropping to its minimum during the cold winter. Ecosystem disturbances such as wildfire and drought can be potentially detected

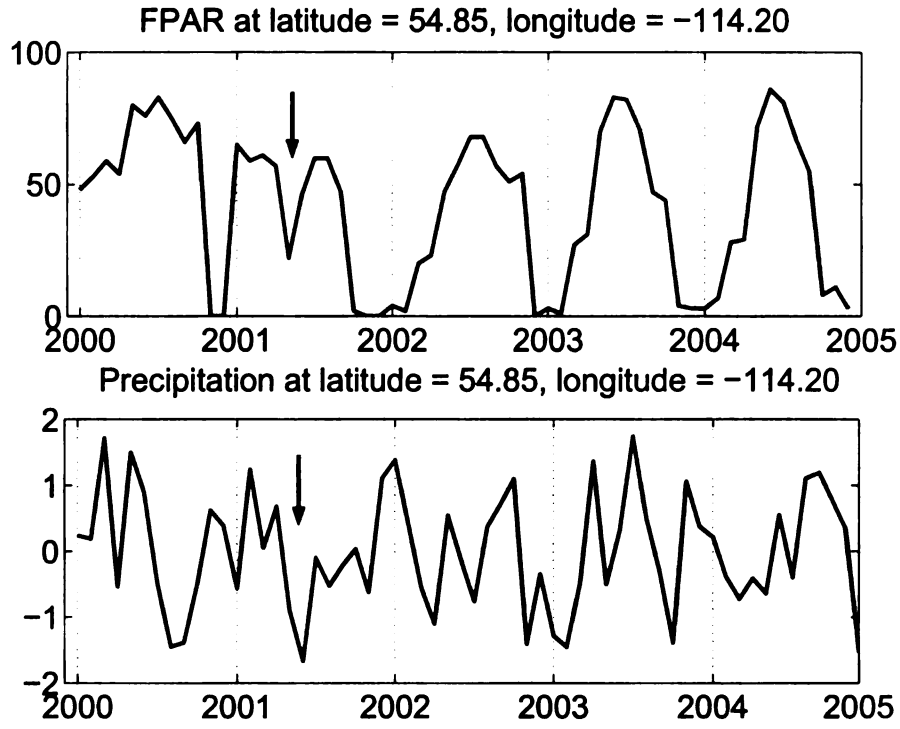


Figure 1.4. FPAR and Precipitation time series at $(latitude = 54.85N, longitude = 114.20W)$ with possible anomalies.

based on the unusually low values of vegetation cover observed in the summer. We consider such anomalies as "local" (as opposed to global anomalies) since their values are abnormally low only when compared to the average values during the summer months. Second, the performance of a multivariate anomaly detection algorithm is highly susceptible to the presence of noise in one or more time series. Therefore, a multivariate anomaly detection algorithm must be robust to noisy measurements in order to improve its detection rate and false alarm rate.

Multivariate time series anomaly can also be used to characterize the types of anomalies found in a target time series. For example, Earth scientists are interested in detecting wildfires by monitoring the anomalies found in vegetation cover data derived from NASA's Earth observing satellites. However, analyzing the vegetation cover data alone is insufficient to distinguish between man-made wildfires from those induced by extreme climate events (such as lightning strikes from severe thunderstorms). Therefore, a key challenge is to combine the time series from other related variables (e.g., temperature and precipitation) to help explain the anomalies found in a target time series (vegetation cover).

Furthermore, previous work on ecosystem disturbance detection mainly focused on low resolution FPAR data, which can only detect sustained disturbance events that alter the structure of vegetation cover significantly for a sufficiently large region and result in an overall decline in the average annual vegetation levels. Smaller scale disturbances could have been missed when using the coarse resolution AVHRR data. With the recent availability of high resolution vegetation cover data from the moderate resolution imaging spectroradiometer (**MODIS**) instrument on board the NASA TERRA satellite, smaller scale disturbances can be potentially uncovered. However, because the size of the data has grown by 4 orders of magnitude, applying the disturbance event detection algorithm to such massive data sets is computationally expensive. The problem is further exacerbated by the fact that a disturbance event detection algorithm (or anomaly detection algorithm, in general) requires specification of one or more thresholds to determine whether a detected event should be flagged as a real disturbance. The thresholds are determined by users through a trial and error process during the exploratory data analysis phase. Because of the large amount of data that must be processed, performing exploratory data analysis on the high resolution MODIS data in a real-time fashion is computationally infeasible. The massive size of the data also produces more events for scientists to validate. This necessitates the development of innovative data mining approaches that can assist Earth scientists in real-time exploration of global scale eco-climatic data.

To address these challenges, we propose to develop a robust graph-based algorithm for detecting anomalies in multivariate time series data. Our algorithm captures the dependence relationships among variables in the multivariate time series using a kernel matrix alignment approach. The alignment helps to eliminate noise and retains only anomalies in the target time series that can be explained by anomalies observed in other time series. The time series anomalies are detected by performing a random walk traversal on the graph induced by the aligned kernel matrix. Since a kernel matrix can be constructed from each time point or subsequence in the time series, our algorithm is flexible enough to handle various types of time series anomalies including subsequence-based and local anomalies. The effectiveness of our algorithm is demonstrated by using a number of synthetic and real data sets. We have also conducted a case study to show the ability of our algorithm to detect and characterize ecosystem disturbances in Earth Science data.

A visualizing system is further developed for detecting anomalies in Earth Science data inter-

actively. We illustrate the use of clustering to group together regions with similar incidents of ecosystem disturbances. This approach provides an unsupervised way to categorize the events and helps reduce the number of events that need to be validated. We also present a multi-level indexing scheme based on clustering to aid the discovery and visual exploration of ecosystem disturbances. We show how the indexing scheme can be used to enable users to quickly focus on regions of interest during the exploratory data analysis phase. While clustering-based methods have been proposed for spatio-temporal indexing and for data reduction purposes [34, 19], none of them are specifically designed for detecting anomalies (such as ecosystem disturbances) in spatio-temporal data. In this work, we evaluate the effectiveness of using clustering-based methods for exploratory analysis of ecosystem disturbances. Our disturbance event detection and clustering algorithms have been integrated into an interactive system we developed. The system enables scientists to explore the data in real-time fashion and to visually inspect clusters of locations where similar types of disturbance events were observed.

1.4 Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 provides a literature survey for spatial and temporal data mining. Chapter 3 presents a localized prediction algorithm based on the Support Vector Machine. Chapter 4 investigates the problems in long term time series forecasting and proposes a semi-supervised method based on Hidden Markov Regression to improve the prediction accuracy. Chapter 5 discusses a graph-based time series anomaly detection algorithm, kernel alignment based anomaly characterization technique and a visualization system for exploratory analysis of large scale spatial and temporal data. Finally, Chapter 6 concludes with a summary of the thesis and suggestions for future research. Note that the techniques developed in this study are potentially applicable to other spatial and temporal domains such as sensor networks, intelligent transportation systems, image processing, and biogeography.

CHAPTER 2

Spatial and Temporal Data Mining

Spatial and temporal data mining is a subfield of data mining and knowledge discovery that refers to the extraction of interesting patterns in spatial and temporal data, which consists of spatial, temporal, or spatio-temporal dependencies as illustrated in Section 2.1. Research on spatial and temporal data mining has received significant attention in the past decade by researchers in the statistics, data mining, and machine learning communities. As illustrated in Figure 2.1, we provide a literature survey on spatial and temporal data mining particularly in three areas: spatial data mining as introduced in Section 2.2, temporal data mining as discussed in Section 2.3, and spatio-temporal data mining as presented in Section 2.4. Each area consists of several mining tasks including prediction, clustering, association rule mining, anomaly detection, etc. For spatio-temporal data mining, an additional survey on visualization is also presented, which is useful for studies on large scale Earth Science data. Finally, the scope of research directions for this thesis is given in Section 2.5.

2.1 Spatial and Temporal Data

Here we give a simple notation of the spatial and temporal data used throughout this thesis in terms of spatial data, temporal data, and spatio-temporal data.

Definition 2. (Temporal Data) *Temporal data corresponds to an ordered series of observations, where each observation is associated with a timestamp.*

Note that we do not consider sequences ordered by other indexes such as text, protein sequence,

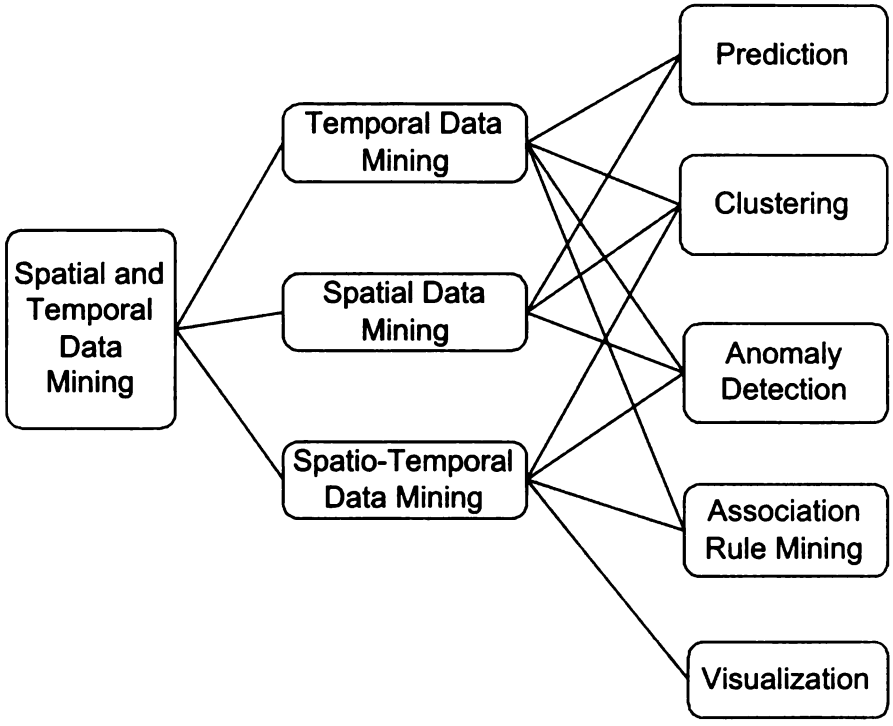


Figure 2.1. Survey on Spatial and Temporal Data Mining.

etc. as temporal data although there has been such definition [135] in the literature. An example of temporal data is formulated as:

$$\left[(t_1, \boldsymbol{x}_1), \ (t_2, \boldsymbol{x}_2), \ \ldots, \ (t_l, \boldsymbol{x}_l) \right]^\top$$

where t_i is the timestamp for example $\boldsymbol{x}_i$ and l is the size of the temporal data. One popular class of temporal data is the *time series* data with numerical values ordered based on timestamps. Time series analysis is important for many applications such as weather forecasting, stock market analysis, sensor data analysis, etc. Another category of temporal data is a sequence of *events* with categorical values ordered based on their timestamps, e.g. telecommunication alarms/events, web server logs, online transaction logs, etc. In this thesis, we are mostly concerned with time series, which are widely available in Earth Science data. Time series data consists of univariate time series and multivariate time series according to the number of available variables. A univariate time series is an ordered sequence of real values for a single variable. When the temporal information t_i represents nothing more than an order from 1 to n as $t_i = i$, the univariate time series data can be simply represented as:

$$\left[x_1, \ x_2, \ \ldots, \ x_l \right]^\top$$

A multivariate time series is a vector of time series involving more than one variable. To accommodate different applications, we use two notations here to represent a multivariate time series with l data examples and p variables. One formulates a multivariate time series as a vector of p variables:

$$\left[\boldsymbol{X}_1, \ \boldsymbol{X}_2, \ \cdots, \ \boldsymbol{X}_p \right]$$

where $\boldsymbol{X}_i$ represents i_{th} time series variable with length l; another formulates a multivariate time series as a vector of l data examples:

$$\left[\boldsymbol{x}_1, \ \boldsymbol{x}_2, \ \cdots, \ \boldsymbol{x}_l \right]^\top$$

where $\boldsymbol{x}_i$ represents i_{th} example with p attributes.

Definition 3. (Spatial Data) *Spatial data corresponds to data characterized by a spatial location and other non-spatial attributes.*

Here we use o_i to denote location feature for the example $\boldsymbol{x}_i$. A spatial data set with l data examples is represented as:

$$\left[(o_1, \boldsymbol{x}_1), \ (o_2, \boldsymbol{x}_2), \ \cdots, \ (o_l, \boldsymbol{x}_l) \right]^\top$$

A popular representation of the spatial features in Earth Science data are the latitude and longitude $o_i = (lat_i, lon_i)$, which determine the locations of the various places on planet earth.

Definition 4. (Spatio-temporal Data) *Spatio-temporal data corresponds to a set of observations with both spatial location features and timestamps.*

Using similar notations introduced above for spatial data and temporal data, a spatio-temporal data is represented as:

$$\left[(t_1, o_1, \boldsymbol{x}_1), \ (t_2, o_2, \boldsymbol{x}_2), \ \cdots, \ (t_l, o_l, \boldsymbol{x}_l) \right]^\top$$

where (t_i, o_i) constitutes a space-time index for each data example $\boldsymbol{x}_i$. Examples of spatio-temporal data include vegetation cover or climate time series recored in different regions, objects moving across different space and time, spatial events or phenomena evolving over time, etc.

2.2 Spatial Data Mining

Spatial data mining is the process of discovering interesting and previously unknown, but potentially useful patterns from large spatial datasets. Mining spatial data is very useful for many applications, ranging from remote sensing, to geographical information systems (GIS), computer cartography, environmental assessment and planning, etc. There has been a long history of spatial data mining in the statistics, geography, data mining, and machine learning communities [80][78][2][190][126]. We identify some representative work in the literature for a number of different tasks such as spatial prediction, spatial clustering, spatial association rule mining, and spatial anomaly detection.

2.2.1 Spatial Prediction

Spatial prediction builds a predictive model from the data in observed locations and predicts the target values for unobserved locations. Previous work on spatial prediction mostly focused on regression but is applicable to classification problems. Spatial prediction has a rich literature in geostatistical analysis. Among them, one of the simplest is the spatial auto-regression model (SAR) [96], which is similar to auto-regression in time series prediction. It builds a global linear regression model to model the spatial dependencies from the entire observed locations and uses it to predict the unobserved location. Parallel formulation of the SAR model is also developed by Kazar et al. [117] to speed up this method. Markov random fields is another popular model that incorporates spatial autocorrelation in the Bayes framework for spatial prediction, which is believed to make less restrictive assumptions than the SAR model as discussed by Shekhar et al. [189]. As spatial data are often influenced by various local phenomena, the idea of using local models was also developed for spatial prediction problems. Geographically weighted regression (GWR) [83] is a local version of spatial regression that estimates parameters taking into account the spatial proximity between training and testing examples. Kriging is a group of geostatistical techniques to interpolate the value of a random field at an unobserved location from values at observed locations [58][68], which is also known as Gaussian process regression in the statistics community. Both the Kriging and SAR methods are *linear* methods based on least square estimation. Recently, there has been increasing interest in applying more sophisticated *nonlinear*

machine learning models to spatial prediction, e.g. Neural Network [161][130] provides models of relationships between inputs and outputs through highly interconnected "neurons", Support Vector Machine [89] learns a nonlinear decision function by minimizing the prediction error while maximizing the margin, Decision Trees [222] algorithm builds a tree-like structure of predictors, etc. However, building global machine learning models is not sufficient, especially for data sets with spatial and temporal dependencies. There has been some work on building local machine learning models for such data, e.g. Gilardi et al. proposed to build a local Support Vector Regression or local Neural Network for each testing example from the training examples in its spatial neighborhood. However, their method is very slow and has the difficulty in selecting a suitable number of nearest neighbors.

2.2.2 Spatial Clustering

Clustering of spatial data aims to group together observations with similar spatial and non-spatial attributes to identify the underlying structure of the data. Spatial clustering has been used to discover "hot spots" in geostatististics [94], e.g. mapping of hot spots for criminal activities or diseases in a city. Spatial clustering algorithms can be classified into three categories: *partitioning based* method, *hierarchical based* method, and *density based* method. Partition based methods try to divide the data into a given number of clusters. The K-means [114], K-medoids (PAM)[116] and EM algorithms [66] are three examples of partition based clustering methods. K-means algorithm finds the clusters by estimating the centroid to represent each cluster and assigning each data point to its closest centroid iteratively. The K-medoids algorithm is similar to K-means except that the data point closest to the centeroid is utilized to represent the cluster. The EM algorithm assumes that each cluster follows a mixture of Gaussian distribution and data points are assigned to the clusters with probability. These methods perform poorly on large scale spatial data. To overcome this problem, a sampling based clustering method CLARA (clustering large application) is proposed by Kaufman et al. [116] to run the K-mediod algorithm on a randomly sampled subset of the data. Although CLARA is more efficient, there is a problem when the true mediods are missed during sampling. An improved algorithm CLARANS [155] is proposed by introducing some randomness in each sampling step and taking advantage of some efficient indexing technique such as R-tree [81]. These methods can only find clusters with spherical shapes

and they require the number of clusters to be predefined. Hierarchical clustering algorithms such as BIRCH [225] and CHAMELEON [115] build a dendrogram to split the data set into smaller subsets. Two strategies are usually used: agglomerative algorithms build the tree in the bottom-up manner, while divisive algorithms build the tree in the up-bottom manner. Hierarchical clustering is powerful in discovering arbitrarily-shaped clusters but is expensive for high dimensional data. Density based methods such as DBSCAN [79] assume a data point in high density belongs to a cluster. GDBSCAN [183] generalizes it to cluster data points with both spatial attributes and non-spatial attributes. DBSCAN and GDBSCAN are useful to find arbitrarily-shaped clusters and to distinguish outliers from clusters. However, they are not very efficient for high dimensional data and need a predefined parameter such as radius. Bethi et al. proposed another density based method called CLIQUE [20], which divides the space into grid cells whose density is computed and grid cells with high density are merged to form clusters. It is more efficient but less effective than DBSCAN for high dimensional data.

2.2.3 Spatial Association Rule Mining

The task of *spatial association rule mining* is to discover spatial relationships between objects or events in one location with those in other locations [127]. For example, a rule like "most big cities in US are close to the coast" is a spatial association rule. Spatial association rule mining has been applied to many applications such as geographic information system [127], image analysis [109], census analysis [142], etc. The earliest algorithm for mining spatial association is to apply standard associate rule mining algorithm such as Apriori [3] method to spatial data points whose spatial relationships satisfy some user specified task [127]. As a result, it separates spatial computations from the process of generating spatial association rules. Recently, there has been increasing interest on mining co-location patterns [149][150][186][226], which are spatial associations dedicated to capturing neighboring relationships. Morimoto [149] and Shekhar et al. [186] proposed an algorithm to discover clique co-location patterns, which requires all involved objects to be close to each other. Finding co-location patterns are very time consuming. To solve this problem, Zhang et al. [226] implement a partition-based optimization strategy to find co-location patterns more efficiently; however, it requires all spatial objects to be data points. Xiong et al. [218] propose co-location mining for extended spatial objects such as polygons and

line-strings. Most of these work on spatial association rule mining consider short-range dependencies only. Mining spatial association rules describing long-range dependencies remain an open research problem.

2.2.4 Spatial Anomaly Detection

Spatial anomaly detection, also known as spatial outlier detection, identifies spatial locations whose non-spatial attributes are significantly different from the values of their neighborhoods even though they may not be significantly different from the entire population [187][139]. Spatial anomaly detection approaches were first studied in geostatistics, which can be generally grouped into two categories: *graphical approaches* and *quantitative tests*. Graphical methods are based on the mapping of spatial data that highlights spatial outliers, e.g. variogram clouds and pocket plots [98][201]. These methods require a domain expert to detect the spatial anomalies visually. Quantitative methods use statistical tests to identify spatial outliers from the remainder of data such as scatterplot proposed by Han et al. [95] and Moran scatterplot approaches developed by Anselin et al. [9]. Data mining methods have also been applied to detect spatial outliers recently. Shekhar et al. [188] introduced a method for detecting spatial outliers in graph data sets based on the distribution of the difference between an attribute value and the average attribute value of its neighbors. Lu et al. [139] also proposed two methods, one is non-iterative algorithm that uses median as the neighborhood function, another is an iterative algorithm that identifies only one outlier in each iteration and modifies its attribute value to prevent negative impact on subsequent iterations. The false alarm rate of these algorithms are still relatively high. Reducing the false alarm rate of spatial outlier detection needs further investigation.

2.3 Temporal Data Mining

Temporal data mining is concerned with extracting interesting and previously unknown, but potentially useful patterns from large scale temporal data sets. There has been increasing interest on mining temporal data during the past several years [135]. Most previous work can be categorized into several major tasks including classification, clustering, time series forecasting, anomaly detection, and temporal association rule mining. Here we present a brief review of the techniques

and algorithms in terms of different temporal data mining tasks.

2.3.1 Time Series Forecasting

The task of *time series forecasting* aims to predict the future values of a time series based on its own historical values or information from other related variables. Extensive studies have been conducted for the problem of time series forecasting in the statistics, data mining, and machine learning communities. These approaches typically use a sliding window to create a training set from the historical data to build a predictive model. The auto-regression (AR) model [29] is the earliest and simplest predictive model, which is designed to predict the future values of a stationary time series [29][41][101] based on a linear combination of historical values using least square linear regression model [118]. A more complex linear model such as ARIMA [147] is also invented for the modeling of non-stationary time series, which has been widely applied to financial stock analysis. Another popular alternative model to the AR model for modeling non-stationary time series is the piece-wise linear regression model, which breaks the time series into a series of stationary pieces and learns a separate linear regression model. A similar approach is also found in the Hidden Markov Regression [85] for time series prediction, which assumes a stochastic process on a set of states, each associated with a separate linear regression models. In addition to those *linear* models, *nonlinear* regression models are also used to model the time series data. For example, Recurrent Neural Network [90], Support Vector Regression[157], Gaussian Process [17], and Decision Trees [138] have been applied to model nonlinear time series for future forecasting. However, these works mostly focused on *univariate* times series prediction. These works mostly utilize a recursive prediction methodology, where the outputs from the last prediction are used as inputs for the next prediction, and thus fails for long term time series prediction. Although an advanced model was proposed by Herrera et al. in [104] to improve the long term time series forecasting by using a noiseless one-step-ahead prediction for recursive training, their method did not address the error accumulation caused by model bias and variance. Chapados et al.[39] developed augmented functional time series representation and a forecasting algorithm based on Gaussian processes to avoid point estimation. However, the prediction window of their algorithm is still relatively short and the error caused by the biased estimation of the posterior distribution will still be accumulated for long term time series prediction.

Multivariate time series forecasting [138] predicts the future values of a time series by utilizing not only its own historical values but also information from other related time series. Methods for multivariate time series prediction are usually extended from univariate time series prediction by including related time series as predictor variables. For example, Popescu et al. [169] proposed to use independent component analysis to decompose the multivariate time series into independent components and perform univariate forecasting for each component separately whose predicted results are recombined at the final stage.

2.3.2 Classification of Temporal Data

Temporal data classification assumes that each sequence or time series belongs to one of the predefined classes or labels and the objective is to automatically determine the class or label for a given sequence. Temporal data classification has been investigated widely in many applications, for example, speech recognition tries to transcribe speech signals into their corresponding textual representation [174][92]; gesture recognition classifies the human body motions into messages people want to express [61] or some sign language [192]; handwritten words recognition tends to recognize the words from a sequence of pixel columns and segments extracted form the image of handwritten texts [43][151][211]. The methodologies used in these works can be categorized into *pattern based* and *model based.* Pattern based methods align the given sequence to a collection of prototype sequences and find the closest match using certain sequence similarity measures such as dynamic time warping (DTW) [123][50][50] and longest common subsequence (LCSS) [62]. Model based methods assign labels to a sequence according to a classification model that is learned from a set of training sequences. Examples of model based methods for temporal data classification include Hidden Markov Model [15][203], Neural Networks [216][195], Support Vector Machine [72], etc.

2.3.3 Clustering of Temporal Data

Clustering of temporal data aims to group together a collection of temporal data based on their similarity. Clustering is important for mining temporal data since it provides a way to automatically reveal the intrinsic structure of the data and helps people to better understand the data. For

example, it would be interesting to cluster gene expression time series [76], stock time series [33] or trajectories of moving objects [64], whose trends and variations behave similarly. Many different models has been proposed to cluster temporal data. For example, mixtures of ARMA model [219] or Hidden Markov Model [7] have been utilized for the clustering of time series data; Sebastiani et al. proposed the Bayes model [185] and Law et al. proposed a rival penalized competitive learning method [133] to cluster sequence data. A number of different similarity measures such as dynamic time warping (DTW) [123][50][50], longest common subsequence (LCSS) [62], etc. have also been proposed to cluster time series data using standard clustering algorithms such as K-means [156], density based [64], and hierarchical clustering algorithms [180].

2.3.4 Temporal Association Rule Mining (Frequent Pattern)

Temporal association rule mining is designed for discovering frequent sequential patterns [208][4][88], which comprise temporal order information between elements, in the collections of temporal transactions. The Apriori [3] method has been adapted for mining temporal association rules by counting the fraction of all the sequences, in which rules are contained, in the temporal database. One extension of frequent pattern mining in temporal data is frequent episode mining, which is to extract subsequences that repeat a sufficiently number of times in the given long time series or sequence [144][10] [13][136][10]. Temporal episode mining has many useful applications such as analyzing alarm streams in a telecommunication network [144] or mining of data from Walmart transactions [10]. Furthermore, time dependence is also incorporated explicitly into the frequent sequential pattern mining or frequent episodes mining. For example, Lee et al. [38] proposed to incorporate the duration of the events into frequent sequential pattern mining and Ozden et al. [160] introduced cyclic association rules that hold with a fixed periodicity along the entire length of the sequence of time intervals.

2.3.5 Time Series Anomaly Detection

Time series anomaly detection aims to discover data points or subsequences that are significantly different than the other parts of the time series. Numerous algorithms have been proposed to discover anomalous patterns in the univariate time series. For example, Keogh et al. [121] proposed

an algorithm for mining time series that contain subsequences with largest nearest neighbor distance. Mahoney and Chan [141] employed a path and box feature trajectory approach to model the time series. Anomalous subsequences are detected based on their deviation from the trajectory path. Bay et al. [16] utilized local AR models to transform the data into its corresponding parameter space and anomalies were detected based on distances computed in the parameter space. Frequency-based methods [122, 212] are designed to discover unusual subsequences who happen less frequently than other patterns. Immunology-based method [63] uses given normal pattern to detect anomaly patterns as templates and uses these templates to find unusual pattern in a new time series. Probability based method [220] calculates the anomaly score for any given data as its deviation from the learned probabilistic AR (Auto-regression) model from the historical data. These methods mostly focused on anomaly detection in *univariate* time series.

Recently, there has been considerable interest in developing anomaly detection algorithms for *multivariate* time series. One category of the methods utilize time series projection [87] and independent component analysis [213] to convert the multivariate time series into univariate time series. One potential limitation of these methods is the loss of information incurred when projecting the data into 1-dimensional space. Despite the rich literature on time series anomaly detection, there are few works on characterizing the discovered anomalies. Lakhina et al. [131] proposed an approach to characterize anomalies in network traffic flows. Their work, however, simply performs a separate anomaly detection for each variable without focusing on any specific target variable. Potter et al. [171] used association analysis to relate extreme climate events with ecosystem disturbances. However, it is achieved by a separate postprocessing step.

2.4 Spatio-Temporal Data Mining

Spatio-temporal data mining is an emerging research area dedicated to the development and application of novel computational techniques for the analysis of large spatio-temporal data sets, e.g. climate time series recorded for different locations in Earth Science data or moving objects tracked at different locations and time. Compared with spatial data mining or temporal data mining, studies on spatio-temporal data mining are limited. Recent studies on spatio-temporal data mining mainly focus on two strategies. One is to decompose the spatio-temporal data mining into two

independent sub-problems: spatial data mining and temporal data mining [146][21][167][168]. Another strategy is to extend techniques in spatial or temporal data mining directly to the mining of spatio-temporal data [162][145][77][145]. Similar to previous sections on spatial data mining and temporal data mining, we present a brief review on the latest work on spatio-temporal data mining methods in terms of spatio-temporal prediction, spatio-temporal clustering, spatio-temporal association rule mining, and spatio-temporal anomaly detection. An additional survey for the indexing and visualization of large scale spatio-temporal data is also provided.

2.4.1 Spatio-Temporal Prediction

Spatio-temporal prediction is to build a predictive model from the observed data with space-time information and predict the target variable at an unobserved location and time. Most spatio-temporal prediction techniques are extended from spatial prediction or time series forecasting models. One example is the spatio-temporal auto-regression model proposed by Kelley et al. [162] for real estate price prediction, which formulates the auto-regression model with both spatial and temporal neighborhood information. Markov Random Field [145] and Kriging [77] have also been extended from spatial prediction to spatio-temporal prediction as well. Another family of spatio-temporal prediction methods decomposes the problem into two separate steps. For example, a two-phase spatio-temporal auto-regression model was proposed by Pokrajac et al. [167][168], which models the spatial data first and then adjusts the prediction by incorporating autoregressive modeling of residuals in time.

2.4.2 Spatio-Temporal Clustering

Spatio-temporal clustering tries to group together the spatio-temporal data points with similar behavior, which is often used to extract interesting groups of trajectories. Most spatio-temporal clustering algorithms are simply extended from the spatial clustering algorithms discussed in Section 2.2.2. For instance, the partition based method K-means is utilized to cluster the climate variables indexed by both space and time to extract time varying climate regions [106]; the density based method DBSCAN is applied to cluster trajectories of moving objects [145] and the hierarchical agglomerative clustering algorithm is adapted to group together similar trajectories

[152]. Other spatio-temporal clustering techniques such as spatial scan has also been developed by Kulldorff et al. [129] for epidemiological data to extract spatio-temporal cylinders (e.g. spatial circular shapes that remain still for some time interval) where the rate of disease cases is higher than outside the cylinder.

2.4.3 Spatio-Temporal Association Rule Mining

Spatio-temporal association rule mining aims to discover how objects move between regions and over time. The most straightforward technique for spatio-temporal association rule mining is conducted by mapping spatio-temporal data into a transaction table and to use the standard Apriori approach to extract spatio-temporal rules [146]. A more concrete definition of spatio-temporal rule is first given by Tao et al. [198], which explores the association between objects or events in different regions over time, e.g. the rule $(i, \nabla t, p) \Longrightarrow (j)$ means if an abject appears in region i at time t, it will appear in region j at time $t + \nabla t$ with probability p, which is discovered by a simple brute force method. A more advanced method to mine such spatio-temporal association rules was proposed by Verhein et al. [205][204] to deal with the spatio-temporal semantics such as area effectively throughout the mining process and prune the search space as much as possible. It also defines a series of special patterns to describe important temporal characteristics of regions, e.g. stationary regions where many objects tend to stay over time and high traffic regions where many objects tend to move in (sink), move out (source), or move through (thoroughfares). There is also some work on the discovery of other interesting spatio-temporal association rules. For example, Mamoulis et al. proposed a technique to mining periodic patterns [143], represented by objects moving between regions and following the same routes approximately over regular time intervals. Mining spatio-temporal association rules is slow for large scale data. Wang et al. proposed an efficient disk based algorithm for mining the events changing over space and time [209].

2.4.4 Spatio-Temporal Anomaly Detection

Spatio-temporal anomaly detection is to detect data points significantly different than other data in their spatio-temporal neighborhoods. A simple strategy is provided in [49] by Cheng et al., which defines spatial outliers that are not always present in consecutive time frames as spatio-temporal

outliers. The temporal dimension is not well treated, since it can only compare spatial outliers between immediate time snapshots. Lu et al. [140] introduced the time dimension by linking the center region outliers in different time frames, which can be used to detect moving outliers. Similarly, Birant et al. [21] proposed to detect spatio-temporal outliers from large scale data sets by clustering data first followed by procedures of checking spatial and temporal neighbors separately. Basically, all these approaches define spatio-temporal outliers by incorporating time information into the spatial outliers. Sun et al. [196] gave a more general study by defining two kinds of spatio-temporal outliers: time period outliers given a region, and region outliers given a time period; however, mining spatial and temporal data for anomalies are mostly treated as two separate sub-problems.

2.4.5 Exploratory Analysis and Visualization of Spatio-Temporal Data

Spatio-temporal data sets are often very large and difficult to analyze and display. Since they are fundamental for decision support in many application contexts, recently a lot of interest has arisen toward data-mining techniques to extract relevant subsets of very large data repositories and effectively display the results by addressing many challenges such as indexing, representation, querying, mining, etc. Compieta et al. [52] proposed a complementary, dual 3D visualization environment for mining spatio-temporal data sets, which is composed of a Google Earth system to display the mining outcomes and a Java3D-based tool for providing advanced interactions with the data set in a non-geo-referenced space, such as displaying association rules. Olga et al. [159] developed a system that enables users to query clusters of time series with similar geometric shapes. These systems have mainly focused on the visualization aspect and do not consider issues such as large scale or high resolution data. Techniques such as wavelet tree [137], R-tree [153], and aRB-tree [164] are some of the well-known indexing techniques for online analytical processing. Camossi et al. [34] presented a multi-level indexing technique based on spatial and temporal granularity. However, it does not consider the similarity between data points when constructing the index. Their technique is also especially designed for clustering tasks. Bertolotto et al. [19] developed an approach to use clustering for data reduction purposes, which is similar to our work. However, they did not consider the problem of choosing representative samples for a specific data mining task such as anomaly detection. Denny et al. [67] presented a multi-level technique for

exploratory data analysis of hot spots. While there are other works on multi-level techniques [70], they are not directly applicable to our disturbance event detection problem.

2.5 Scope of Thesis Research

In this chapter, we reviewed some related work on spatial and temporal data mining in terms of three research areas: spatial data mining, temporal data mining and spatio-temporal data mining. The contributions and limitations of a number of algorithms and statistical models proposed in the literature for different mining tasks as clustering, prediction, anomaly detection and association rule mining were discussed. Despite the broad scope of research areas in spatial and temporal data mining, this thesis will mainly focus on the following research areas that are motivated by the problem of mining Earth Science data: spatial and temporal prediction, anomaly detection, anomaly characterization, and visualization.

CHAPTER 3

Localized Prediction with Spatial and Temporal Dependencies

Prediction is one of the most important tasks in data mining. Consider a training set:

$$\mathcal{D}_L = \{\boldsymbol{X}_L, \boldsymbol{Y}_L\} = [(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l)]^\top,$$

and test set:

$$\mathcal{D}_U = \{\boldsymbol{X}_U\} = [\boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, \ldots, \boldsymbol{x}_{l+u}]^\top,$$

the task of prediction is to build a predictive model from the training set $\mathcal{D}_L$ that maps input to output as:

$$f : \boldsymbol{X} \rightarrow \boldsymbol{Y}$$

The constructed model is then used to predict the target y for any given test example $\boldsymbol{x} \in \boldsymbol{X}_U$ as:

$$y = f(\boldsymbol{x})$$

Generally speaking, a prediction task can be divided into classification and regression tasks depending on the format of target value y. Classification task predicts a predefined discrete class label. For example, $y \in \{-1, +1\}$ for a two class problem. Regression task tries to predict a continuous target $y \in \mathcal{R}$.

In this chapter, we explore a localized method for the prediction problems in mining Earth Science data with spatial and temporal dependencies. The main contributions of this chapter are as follows:

- We develop an integrated localized prediction framework based on Support Vector Machine by incorporating the neighborhood information between the test examples and the training examples.

- The localized algorithm proposed is slow due to requirement of learning of one model for each testing example. A more efficient algorithm profile Support Vector Machine is proposed based on supervised clustering to reduce the computational cost in learning.

- Experiments on a number of real and synthetic data sets demonstrate the efficacy and efficiency of the proposed algorithms compared to K-nearest neighbor and Support Vector Machine. The applicability of our algorithms on Earth Science data is also investigated by incorporating spatial and temporal dependencies.

The remainder of this chapter is organized as follows. Section 3.1 introduces the problem of localized prediction and some background. Section 3.2 proposes the general framework of our localized Support Vector Machine. Section 3.3 presents the profile Support Vector Machine algorithms. We extend our algorithms to incorporate spatial and temporal dependencies in Section 3.4. The experimental results on both benchmark and Earth Science data sets are shown in Section 3.5.

3.1 Preliminaries

One of the most important questions in building a prediction model for a data set is whether to use a global learning or local learning framework. **Global prediction** learns a global predictive model $f : \boldsymbol{X} \rightarrow \boldsymbol{Y}$ from the entire training set $\mathcal{D}_L$. The formal definition of global prediction is given as following:

Definition 5. (Global Prediction) *Global prediction learns a single classification or regression model, which is applicable to all the test data, using the global characteristics of the entire training data.*

Many classification and regression algorithms have been proposed in the data mining and machine learning literature including Support Vector Machine [32][191], Decision Trees [222][221],

Neural Network [22], Hidden Markov Model [40][85], etc. They typically fall into the global prediction framework. In this work, we mostly focus on the Support Vector Machine algorithm. A detailed description of the Support Vector Machine algorithm will be discussed in Section 3.1.2.

Localized prediction algorithms tend to build multiple local prediction models instead of one global prediction model as: $\{f_1, f_2, \ldots, f_k\}$ where each local model f_i is associated a local subset of the training set. A formal definition of local prediction is given as following:

Definition 6. (Local Prediction) *Local prediction constructs multiple classification or regression models, each of which is applicable to the corresponding local part of the test data only, using the local characteristics of the training data.*

Localized prediction attempts to adjust the training system locally to the properties of the training set in each area of the input space, and thus has the potential to outperform global prediction especially when the data is non-evenly distributed [28]. The simplest localized prediction methods proposed in data mining literature is the K-nearest neighbor algorithm [177], which will be discussed in Section 3.1.1.

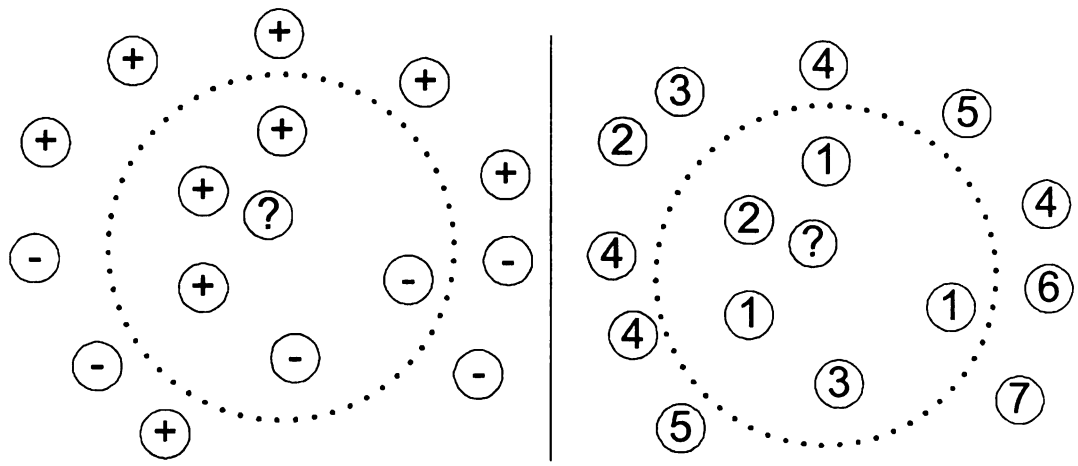
In this work, localized prediction is applied to Earth Science data with strong spatial or temporal dependencies. Given a training set with spatial and temporal features:

$$\mathcal{D}_L = [(t_1, o_1, \boldsymbol{x}_1, y_1), (t_2, o_2, \boldsymbol{x}_2, y_2), \ldots, (t_l, o_l, \boldsymbol{x}_l, y_l)]^\top,$$

localized prediction with spatial and temporal independences builds multiple local prediction models from the data by incorporating the spatial, temporal, or spatio-temporal neighborhood information.

3.1.1 K-nearest Neighbor Prediction

The K-nearest neighbor [55] [177] (KNN) predicts the target of a test example based on the distribution of training examples in its local neighborhood. Among the appealing features of the KNN learning include its simplicity and flexible modeling scheme. Nevertheless, it is susceptible to the curse of dimensionality [18] problem and is sensitive to the choice of neighborhood size K [97]. When K is too small, the KNN model may overfit the training data. On the other hand, a large K may also hurt the performance of the classifier by incorporating training examples that



(a) K-nearest Neighbor Classification. (b) K-nearest Neighbor Regression.

Figure 3.1. K-nearest Neighbor Prediction.

are unsuitable to the prediction of a test example. To overcome the difficulty of choosing K, the weighted K-nearest neighbor [102] model was introduced. In this method, the influence of each training example $\boldsymbol{x}_i$ is weighted by its similarity to the test example $\boldsymbol{x}$. Thus, it is important to define a a similarity measure to determine how closely matched two given examples are for the K-nearest neighbor algorithm. One of the most popular similarity metrics between two examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is using Radial Basis Function [165]:

$$\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left[-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{\sigma^2}\right] \tag{3.1}$$

where σ is a user-specified parameter to control the spread of the function values. Throughout this thesis, RBF function is used to measure the similarity between two examples.

In **K-nearest neighbor classification** as shown in Figure 3.1(a), the probability that a test example belongs to class y is computed as follows:

$$p(y|\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}_i \in \Omega_K(\boldsymbol{x})} \delta(y, y_i)\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{\boldsymbol{x}_i \in \Omega_K(\boldsymbol{x})} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x})} \tag{3.2}$$

where $\Omega_K(\boldsymbol{x})$ is a subset of observations in $\boldsymbol{\mathcal{D}}_L$ that correspond to the K-nearest neighbors of the test example $\boldsymbol{x}$, $\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x})$ denote the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}$, $\delta(y, y_i)$ as follows:

$$\delta(y, y_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

In **K-nearest neighbor regression** as shown in Figure 3.1(b), the continuous target value y for a test example $\boldsymbol{x}$ is computed as follows:

$$y = \frac{\sum_{\boldsymbol{x}_i \in \Omega_K(\boldsymbol{x})} y_i \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{\boldsymbol{x}_i \in \Omega_K(\boldsymbol{x})} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x})} \tag{3.4}$$

The similarity matrix $\boldsymbol{S}$ is often normalized so that its values range between 0 and 1. One popular choice for the similarity measure is the radial basis function (RBF) kernel [165] introduced in Equation (3.1).

K-nearest neighbor methods are simple and easy to understand. However, they have limitations. For example, they fail to produce accurate prediction results for high dimensional data, which is known as the "curse of dimensionality" problem; they are slow especially when the number of test examples is large; and they have difficulty in choosing the appropriate value for K [97], etc. Several variants of the nearest-neighbor classifier have been proposed in the literature to overcome some of its present limitations. Hastie and Tibshirani [100] attempted to address the curse of dimensionality problem by estimating the effective metric for computing the neighborhoods of the test examples. [69] also proposed a feature weighting scheme to alleviate the curse of dimensionality problem. Vincent and Bengio [206] developed the HKNN algorithm to fix the *missing sample* problem, which was shown to introduce artifacts in the decision surfaces produced by regular KNN. Both of these methods, however, are still computationally expensive when the number of test examples is large.

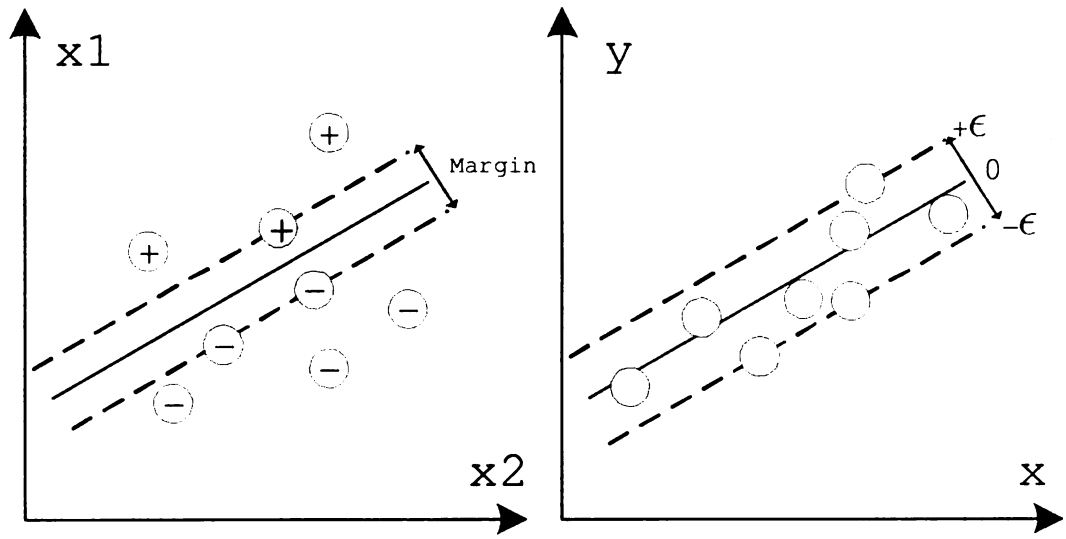
3.1.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method that can be applied to classification or regression tasks. Consider a training set $\boldsymbol{\mathcal{D}}_L$ and test set $\boldsymbol{\mathcal{D}}_U$, a nonlinear Support Vector Machine builds a classification or regression model f with form:

$$f(\boldsymbol{x}) = < \mathbf{w}, \varphi(\boldsymbol{x}) > + b$$

where $\mathbf{w}$ is the coefficient vector $[w_1, w_2, \cdots, w_p]$, b is the bias and $\varphi(\boldsymbol{x})$ is a function that maps $\boldsymbol{x}$ to the higher dimensional feature space so that decision function will be linear.

Support vector classification (SVC) learns a global decision boundary by maximizing the classification margin and minimizing the empirical classification error on the training set [32].



(a) Support Vector Classification. (b) Support Vector Regression.

Figure 3.2. Support Vector Machine.

For brevity, Support Vector Classification is discussed based on a two-class problem based on a discrete class label $y \in \{-1, +1\}$. Figure 3.2(a) shows a SVC decision boundary learned from a two-class training set, where the margin is defined as the minimum distance between the training examples and the hyperplane $f(\boldsymbol{x}) = 0$ generated by the decision function $f(\boldsymbol{x})$. The objective function of nonlinear Support Vector Machine is shown as below:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s. t} \quad & y_i(<\mathbf{w}, \varphi(\boldsymbol{x}_i)> -b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, 2, \ldots, l \end{aligned} \tag{3.5}$$

where ξ_i are slack variables introduced as an allowable error for misclassified data examples, C is a parameter that controls the tradeoff between the classification margin and cost of misclassification on training data. The preceding optimization function can be transformed into its dual form using the Lagrange multiplier method:

$$\begin{aligned} \max_{\alpha_1 \cdots \alpha_l} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s. t.} \quad & \sum_{i=1}^{l}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, 2, \ldots, l \end{aligned} \tag{3.6}$$

where ϕ is the kernel function used to compute the dot product of two examples in a high dimensional space as: $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$ and α_i is Lagrange multiplier or also called "support weight" assigned to each training example $\boldsymbol{x}_i$. Once the weights are determined, a test example $\boldsymbol{x}$ is classified as:

$$y = sign\left(\sum_{i=1}^{l} \alpha_i y_i \phi(\boldsymbol{x}, \boldsymbol{x}_i)\right) \tag{3.7}$$

Note that examples with large support weight α_i are called **support vectors**, which are essential examples to determine the decision boundary. Three examples of support vectors are shown as gray data points lying on the dotted lines in Figure 3.2(a).

Support Vector Classification can be extended to handle multi-class problems [108] using one-versus-all [178] method. Given a data set with N classes, one may construct N binary SVC models $\{f_1, f_2, \cdots, f_N\}$, where each f_i is trained to separate training examples that belong to class i from all other classes. The class label of a test example $\boldsymbol{x}$ can be predicted by applying the N binary SVC models and choosing the class i that yields the highest output value $f_i(\boldsymbol{x})$. Other strategies such as one-versus-one and error correcting code [6] are also available.

Support Vector Regression (SVR) learns a prediction function f from training examples by enforcing two constraints: one is to guarantee the prediction $f(\boldsymbol{x}_i)$ has at most ϵ deviation from the actual continuous response value y_i for all the training data; another is to make the prediction function f as flat as possible [191]. Figure 3.2(b) shows a simple illustration of a Support Vector Regression model, where all the training examples fall into a tube with width ϵ. The optimization problem of nonlinear Support Vector Regression is formulated as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{s. t} \quad & y_i - (<\mathbf{w}, \varphi(\boldsymbol{x}_i)> + b) \leq \epsilon + \xi_i \\ & (<\mathbf{w}, \varphi(\boldsymbol{x}_i)> + b) - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \; i = 1, 2, \ldots, l \end{aligned} \tag{3.8}$$

where ξ_i, ξ_i^* are slack variables with similar properties to those in SVC and C controls the tradeoff between minimizing prediction error and maximizing function flatness. To simplify the computa-

tion, the problem is transformed to its dual form as follows:

$$\begin{aligned} \max \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\phi(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ - \quad & \epsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \\ \text{s. t.} \quad & \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0 \\ & 0\le\alpha_i,\alpha_i^*\le C\ i=1,2,\ldots,l \end{aligned} \tag{3.9}$$

where ϕ is the kernel function and $(\alpha_i - \alpha_i^*)$ is the weight assigned to the training example $\boldsymbol{x}_i$. Once the weights have been determined, the response value for a test example $\boldsymbol{x}$ is predicted in the following way:

$$\mathbf{w} = \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\varphi(\boldsymbol{x}_i),\ \text{thus}\ , y = \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\phi(\boldsymbol{x}_i,\boldsymbol{x})+b$$

As it indicates, the coefficient vector $\mathbf{w}$ is represented as a combination of the training data $\boldsymbol{x}_i$.

Finding the right kernel for global nonlinear SVM given a data set can be quite a challenging task. On the one hand, a simple kernel may not capture all the intricacies of a complex decision surface. On the other hand, a kernel that is too flexible is susceptible to model overfitting. Although there have been several studies devoted to kernel learning [12], the proposed methods are generally expensive and do not scale well to very large data sets.

3.2 Localized Support Vector Machine

In this section, we present a framework called localized Support Vector Machine algorithm (LSVM), which integrates K-Nearest Neighbor with Support Vector Machine by incorporating the neighborhood information directly into SVM learning. The localized Support Vector Classification and Regression is formulated separately in Section 3.2.1 and Section 3.2.2.

3.2.1 Localized Support Vector Classification (LSVC)

We first discuss the classification case when $y \in \{-1, 1\}$. For each test example $\boldsymbol{x}_{l+s} \in \mathcal{D}_U, s = 1, 2, \ldots, u$, we construct a local Support Vector Machine from all the training examples whose

allowed misclassification errors ξ_i are weighted by their similarity to the test example. This idea can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})\xi_i \\ \text{s. t} \quad & y_i(<\mathbf{w}, \varphi(\boldsymbol{x}_i)> -b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, 2, \ldots, l \end{aligned} \tag{3.10}$$

Compared with regular SVC in Equation (3.5), the constraint ξ_i on misclassification error for training example $\boldsymbol{x}_i$ is weighted by its similarity to the test example $\boldsymbol{x}_{l+s}$. As a result, training examples close to the test example are promoted, while training examples located far away from the test example are penalized. The prediction of the test example is determined mostly by the training examples in its local neighborhood.

To further appreciate the role of the weight function, consider the dual form of the preceding optimization problem:

$$\begin{aligned} \max_{\alpha_1 \cdots \alpha_l} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s. t.} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s}), i = 1, 2, \ldots, l \end{aligned} \tag{3.11}$$

where a linear kernel $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ is employed for the LSVC. The details of the derivation can be found in the Appendix section. Unlike the optimization problem given in (3.6) for global nonlinear SVC, the upper bound for α_i has changed from C to $C\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})$.

3.2.2 Localized Support Vector Regression (LSVR)

For the regression case when $y \in \mathcal{R}$, we construct a local Support Vector Regression model for each test example $\boldsymbol{x}_{l+s} \in \boldsymbol{\mathcal{D}}_U$ by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{l} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})(\xi_i + \xi_i^*) \\ \text{s. t} \quad & y_i - (<\mathbf{w}, \varphi(\boldsymbol{x}_i)> +b) \leq \epsilon + \xi_i \\ & (<\mathbf{w}, \varphi(\boldsymbol{x}_i)> +b) - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \ i = 1, 2, \ldots, l \end{aligned} \tag{3.12}$$

This is analogous to the LSVC approach given in Section 3.2.1, where weight matrix S is used to weight the slack variables ξ_i and ξ_i^*. The LSVR formulation can be transformed to its dual form as follows:

$$\begin{aligned} \max \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ - \quad & \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \\ \text{s. t.} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ & 0 \le \alpha_i, \alpha_i^* \le C\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s}),\ i = 1, 2, \ldots, l \end{aligned} \tag{3.13}$$

where is ϕ is a linear kernel. Note again the upper bound for α_i,α_i^* has changed from C to $C\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})$ in localized Support Vector Regression compared with global nonlinear SVR in (3.9).

3.2.3 Discussion

Comparing the objective functions of LSVM (LSVC in Equation (3.11) and LSVR in Equation (3.13)) to the nonlinear global SVM (SVC in Equation (3.6) and SVR in Equation (3.9)), the only difference is that the constraint on the upper bound for α_i of LSVC and α_i,α_i^* of LSVR has changed from C to $C\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})$. The resulting effects of such modification is two-fold:

1. **Some of the support vectors for nonlinear SVM are no longer support vectors for LSVM.** As previously noted, a support vector for nonlinear SVM corresponds to a training example with positive support weight ($\alpha_i > 0$ for LSVC and $\alpha_i - \alpha_i^* > 0$ for LSVR) while a non-support vector has zero support weight ($\alpha_i = 0$ for LSVC and $\alpha_i - \alpha_i^* = 0$ for LSVR). Since the upper bound $C\boldsymbol{S}(\boldsymbol{x}_{l+s}, \boldsymbol{x}_i)$ decreases to zero when $\boldsymbol{x}_{l+s}$ and $\boldsymbol{x}_i$ are dissimilar, a support vector for nonlinear SVM that is dissimilar to the test example will have an upper bound close to zero, converting it into a non-support vector for LSVM.

2. **Some of the non-support vectors for nonlinear SVM become support vectors for LSVM.** As the support weight (α_i for LSVC and $\alpha_i - \alpha_i^*$ for LSVR) for support vectors that are dissimilar to the test example reduces to zero, some of the training examples

that are similar to the test example will be assigned weights larger than zero to ensure that the equality constraint ($\sum_{i=1}^{n} \alpha_i y_i = 0$ for LSVC and $\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$ for LSVR) is preserved.

There are two variants of LSVM considered in this study. If S is a real-valued similarity measure defined on the closed interval $[0, 1]$, the resulting algorithm is called *Soft Localized Support Vector Machine* (**SLSVM**). We employ the RBF kernel as our similarity measure for SLSVM, similar to the approach taken by weighted KNN (see Equation (3.1)). On the other hand, if S is a binary-valued function, the resulting algorithm is called *Hard Localized Support Vector Machine* (**HLSVM**). The similarity measure $S(\boldsymbol{x}_{l+s}, \boldsymbol{x}_i)$ is equal to 1 if $\boldsymbol{x}_i$ is part of the K nearest neighbor list for $\boldsymbol{x}_{l+s}$, and 0 otherwise. For HLSVM, the upper bound constraint for support weight is equal to C. Basically, it constructs a local SVM model for each test example based on its K-nearest neighbors [166]. This approach is equivalent to the KNN-SVM algorithm proposed by Zhang et al. [223]. This algorithm, which is a straightforward adaptation of KNN to SVM, has several limitations. First, the performance of the algorithm depends on the choice of K. Second, the model is not that flexible because it uses the same neighborhood size for each test example. Finally, the algorithm does not consider the similarity between the training and test examples when constructing the local SVM models. Once the nearest neighbors have been found, it will identify the local support vectors by solving the optimization problem given in (3.5). The performance of this algorithm is not as good as SLSVM according to our experiments.

One issue about LSVM is that it must build a local SVM model for each test example. Applying it to large data sets can be very costly. In the next section, we will present an algorithm to improve its efficiency.

3.3 Profile Support Vector Machine (PSVM)

Profile Support Vector (PSVM) algorithms reduce the computation cost of Localized Support Vector Machine by reducing the number of local SVM models that need to be built. To understand the intuition behind this approach, consider the optimization problem given in Equation (3.13) and Equation (3.11), which must be solved separately for each test example $\boldsymbol{x}_{l+s}$. Notice that the optimization problem is nearly identical for each test example, except for the up-

per bound constraint for α_i of LSVC and α_i,α_i^* of LSVR, which depends on $\boldsymbol{S}(\boldsymbol{x}_{l+s}, \boldsymbol{x}_i)$. Let $\vec{\boldsymbol{S}}(\boldsymbol{x}_{l+s}) = \left[\boldsymbol{S}(\boldsymbol{x}_{l+s}, \boldsymbol{x}_1), \boldsymbol{S}(\boldsymbol{x}_{l+s}, \boldsymbol{x}_2), \ldots, \boldsymbol{S}(\boldsymbol{x}_{l+s}, \boldsymbol{x}_n)\right]^\top$ denote a column vector of similarity values between a test example $\boldsymbol{x}_{l+s}$ and the set of training examples $\{\boldsymbol{x}_i\}_{i=1}^{l}$. Since α_i determines whether $\boldsymbol{x}_i$ is a support vector, we expect test examples with highly correlated similarity vector $\vec{\boldsymbol{S}}$ to share many common support vectors. For such test examples, it may be sufficient to build a single LSVM model to predict them. To achieve this, a simple strategy is to group the training and test examples using standard clustering algorithm such as K-means. However, building local models for such clusters will be problematic, e.g. one cluster may contain examples from one class only for the classification problem.

3.3.1 Supervised Clustering Algorithms: MagKmeans and VarKmeans

We have developed supervised clustering algorithms called MagKmeans and VarKmeans, for LSVC and LSVR respectively, to determine the set of training examples that can be trained together and the set of test examples that can be predicted together. Let $\boldsymbol{S} = [\vec{\boldsymbol{S}}(\boldsymbol{x}_{l+1})\ \vec{\boldsymbol{S}}(\boldsymbol{x}_{l+2}) \ldots \vec{\boldsymbol{S}}(\boldsymbol{x}_{l+u})]$ be an $l \times u$ matrix, where l is the training set size, u is the test set size, and the (i, j)-th entry of the matrix denote the similarity between the training example $\boldsymbol{x}_i$ and the test example $\boldsymbol{x}_{l+j}$. The goal is to find κ clusters from $\boldsymbol{S}$ such that $\kappa << l$.

MagKmeans: A Supervised Clustering Algorithm for LSVC

We consider the classification case when $y \in \{-1, 1\}$ in this section. Our clustering task is somewhat different than conventional unsupervised clustering. First, the data to be clustered is $\boldsymbol{S}$, which contains the similarity between every training-test example pair. Second, conventional clustering methods consider only the proximity between examples and ignore their class distributions. As a result, some clusters may not contain enough representative examples from the different classes to construct a reliable local Support Vector Classification model.

The **MagKmeans** algorithm extends regular K-means [114] by considering the class distribution of training examples within each cluster. Let $Y = [y_1, y_2, \ldots, y_l]^\top$ be the class labels of the training examples. The objective function for MagKmeans is:

$$\min_{\overline{X}, Z} \sum_{j=1}^{\kappa} \sum_{i=1}^{n} Z_{i,j} \|X_i - \overline{X}_j\|_2^2 + R \sum_{j=1}^{\kappa} \left| \sum_{i=1}^{l} Z_{i,j} y_i \right| \tag{3.14}$$

where X_i is the i-th row of the similarity matrix $\boldsymbol{S}$, $\overline{X}_j$ is an $1 \times u$ row vector representing the centroid of the jth cluster, R is a non-negative scaling parameter, and Z is the cluster membership matrix, whose (i, j)-th element is equal to one if the ith training example belongs to the jth cluster, and zero otherwise. The first term in the objective function corresponds to a cluster cohesion measure. Minimizing this term would ensure that training examples in the same cluster have highly correlated similarity vectors. The second term in the objective function measures the uniformity of class distributions in each cluster. Minimizing this term would ensure that each cluster contains a balanced number of positive and negative examples.

The cluster centroids $\overline{X}$ and cluster membership matrix Z are estimated iteratively as follows. First, we fix the cluster centroids and use them to determine the cluster membership matrix. Next, the revised cluster membership matrix is used to update the centroids. This procedure is repeated until the algorithm converges to a local minimum. To compute the cluster membership matrix Z, we transform the original optimization problem into the following form using κ slack variables t_j:

$$\begin{aligned} \min_{Z_{i,j}} \quad & \sum_{j=1}^{\kappa} \sum_{i=1}^{l} Z_{i,j} \|X_i - \overline{X}_j\|_2^2 + R \sum_{j=1}^{\kappa} t_j \\ \text{s. t.} \quad & -t_j \leq \sum_{i=1}^{n} Z_{i,j} y_i \leq t_j \\ & t_j \geq 0,\ 0 \leq Z_{i,j} \leq 1 \\ & \sum_{j=1}^{\kappa} Z_{i,j} = 1 \end{aligned} \tag{3.15}$$

The preceding optimization problem can be solved using linear programming [184]. When the cluster membership matrix is fixed, the centroids are updated based on the following equation:

$$\overline{X}_j = \frac{\sum_{i=1}^{l} Z_{i,j} X_i}{\sum_{i=1}^{l} Z_{i,j}} \tag{3.16}$$

Here R serves as a scaling parameter to handle the tradeoff between the two terms in our objective function: cluster cohesion and class imbalance. For our experiments, R is set to $1/\kappa$ times the diameter of the data set.

Figure 3.3 illustrates how the MagKmeans algorithm works. The initial cluster (the left figure) contains only positive examples. As the algorithm progresses, some positive examples are expelled from the cluster and replaced by the nearby negative examples (the right figure). By

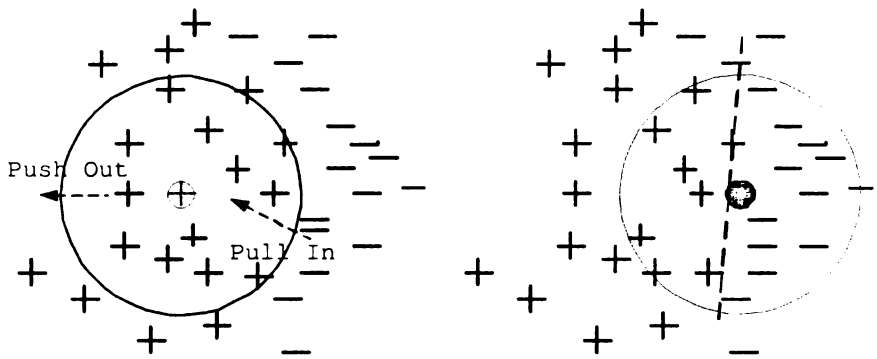


Figure 3.3. An illustration of the MagKmeans clustering algorithm. The cluster in the left figure contains only positive examples. After several iterations, some of the positive examples will be expelled from the cluster while some negative example are absorbed into the cluster to achieve balance in the class distribution (the right figure).

Algorithm 1 MagKmeans algorithm

Input: Similarity matrix $\boldsymbol{S}$, class vector Y, and number of clusters κ

Output: Cluster membership matrix Z and the centroid matrix $\overline{X}$

1: Randomly initialize the centroid matrix $\overline{X}$.

2: **repeat**

3: Update the cluster membership matrix $Z_{i,j}$ by solving the linear programming problem given in Equation (3.15).

4: Update the centroid matrix $\overline{X}$ using Equation (3.16).

5: **until** convergence

ensuring that the cluster has almost equal representation from each class, a local SVM model can be constructed from the training examples. A summary of the MagKmeans algorithm is given in Algorithm 1.

VarKmeans: A Supervised Clustering Algorithm for LSVR

Next we consider the case for regression. Similarly to the **MagKmeans** algorithm proposed in Section 3.3.1, the **VarKmeans** algorithm proposed in this thesis modifies the objective function of the K-means algorithm to consider the variance of the response values in the training examples of each cluster. The data used for clustering consists of two parts: (1) the weight matrix $\boldsymbol{S}$ and (2)

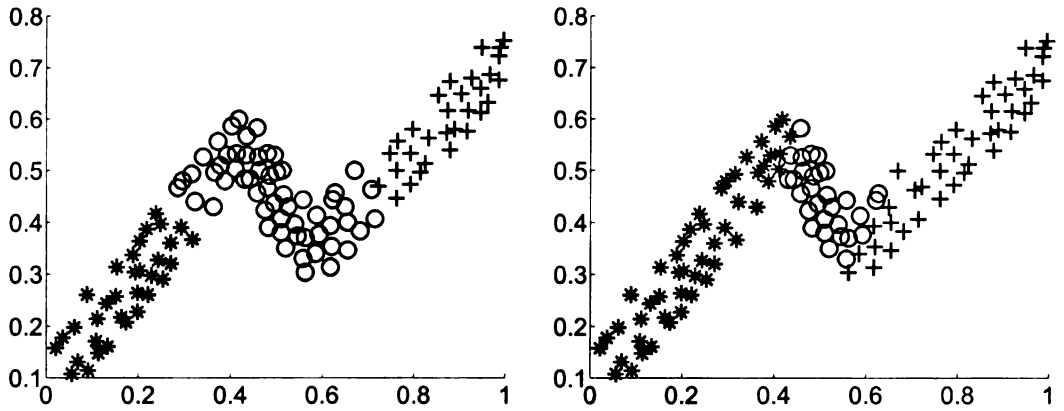


Figure 3.4. An illustration of the VarKmeans clustering algorithm. The left figure plots the clustering results by K-means on the data set. The right figure plots the clustering results by VarKmeans on the data set. VarKmeans produce much better clusters for learning three linear regression segments.

the response values $Y = (y_1, \ldots, y_l)^\top$. The objective function for VarKmeans is:

$$\min_{Z,\overline{X},\overline{Y}} \sum_{j=1}^{\kappa}\sum_{i=1}^{l} Z_{i,j}\|X_i - \overline{X}_j\|_2^2 - R\sum_{j=1}^{\kappa}\frac{\sum_{i=1}^{l} Z_{i,j}(Y_i - \overline{Y}_j)^2}{\sum_{i=1}^{l} Z_{i,j}} \tag{3.17}$$

where $\overline{X}$ is the centroid matrix for predictor variables, $\overline{Y}$ is the centroid vector for response variable, and R is again set to $1/\kappa$ times the diameter of the data set.

Note that the first part of the objective function groups data points together based on the similarity of their predictor values, while the second part maximizes the variance of the response values in each cluster. We employ an EM-like algorithm [66] to optimize the objective function given in Equation (3.17). More specifically, we first search for the optimal cluster membership matrix Z by fixing the centroid $[\overline{X}_j, \overline{Y}_j]$ for all j. We then search for the optimal centroid $[\overline{X}_j, \overline{Y}_j]$ by fixing the cluster memberships Z. These steps are repeated until the algorithm converges to a local minimum.

When $[\overline{X}_j, \overline{Y}_j]$ is fixed, the cluster membership matrix Z can be computed efficiently using linear programming. To do this, we transform the original optimization problem into the following

Algorithm 2 VarKmeans algorithm

Input: Similarity matrix $\boldsymbol{S}$, Response vector Y, and number of clusters κ

Output: Cluster membership matrix Z and the centroid matrix $\overline{X}, \overline{Y}$

1: Randomly initialize the centroid matrix $\overline{X}, \overline{Y}$.

2: **repeat**

3: Update the cluster membership matrix $Z_{i,j}$ by solving the linear programming problem given in Equation (3.18).

4: Update the centroid matrix $\overline{X}$ using Equation (3.19) and $\overline{Y}$ using Equation (3.20).

5: **until** convergence

form using κ slack variables t_j ($j = 1, \cdots, \kappa$):

$$\begin{aligned} \min_{Z_{i,j}} \quad & \left(\sum_{j=1}^{\kappa} \sum_{i=1}^{l} Z_{i,j}(X_i - \overline{X}_j)^2 - R \sum_{j=1}^{\kappa} t_j \right) \\ \text{s. t.} \quad & \sum_{i=1}^{l} Z_{i,j}(Y_i - \overline{Y}_j)^2 \geq t_j \sum_{i=1}^{l} Z_{i,j} \\ & t_j > 0,\ 0 \leq Z_{i,j} \leq 1 \\ & \sum_{j=1}^{\kappa} Z_{i,j} = 1 \end{aligned} \tag{3.18}$$

When the cluster membership matrix Z is fixed, the following equation is used to update each centroid $[\overline{X}_j, \overline{Y}_j]$:

$$\overline{X}_j = \frac{\sum_{i=1}^{l} Z_{i,j} X_i}{\sum_{i=1}^{l} Z_{i,j}} \tag{3.19}$$

$$\overline{Y}_j = \frac{\sum_{i=1}^{l} Z_{i,j} Y_i}{\sum_{i=1}^{l} Z_{i,j}} \tag{3.20}$$

By ensuring that the cluster has almost uniform distribution of response values, a local SVR model can be constructed from the training examples. A summary of the VarKmeans algorithm is given in Algorithm 2.

We illustrate the difference between the clustering results produced by regular K-means and VarKmeans using the synthetic data set shown in Figure 3.4. The data set contains a data set in the shape of a nonlinear regression line with three linear segments. The figure in the left shows the results of regular K-means, whereas the figure on the right shows the results of VarKmeans,

which produces clusters that contain more widely spread response values in the Y-axis. The latter clustering is clearly more desirable for constructing local linear SVR models.

3.3.2 Profile Support Vector Classification (PSVC)

After clustering the data using our MagKmeans algorithm, a local LSVC model is constructed for each cluster. The SVC model for the kth cluster is obtained by solving the following optimization problem:

$$\begin{aligned} \max_{\widetilde{\alpha}} \quad & \sum_{i=1}^{l} \widetilde{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{l} \widetilde{\alpha}_i \widetilde{\alpha}_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \\ \text{s. t.} \quad & \sum_{i=1}^{l} \widetilde{\alpha}_i y_i = 0 \\ & 0 \leq \widetilde{\alpha}_i \leq \beta Z_{i,k}, \; i = 1, 2, \ldots, l \end{aligned} \tag{3.21}$$

Since $Z_{i,k} \in \{0, 1\}$, only the training examples assigned to the cluster will be used to build the local LSVC model.

Algorithm 3 Profile SVC algorithm

Input: Training set $\mathcal{D}_L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, test set $\mathcal{D}_U = \{\boldsymbol{x}_{l+j}\}_{j=1}^{u}$, and number of clusters κ

Output: Class labels for the test set $\{y_{l+j}\}_{j=l}^{u}$

1: Compute the $l \times u$ similarity matrix $\boldsymbol{S}$.
2: $(Z, \overline{X}) \leftarrow$ MagKmeans($\boldsymbol{S}$, Y, κ)
3: **for** k = 1 to κ **do**
4: SVC$_k \leftarrow$ Build_Local_SVC($\mathcal{D}_L$, Z, k)
5: **end for**
6: **for** each test example $\boldsymbol{x}_{l+j} \in \mathcal{D}_U$ **do**
7: $k = \arg\max_i \overline{X}_{i,j}$
8: $y_{l+j} =$ SVC$_k(\boldsymbol{x}_{l+j})$
9: **end for**

To classify the test set, we need to determine the local SVC model that should be invoked for each test example. Let $\overline{X}$ be the $\kappa \times u$ centroid matrix produced by the MagKmeans algorithm. Since the (i, j)-th element of the centroid matrix indicates the average similarity between the test

example $\boldsymbol{x}_{l+j}$ and the training examples in cluster i, we assign $\boldsymbol{x}_{l+j}$ to the cluster with highest similarity. We then apply the local SVM model associated with the cluster to predict the class label of $\boldsymbol{x}_{l+j}$. The procedure for model building and test of the PSVC algorithm is summarized in Algorithm 3.

3.3.3 Profile Support Vector Regression (PSVR)

Similar to PSVC, we construct a local SVR for each cluster after clustering the data using the VarKmeans algorithm. More specifically, the LSVR model for the kth cluster is obtained by solving the following optimization problem:

$$\begin{aligned}
\max \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\widetilde{\alpha}_i - \widetilde{\alpha}_i^*)(\widetilde{\alpha}_j - \widetilde{\alpha}_j^*)\boldsymbol{x}_i\boldsymbol{x}_j \\
- \quad & \epsilon\sum_{i=1}^{l}(\widetilde{\alpha}_i + \widetilde{\alpha}_i^*) + \sum_{i=1}^{l} y_i(\widetilde{\alpha}_i - \widetilde{\alpha}_i^*) \\
\text{s. t.} \quad & \sum_{i=1}^{l}(\widetilde{\alpha}_i - \widetilde{\alpha}_i^*) = 0 \\
& 0 \le \widetilde{\alpha}_i, \widetilde{\alpha}_i^* \le CZ_{i,k},\ i = 1, 2, \ldots, l
\end{aligned}$$

For hard clustering, since $Z_{i,k} \in \{0, 1\}$, the above optimization problem is equivalent to building a linear SVR using only the training examples assigned to the cluster. In turn, the κ LSVRs constructed from the clusters form a piecewise decision boundary with κ linear segments.

Now comes the testing step. We need to decide which local model should be used to predict a test example. To do this, we refer to the centroid matrix $\overline{X}$ obtained by the VarKmeans algorithm. Since the (i, j)-th element of the centroid matrix indicates the similarity between a test example $\boldsymbol{x}_j$ to cluster i, we assign the test example to the cluster with highest similarity. Finally, the response value of the test example is determined by applying the corresponding LSVR model for its assigned cluster. The procedure for model building and testing of the PSVR algorithm is summarized in Algorithm 4.

3.3.4 Summary

In short, to avoid training a separate LSVM model for each test example, the PSVM algorithm partitions the data into κ clusters and trains a local SVM model for each cluster. κ is typically

Algorithm 4 Profile SVR algorithm

Input: Training set $\mathcal{D}_L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, test set $\mathcal{D}_U = \{\boldsymbol{x}_{l+j}\}_{j=1}^{u}$, and number of clusters κ

Output: Predictions for the test set $\{y_{l+j}\}_{j=1}^{u}$

1: Compute the $l \times u$ similarity matrix $\boldsymbol{S}$.

2: $(Z, \overline{X}, XY) \leftarrow$ VarKmeans($\boldsymbol{S}, Y, \kappa$)

3: **for** $k = 1$ to κ **do**

4: SVR$_k \leftarrow$ Build_Local_SVR($\mathcal{D}_L, Z, k$)

5: **end for**

6: **for** each test example $\boldsymbol{x}_{l+j} \in \mathcal{D}_U$ **do**

7: $k = \arg\max_i \overline{X}_{i,j}$

8: $y_{l+j} =$ SVR$_k(\boldsymbol{x}_{l+j})$

9: **end for**

chosen to be considerably smaller than the number of test examples u to reduce the computational cost.

3.4 LSVM and PSVM for Spatial and Temporal Prediction

According to a statement in [28], building multiple local models performs better than one global model, especially for data that are not evenly distributed. There has been some recent research on local learning for spatial and temporal prediction, which demonstrated the superior performance of building multiple local machine learning models compared with global machine algorithms [89][132][128][125]. For example, a similar approach to KNN-SVM has been applied to spatial prediction problem by Gilardi et al. in [89] for data sets with strong spatial dependencies.

This section describes how our LSVM and PSVM formulations can be extended to prediction problems with spatial and temporal dependencies. For temporal data, each example is associated with a time stamp and the prediction of a test example is often influenced by recently arrived training examples. Let $\boldsymbol{x}_{l+s}$ be a test example that arrives at time t_{l+s} and $\boldsymbol{x}_i$ be a training example that arrives at an earlier time t_i. We may define a temporal similarity function $\boldsymbol{S}(t_i, t_{l+s})$ to measure the proximity between two examples in terms of their time stamp. Such timing information can then be incorporated into LSVM by constraining the upper bound for α_i of LSVC and α_i,α_i^*

of LSVR from C to $C\boldsymbol{S}(t_i, t_{l+s})$.

Spatial dependencies can be incorporated into the proposed framework in a similar manner. Let $\boldsymbol{x}_{l+s}$ be a test example located at o_{l+s} and $\boldsymbol{x}_i$ is a training example located at o_i. The spatial proximity between the examples is given by the spatial similarity function $\boldsymbol{S}(o_i, o_{l+s})$. The function can be incorporated into LSVM by changing the upper bound for α_i of LSVC and α_i,α_i^* of LSVR from C to $C\boldsymbol{S}(o_i, o_{l+s})$.

Our approach is also flexible enough to accommodate data with spatio-temporal dependencies. Let $\boldsymbol{x}_{l+s}$ be a test example located at o_{l+s} with time stamp t_{l+s} and $\boldsymbol{x}_i$ is a training example located at o_i with time stamp t_i. The spatio-temporal proximity between the examples is given by the spatio-temporal similarity function $\boldsymbol{S}([t_i\ o_i], [t_{l+s}\ o_{l+s}])$. The function can be incorporated into LSVM by changing the upper bound for α_i of LSVC and α_i,α_i^* of LSVR from C to $C\boldsymbol{S}([t_i\ o_i], [t_{l+s}\ o_{l+s}])$.

Specifically, the modified optimization problem for LSVC in Equation (3.11) with spatial, temporal, or spatio-temporal constraint becomes:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s. t.} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C\boldsymbol{S},\ i = 1, 2, \ldots, l \end{aligned} \tag{3.22}$$

where:

$$\boldsymbol{S} = \begin{cases} \boldsymbol{S}(t_i, t_{l+s}) & \text{temporal} \\ \boldsymbol{S}(o_i, o_{l+s}) & \text{spatial} \\ \boldsymbol{S}([t_i\ o_i], [t_{l+s}\ o_{l+s}]) & \text{spatio-temporal} \end{cases} \tag{3.23}$$

The optimization problem for LSVR in Equation (3.13) is modified similarly as:

$$\begin{aligned} \max \quad & -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ - \quad & \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*) \\ \text{s. t.} \quad & \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 \\ & 0 \leq \alpha_i, \alpha_i^* \leq C\boldsymbol{S},\ i = 1, 2, \ldots, l \end{aligned} \tag{3.24}$$

where $\boldsymbol{S}$ is defined in Equation (3.23).

A PSVM formulation can also be developed to incorporate the spatial or temporal dependencies. Unlike its original formulation, clustering is performed on the spatial similarity matrix $\boldsymbol{S} = [\boldsymbol{S}(o_i, o_{l+s})]$, temporal similarity matrix $\boldsymbol{S} = [\boldsymbol{S}(t_i, t_{l+s})]$, or spatio-temporal similarity matrix $\boldsymbol{S} = [\boldsymbol{S}([t_i\ o_i], [t_{l+s}\ o_{l+s}])]$. Depending on whether a prediction task is classification or regression, one could use MagKmeans or VarKmeans to partition the training and test data. A local SVM model can then be constructed for each cluster partition. The experimental results demonstrate the successful application of spatial and temporal LSVM and PSVM to the Earth Science data.

3.5 Experimental Results

This section describes the experimental results obtained by applying the proposed algorithms to a variety of data sets. The proposed LSVC and LSVR algorithms are implemented by modifying the C++ code for the LIBSVM tool developed by Chang and Lin [37]. For LSVC, the tool is applicable to multi-class classification problems using the one-versus-all approach as described in Section 3.1.2. To support PSVC and PSVR, we have also implemented the MagKmeans and VarKmeans algorithms to cluster the similarity matrix $\boldsymbol{S}$. Our experiments were conducted on a Windows XP machine with 3.0GHz CPU and 1.0GB RAM. The main objectives of the experiments are:

1. To compare the difference between the support vectors and decision surfaces obtained using LSVM and nonlinear SVM (Section 3.5.1).

2. To compare the relative performance of KNN, SVM, LSVM, and PSVM on a variety of real-world data sets (Section 3.5.2).

3. To illustrate the application of LSVM and PSVM to Earth Science data sets with spatial and temporal dependencies (Section 3.5.3).

4. To perform sensitivity analysis on parameters of the proposed framework (Section 3.5.4).

To assess the goodness of a prediction algorithm on the test set $\mathcal{D}_U$, two metrics are used in this thesis. The **accuracy** of classification model is measured by the percentage of unlabeled data

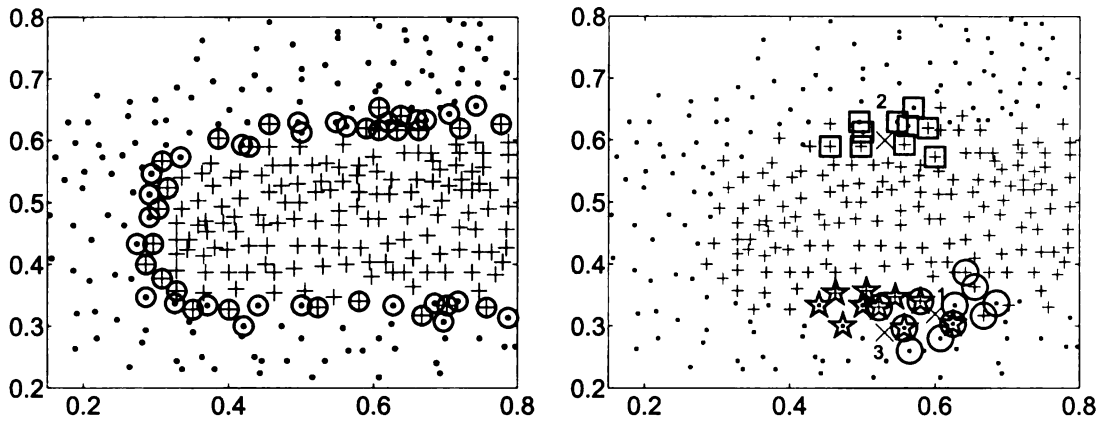


Figure 3.5. Comparison between LSVC and nonlinear SVC. The left diagram shows the support vectors obtained by nonlinear SVC; whereas the right diagram shows the support vectors obtained for three test examples using LSVC.

whose labels are accurately predicted:

$$\text{Accuracy} = \frac{\sum_{i=l+1}^{l+u} \delta(y_i, \widehat{y}_i)}{u} \tag{3.25}$$

where y_i is the true label, $\widehat{y}_i$ is the predicted label for text example $\boldsymbol{x}_i$ and $\delta(y_i, \widehat{y}_i)$ is defined similarly as in Equation (3.3).

R-square can be used to evaluate the goodness of the learned regression model, which is defined as the ratio of the sum of squares explained by a regression model and the total sum of squares around the mean:

$$R^2 = 1 - \frac{\sum_{i=l+1}^{l+u} (y_i - \widehat{y}_i)^2}{\sum_{i=l+1}^{l+u} (y_i - \overline{y}_i)^2} \tag{3.26}$$

where y_i is the true value, $\widehat{y}_i$ is the predicted value for text example $\boldsymbol{x}_i$ and $\overline{y}_i$ is the expected value (usually it is the average of the true values in the test data). Note that a regression model with larger value of R^2 fits the data better.

3.5.1 Comparison between LSVM/PSVM and Nonlinear SVM

For this experiment, we use synthetic data sets to demonstrate the difference between the support vectors and decision boundaries found by LSVM/PSVM in contrast to those found by nonlinear

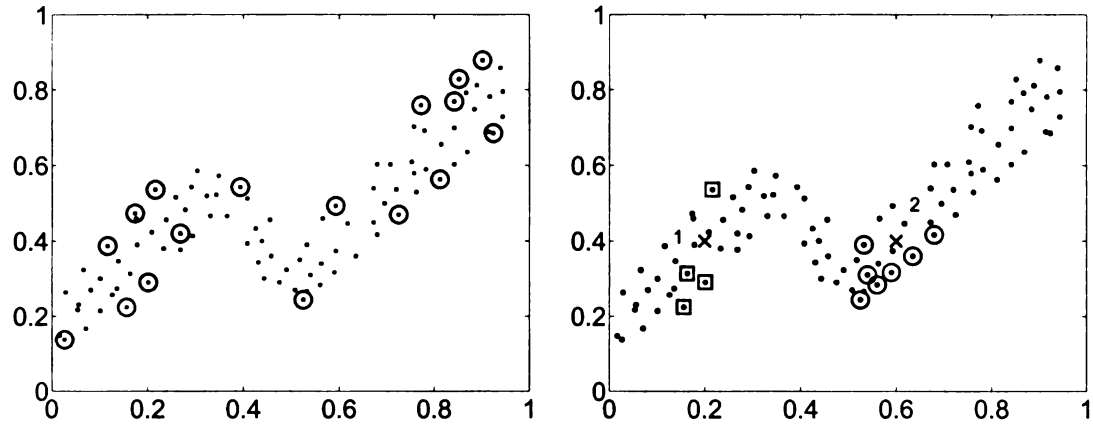


Figure 3.6. Comparison between LSVR and nonlinear SVR. The left diagram shows the support vectors obtained by nonlinear SVR; whereas the right diagram shows the support vectors obtained for two test examples using LSVR.

SVM. As previously mentioned in Section 3.2, LSVM may reduce the influence of support vectors (for nonlinear SVM) that are far away from a test example while converting other training examples in its vicinity into support vectors.

A. Comparison of Support Vectors

Consider the synthetic data set depicted in Figure 3.5 for a two class classification problem. The left diagram shows the data distributions for the two classes, represented as $\cdot$ and $+$, respectively. The support vectors found using a nonlinear SVC with an RBF kernel function are represented by the symbols $\odot$ and $\oplus$. Observe that the support vectors are located along the whole boundaries between the two classes. The right diagram of Figure 3.5 shows the corresponding support vectors found by LSVC for three selected test examples. The locations of the test examples are marked by the symbol $\times$ while their support vectors are represented with three different symbols. We can see that those support vectors found by LSVC lie closer to their corresponding test examples. A similar phenomenon is found in Figure 3.6 for a regression task. The left diagram shows a synthetic data set represented as $\cdot$ and the support vectors found using a nonlinear SVR algorithm denoted as $\odot$ surrounding the whole data set. The right diagram demonstrates two individual test examples marked by the symbol $\times$ and the support vectors found for them by LSVR algorithms. Again, the support vectors moved closer to those test examples.

Upon comparing left and right diagrams in both Figure 3.5 and Figure 3.6, it is clear that some

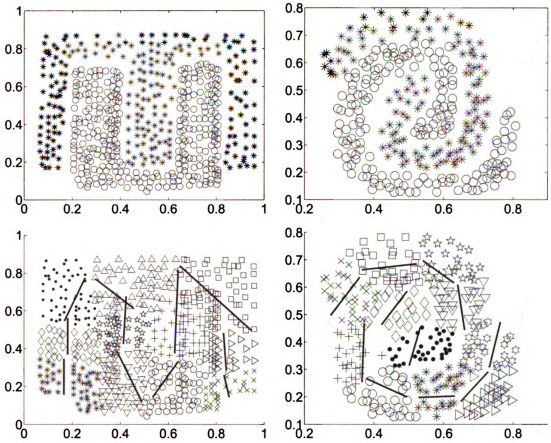


Figure 3.7. The diagrams in the top panel show the distribution for two synthetic data sets. Each data set comprises of two classes marked by $\circ$ and $*$, respectively. The diagrams in the bottom panel show the decision boundaries generated by PSVC. Each symbol represents one of the clusters produced by the MagKmeans algorithm.

support vectors for nonlinear SVM are not chosen as support vectors for LSVM because they are far away from the test examples. In the meantime, some non-support vectors for nonlinear SVM have become support vectors for LSVM because of their proximity to the test examples. Furthermore, the test examples marked as 2 and 3 in Figure 3.5 share several common support vectors because of their close proximity to each other. Even though their support vectors are not exactly identical, they have the potential of sharing the same decision surface, which justifies our motivation for using clustering to reduce the computational cost of LSVM.

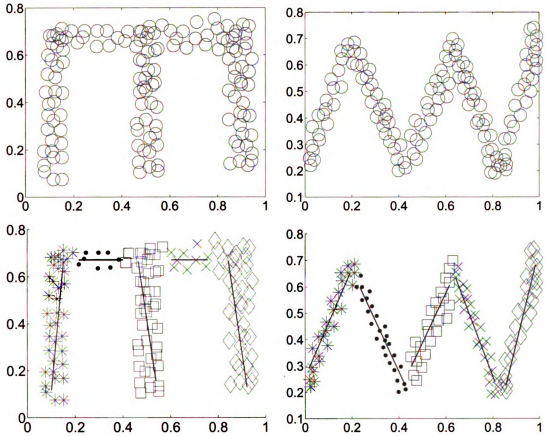


Figure 3.8. The diagrams in the top panel show the distribution for two synthetic data sets. Y-axis represents the response and X-axis represents the predictor. The diagrams in the bottom panel show the regression generated by PSVR. Each symbol represents one of the clusters produced by the VarKmeans algorithm.

B. Effect of MagKmeans and VarKmeans in PSVM

Two synthetic data sets are generated for classification as shown in the top panel of Figure 3.7. The bottom panel shows the corresponding decision boundaries generated by PSVC. For the first data set shown in the left diagram of Figure 3.7, the horse-shoe shaped decision boundary is approximated by 11 piecewise linear decision boundaries. For the second data set shown in the right diagram, the spiral-shaped decision boundary is also approximated by 11 piecewise linear decision boundaries. In summary, the results of this experiment demonstrate the ability of PSVC to fit a complex decision boundary using multiple piecewise linear segments.

We generate another two synthetic data sets for regression tasks as shown in the top panel of Figure 3.8. The bottom panels shows the regression function learned by PSVR. For the first data

Table 3.1. Description of the 18 UCI datasets used for classification.

Data	# Instances	# Attributes
Breast	699	10
Glass	214	9
Iris	150	4
KDDCup	1015062	38
Physics	171017	10
Yeast	1484	8
Robot	173841	26
Ecoli	336	7
Arcene	200	10000
Arrhyth	452	279
Balance	625	4
Car	1728	6
OptDigit	3477	64
Contrace	1473	9
Dermatology	366	34
Sonar	208	60
Gisette	6000	5000
Krkopt	28056	6

set in the left diagram of Figure 3.8, M-shape regression function is approximated by 5 piecewise linear regression lines. For the second data set shown in right diagram, the sin wave regression function is also approximated by 5 piecewise linear regression lines. The results of this experiment demonstrate the ability of PSVR to fit a complex regression function using multiple piecewise linear segments.

3.5.2 Performance Comparison on Real-World Data

In this experiment, we conduct several experiments on real-word data to compare the performance of LSVM and PSVM against other supervised prediction methods.

We use 18 data sets (described in Table 3.1) from the UCI repository [154] to compare the classification performances of HLSVC, SLSVC, and PSVC against KNN and nonlinear SVC in terms of their accuracy. Some of the data sets such as "Breast", "Glass", "Iris", "KDDcup IDS", "Physics", "Yeast", "Robot", and "Ecoli" are multi-class prediction problems. Since the clustering part of our proposed PSVM algorithms are designed for binary classification problems, we divide the classes for these data sets into two groups and relabel one group as the positive class and

Table 3.2. Description of the 18 UCI datasets used for regression.

Data	# Instances	# Attributes
AutoPrice	205	25
AutoMpg	398	7
Cancer	699	9
Concrete	1030	8
Hardware	209	8
Servo	167	12
Abalone	4177	8
Stock	950	9

the others as negative class. For some of the large data sets such as "KDDCup", "Physics", and "Robot" we randomly sample 4500 examples from each class to form the data sets. Another 8 data sets are collected from UCI repository [154] to compare the regression performances of HLSVR, SLSVR, and PSVR against KNNR and nonlinear SVR in terms of their R^2. The data sets used here are "AutoPrice", "AutoMpg", "Cancer", "Concrete", "Hardware", "Servo", "Abalone", and "Stock" (described in Table 3.2). The attributes for every data set have been normalized so that their range goes from 0 to 1. The experimental results reported in this study are obtained by applying five-fold cross validation on the data sets. To make the problem more challenging, we use one-fifth of the data for training and the remaining four-fifths for testing. Each experiment is also repeated ten times and the final results reported are obtained by averaging the results over ten trials.

A. Model Selection

Each prediction algorithm investigated in this study has one or more parameters that must be specified by the user. For the KNN prediction, this corresponds to the number of nearest neighbors to be considered for each test example. For nonlinear SVC and SVR with RBF kernel (Equation (3.1)), the user must specify the kernel width σ to control the model complexity. Scaling parameter C in SVC controls the tradeoff between classification margin and cost of misclassification on the training set, while in SVR it controls the tradeoff between flatness of regression function and prediction error. The same parameter C is also used in HLSVM, SLSVM, and PSVM. There is an additional particular parameter ϵ for SVR, HLSVR, SLSVR, and PSVR, which is used to specify the width of the tube. Both SLSVM and PSVM also employ the RBF kernel to define the

Table 3.3. Classification accuracies (%) for SVC, KNNC, HLSVC (KNN-SVC), SLSVC, and PSVC on the 18 UCI datasets.

Data	SVC	KNNC	HLSVC	SLSVC	PSVC
Breast	94.57	95.70	95.42	**96.85**	96.52
Glass	64.33	54.30	62.67	66.39	**66.91**
Iris	92.75	89.75	74.71	96.42	**97.06**
KDDCup	98.14	95.21	98.01	**99.71**	99.29
Physics	82.98	67.82	83.57	**86.42**	85.57
Yeast	92.31	93.36	94.62	**96.00**	95.83
Robot	85.35	78.64	85.87	**87.23**	86.23
Ecoli	93.41	93.30	93.33	**94.89**	94.44
Arcene	63.90	68.94	70.06	**72.30**	71.93
Arrhyth	68.78	66.74	68.45	**70.22**	69.86
Balance	89.10	88.70	90.62	**92.02**	89.94
Car	79.16	78.24	75.90	**81.65**	79.22
OptDigit	97.76	96.98	98.18	**99.31**	98.42
Contrace	69.85	69.01	71.79	**75.01**	72.44
Dermatology	97.65	97.37	98.41	98.68	**98.72**
Sonar	73.29	69.58	72.57	**75.21**	74.79
Gisette	85.91	78.53	85.93	**88.14**	86.99
Krkopt	69.04	69.09	68.16	**72.15**	71.44

similarity between training and test examples. For HLSVM, the similarity measure $S(\boldsymbol{x}_{l+s}, \boldsymbol{x}_i)$ is equal to 1 if the training example $\boldsymbol{x}_i$ belongs to the K-nearest neighbor list of the test example $\boldsymbol{x}_{l+s}$, and 0 otherwise. For PSVM, the number of clusters κ is another parameter that must be determined. Throughout our experiments, the parameter values are selected using ten-fold cross validation on the training set. For example, the parameter K for the KNN classifier is chosen based on the number of nearest neighbors that yields the highest accuracy according to ten-fold cross validation on the training set. The same approach is also used to determine the parameters σ, C, K, ϵ and/or κ for nonlinear SVM, SLSVM, HLSVM, and PSVM. For our experiments, κ usually takes a value between $\sqrt{l}/2$ and $\sqrt{l}$.

B. Comparison of Classification Accuracy

The experimental results reported in this study are obtained by applying five-fold cross validation on the data sets. To make the problem more challenging, we use one-fifth of the data for training and the remaining four-fifths for testing. Each experiment is also repeated ten times and the accu-

Table 3.4. Regression R^2 for SVR, KNNR, HLSVR (KNN-SVR), SLSVR, and PSVR on the 8 UCI datasets.

Data	SVR	KNNR	HLSVR	SLSVR	PSVR
AutoPrice	0.8934	0.8491	0.9166	**0.9221**	0.9209
AutoMpg	0.9425	0.9592	0.9580	**0.9681**	0.9643
Cancer	0.9126	0.8600	0.9423	0.9491	**0.9584**
Concrete	0.9039	0.9396	0.9473	**0.9477**	0.9424
Hardware	0.8274	0.8147	0.9078	0.9219	**0.9262**
Servo	0.6625	0.7109	0.7455	**0.7986**	0.7962
Abalone	0.9400	0.9367	0.9422	**0.9438**	0.9424
Stock	0.9757	0.9914	0.9901	0.9910	**0.9963**

racy reported is obtained by averaging the results over ten trials. Table 3.3 summarizes the results of our experiments. First, observe that, for most of the data sets (14 out of 18), nonlinear SVC outperforms the KNN algorithm. One example is the "Physics" data set, in which the accuracy for SVC is 82.98% whereas the accuracy for KNNC is considerably lower at 67.82%. Second, the accuracy for HLSVC does not seem to show significant improvement over nonlinear SVC. In fact, the accuracy for HLSVC is worse than nonlinear SVC for the 8 of the 18 data sets: "Glass", "Iris", "Car", "Sonar", "Ecoli", "Arrhyth" and "Krkopt". For "Iris", the classification accuracy drops from 92.75% (for nonlinear SVC) to 74.71% (for HLSVC). One possible explanation for HLSVC's poor performance is the difficulty in choosing the right number of nearest neighbors when there are limited training examples available. Unlike HLSVC, the SLSVC algorithm consistently outperforms nonlinear SVC. With the exception of "KDD Cup", the difference observed in their classification accuracies can be shown to be statistically significant based on Student's t-test. This should not come as a surprise because SVC can be considered as a special case of SLSVC by setting the kernel width of similarity matrix $\boldsymbol{S}$ to ∞. Finally, we observe that PSVC, which is an efficient implementation of LSVC, achieves comparable accuracy as SLSVC but still outperforms nonlinear SVC for all the data sets.

C. Comparison of Regression Performance

Table 3.4 summarizes the results of regression performance on 8 data sets. It is observed that SLSVR outperforms nonlinear SVR, KNNR, and HLSVR on most data sets with significantly higher R^2. The only exception is the "Stock" data set, where KNNR seems to perform slightly

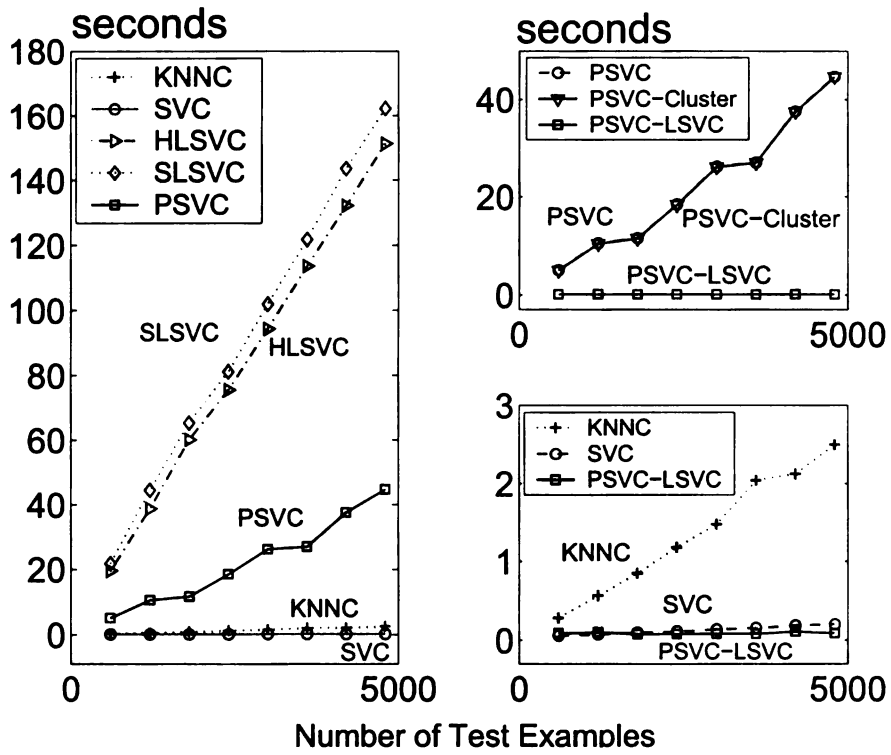


Figure 3.9. The left panel shows the runtime comparison among SVC, KNNC, HLSVC, SLSVC, and PSVC in terms of classification task. The right top panel shows the amount of time spent for MagKmeans clustering by PSVC (i.e., PSVC-Clustering) as opposed to the amount of time spent for training LSVC and applying them to the test examples (i.e., PSVC-LSVC). The right bottom panel compares the time spent by each algorithm to predict the class labels of test examples.

better than SLSVR. HLSVR performs better than KNNR and SVR on 6 of the 8 data sets, however, it is worse than SLVSR on all the data sets. The performance of PSVR is comparable with SLSVR and better than other methods on most data sets except the "Concrete". PSVR even outperforms SLSVR on the data set "Cancer", "Hardware" and "Stock". For example, the R^2 for PSVR is 0.9584 on "Cancer", which is higher than 0.9491 for SLSVR.

D. Runtime Comparison

The purpose of this experiment is to compare the efficiency of LSVM against PSVM. Recall from Section 3.2 that the main limitation of LSVM is its high computational cost since a unique LSVM model must be constructed for each test example. PSVM attempts to overcome this limitation by partitioning the training examples into a small number of clusters and building a linear SVM model for each cluster. Figure 3.9 shows the computational time (in seconds) for training and

testing different classification models using the "Physics" data set. To evaluate its performance, we choose the two largest classes of the data set and randomly sample 600 records from each class to form the training set. We then apply LSVC and PSVC separately on the data set, while varying the number of test examples from 600 to 4800. The diagram on the left panel of Figure 3.9 shows the total time spent by each algorithm for building models and testing. Notice that the time consumed by SLSVC, HLSVC, and PSVC grows linearly with the number of test examples. However, the run times for both HLSVC and SLSVC are considerably longer than PSVC. We may further decompose the runtime for PSVC into two parts: PSVC-Cluster, which is the time needed to apply the MagKmeans clustering algorithm, and PSVC-LSVC, which is the time needed to build a separate model for each cluster. The diagram on the top right panel of Figure 3.9 shows that most of the computational time for PSVC is spent on clustering. If the clustering time is excluded, the remaining time needed to build a linear SVC model for each cluster as well as the time to evaluate each test example is considerably shorter than the overall training and testing times for nonlinear SVC, as demonstrated in the bottom right panel of Figure 3.9. Although we only show the results for the classification task here, regression is expected to behave similarly since it uses a similar optimization procedure.

E. Summary

The SLSVM algorithm generally outperforms both SVM and KNN but at the expense of longer computational time. PSVM helps to improve its computational efficiency, while achieving comparable accuracy as the SLSVM algorithm.

3.5.3 Application of LSVM and PSVM to Earth Science Data

In this experiment, we applied our LSVM and PSVM algorithms to Earth Science data sets with spatial and temporal dependencies. Four prediction tasks are considered, two for classification and another two for regression. All these four data sets are downloaded from the UCI machine learning repository [154]. A summary of the data sets and their associated prediction tasks is provided in Table 3.5, while the details are described as follows.

Table 3.5. Description of data sets used for prediction with spatial and temporal dependencies.

Data	# Instances	# Attributes	Prediction Task	Dependencies
CoverType	581012	54	Classification	Spatial
Ozone	2536	73	Classification	Temporal
Elnino	178080	12	Regression	Spatio-temporal
ForestFire	517	13	Regression	Spatio-temporal

- The CoverType data set is used to predict the types of forest cover [22] of an unknown region. It is an important classification task to help natural resource managers obtain basic descriptive information of forested lands to support their decision-making ecosystem management strategies. A set of cartographic measures are used as predictor variables such as soil type, hill shade, slope, etc. Spatial information such as the distance from a region to water surface or roadway is available to be incorporated to the LSVM models, which makes it a **spatial prediction** problem.

- The Ozone data set is used to classify the days in which the ground-level ozone reaches a dangerous level [224]. Studies have shown that elevated ground-level ozone is harmful to human health, e.g. it causes asthma, chest pain, coughing, and decreases in lung function [71]. It also has negative effects on vegetation and ecosystems, leading to reductions in agricultural and commercial forest yields, and increases in plant's susceptibility to disease, pests, and other environmental stresses. A number of climate variables such as wind speed, solar radiation, temperature, and precipitation are used as the predictor variables. The temporal neighborhood is defined based on the day and month in which the measurement was taken. As a result,it is a **temporal prediction** problem.

- The Elnino data set is used to predict the sea surface temperature [124] in equatorial Pacific. It is a regression task that can help in the understanding and prediction of El Nino/Southern Oscillation (ENSO) cycles, which caused many problems throughout the world, e.g. destructive flooding from increased rainfalls in Peru and the Unites States and the drought and devastating brush fires in western pacific areas. The data set contains oceanographic and surface meteorological attributes such as wind speed, humidity etc., which are recorded from a series of buoys positioned throughout the equatorial pacific. Both spatial (latitude and longitude of the locations for bouys) and temporal information (day/month/year read

Table 3.6. Classification accuracies (%) for SVC, KNNC, HLSVC (KNN-SVC), SLSVC, and PSVC (using spatial or temporal dependencies) on the 2 Earth Science data: CoverType and Ozone.

Data	SVC	KNNC	HSVC	SLSVC	PSVC
CoverType	86.21	67.40	73.33	89.12	**89.43**
Ozone	93.35	92.30	91.56	**94.80**	94.38

Table 3.7. Regression R^2 for SVR, KNNR, HLSVR (KNN-SVR), SLSVR, and PSVR (using spatial or temporal dependencies) on the 2 Earth Science data: Elnino and Forest Fire.

Data	SVR	KNNR	HSVR	SLSVR	PSVR
Elnino	0.9215	0.9692	0.9437	0.9817	**0.9893**
ForestFire	0.0810	0.0034	0.0914	**0.1124**	0.1103

from the bouys) is available to be incorporated into LSVM models, which makes it a **spatio-temporal prediction** problem.

- The ForeFire data set is used to predict the burned area of possible forest [54] fires for a given region. It is a regression task that can help to reduce the damages that can appear if fire occurs by improving the organization of prevention measures and storage of firefighting resources. Meteorological information such as wind speed, relative humidity, temperature etc. are available as the predictor variables. Both spatial (x,y spatial coordinate) and temporal (day of the week, month of the year) are used for "localization" in LSVM model, which makes it a **spatio-temporal prediction** problem.

Table 3.6 presents the classification results for the CoverType and Ozone data sets. For the CoverType data, PSVC performs the best with highest accuracy 89.43%. SLSVC performs better than nonlinear SVC, KNNC and HLSVC. For the Ozone data, SLSVC is the best with 94.80% accuracy. PSVC is slightly worse but still comparable to SLVC, and it is better than SVC, KNNC and HLSVC. Table 3.7 shows the R^2 of regression on the Elnino and ForestFire data sets. PSVR performs the best with $R^2 = 0.9893$ on the Elnino data set. SLSVR performs the best on the forest fire data set with $R^2 = 0.1124$, while PSVR second it with comparable $R^2 = 0.1103$. Both results shows the advantage of LSVM and PSVM on predicting Earth Science data by incorporating spatial and temporal neighborhood information.

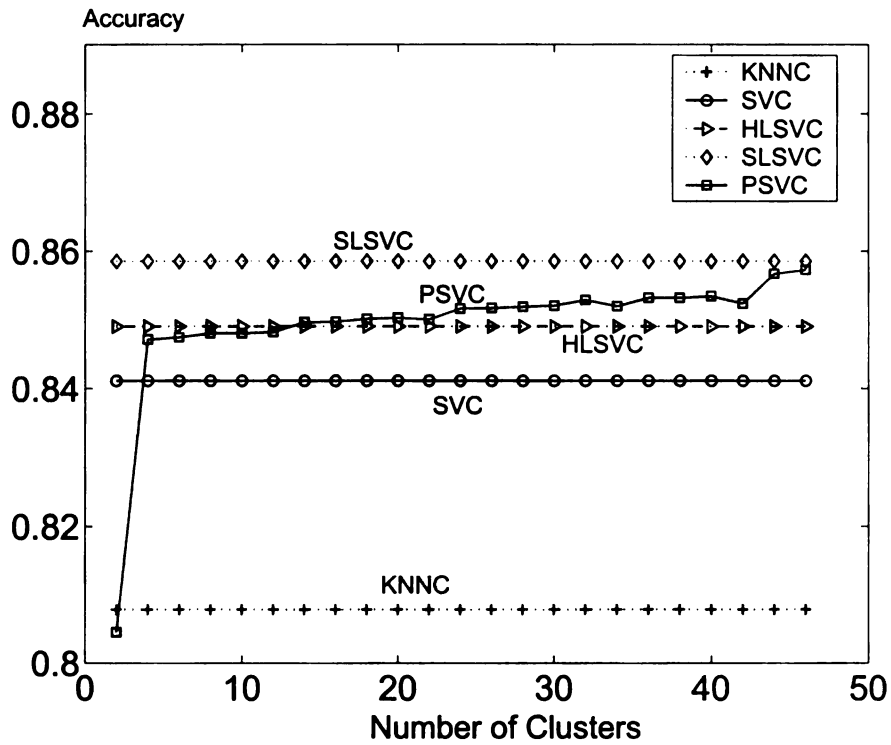


Figure 3.10. Effect of varying the number of clusters κ on the performance of PSVC.

3.5.4 Sensitivity Analysis

The number of clusters κ is an important parameter in PSVM. In this experiment, we perform sensitivity analysis on this parameter using the "Robot" data set. We sampled 4500 records from each class of the data set and varied κ from 2 to 46. For each value of κ, we perform 10-fold cross validation, using one-tenth of the data for training and the rest for testing. The experiment is repeated ten times and we reported their average accuracy. Figure 3.10 shows the accuracies of PSVC for each κ along with those of SVC, KNN, SLSVC, and HLSVC. The accuracy of PSVC increases rapidly until κ reaches 5. PSVC outperforms HLSVC after $\kappa = 15$ and becomes comparable to SLSVC when κ is sufficiently large. We also recorded the computational times for SVC, KNN, HLSVC, SLSVC, and PSVC in Figure 3.11. The results confirmed our earlier observation that PSVC spends considerably less time than SLSVC and HLSVC. Furthermore, the computational time for the training and testing parts of PSVC (PSVC-LSVC) is more efficient than regular SVC when $\kappa \leq 30$, as shown in the right panel of Figure 3.11. Although we only present experiments here for the classification task, our conclusion applied to PSVR too.

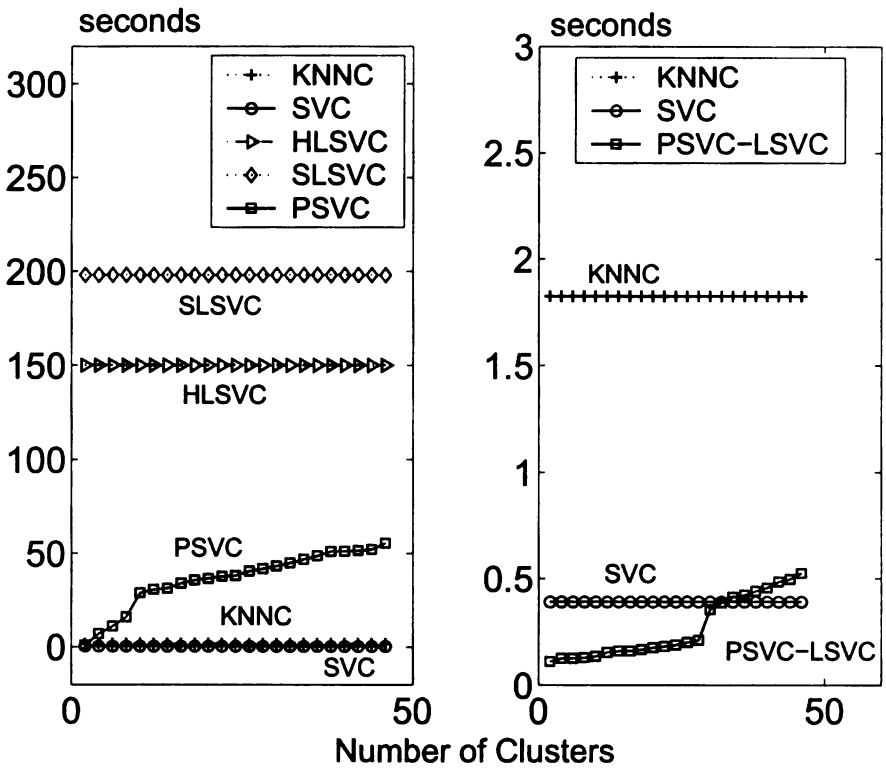


Figure 3.11. Classification runtime comparison among SVC, KNNC, HLSVC, SLSVC, and PSVC when varying the number of clusters κ in PSVC.

3.6 Summary

Prediction tasks such as classification and regression for regular data sets have been studied for decades in statistics, data mining, and machine learning. Prediction with spatial and temporal dependencies remains a challenging problem, especially for data sets that are not evenly distributed across different time periods and locations. In this chapter, we proposed a localized prediction algorithm to address this issue. The main contributions of this work are:

- We proposed a localized Support Vector Machine framework for both classification and regression, which incorporates the neighborhood information between examples into SVM learning and testing such that those training examples that are close to the test example contribute more to the development of local model. This framework has several advantages compared with standard Support Vector Machine and K-nearest neighbor algorithm. First, by building local linear Support Vector Machine model for each test example, the difficulty in choosing an appropriate kernel function for learning global nonlinear Support Vector

Machine is avoided. Second, building multiple local SVM models performs much better than one global SVM model as demonstrated by experiments on a number of real-world data sets.

- However, building a separate SVM model for each test example is computationally infeasible for large scale data sets. To address this limitation, we developed a more efficient implementation of the algorithm called profile Support Vector Machine. PSVM first extracts a small number of clusters from the data set using supervised algorithms called MagK-means and VarKmeans, for classification and regression tasks respectively, which extend the K-means algorithm to maximize the variance of the labels or response values inside each cluster. After applying the supervised clustering step, a local model is trained for each cluster. Our experiments on real data sets showed that PSVM reduces the training cost significantly while maintaining a comparable accuracy to LSVM.

- We further extend the LSVM and PSVM frameworks to spatial and temporal prediction tasks by incorporating the spatial, temporal, or spatio-temporal neighborhood information. The extended methods are applied to Earth Science data sets and the results demonstrate the success of our proposed algorithm with better performance compared with K-nearest neighbor and Support Vector Machine algorithm.

CHAPTER 4

Long-Term Time Series Forecasting

Time series forecasting is a particular prediction task that is concerned with predicting the future values of the time series based on its historical observations or information from other variables. The formal definition of time series prediction is given as follows:

Definition 7. (Time Series Forecasting) *Given a time series* $\boldsymbol{X}_L = [x_1, x_2, ..., x_l]^\top$ *with* l *historical values, the task of time series prediction is to predict its future* u *values* $\boldsymbol{X}_L = [x_{l+1}, x_{l+2}, ..., x_{l+u}]^\top$.

A typical time series prediction technique first constructs a training set (as shown in Table 4.1) by running a sliding window of length $p + u$ (p is the training window and u is the prediction window) along the historical time series as illustrated in Figure 4.1. Then a regression model is built from the training set and is further applied to predict the future u values. When the prediction window u is long, we called it a long term time series forecasting problem.

In this chapter, we investigate the long term time series forecasting problem with applications to climate projection. The main aspects of the work are summarized as follows:

- We first discuss the existing issues in multi-step ahead time series prediction by investigating three prediction frameworks: multi-stage prediction, independent value prediction and parameter prediction. Their advantages and disadvantages are compared with each other from both theoretical and empirical perspectives.

- A semi-supervised multivariate time series prediction method is proposed to alleviate the error accumulation problem in long term time series forecasting. Extensive experiments

Table 4.1. Traning Set $D = X \times Y$

$X' = [X_1, ..., X_p]$	$Y = [Y_1, ..., Y_u]$
$[x_1, x_2, ..., x_p]$	$[x_{p+1}, x_{p+2}, ..., x_{p+u}]$
$[x_2, x_3, ..., x_{p+1}]$	$[x_{p+2}, x_{p+3}, ..., x_{p+u+1}]$
...	...

have been conducted on a variety of data sets to demonstrate the effectiveness of the proposed algorithm compared to other supervised learning methods for long term time series forecasting.

- A covariance alignment method is also developed to deal with the inconsistencies between historical observation data and future simulation data of the predictor variables in climate projection. Experimental results show significant improvement by semi-supervised prediction with data calibration.

The rest of this chapter is organized as follows. Section 4.1 introduces some notations and concepts for time series forecasting. Section 4.2 presents a study on the issues of the long term time series forecasting. Section 4.3 demonstrates the value of unlabeled data for regression. Section 4.4 proposes the semi-supervised multivariate time series prediction algorithm with data calibration technique for long term forecasting. Section 4.5 presents the experimental results on both benchmark and Earth Science data sets.

4.1 Preliminaries

This section presents some background on time series forecasting.

4.1.1 Single Step and Multi-step Time Series Forecasting

Time series prediction can be categorized into single step time series prediction and multi-step time series prediction based on the number of prediction steps. The problem of single step ahead prediction is to the predict only the next value $\boldsymbol{x}_{l+1}$ with $u = 1$ while the multi-step time series prediction problem is to predict the future $u > 1$ values $[x_{l+1}, x_{l+2}, ..., x_{l+u}]^\top$. Multi-step time series prediction with long prediction window is identified as the long term time series forecasting problem. In long term time series forecasting, the length of prediction window can be as long as

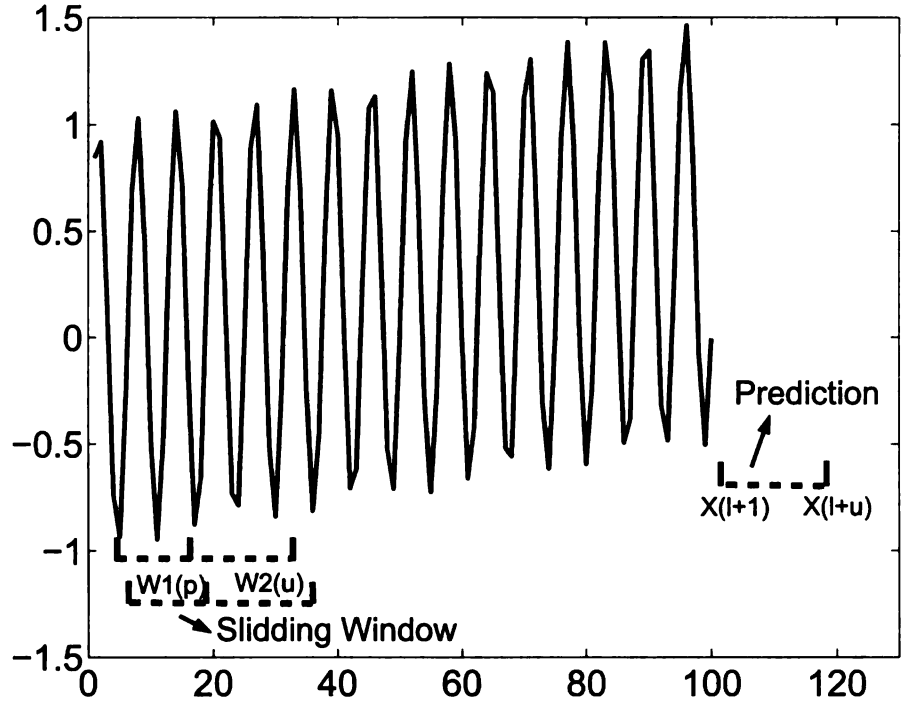


Figure 4.1. A sliding window is used to create the regression training set D=X'+Y.

the length of historical data or even much longer. The prediction error for multi-step ahead time series forecasting is evaluated using the R-square method described in Equation (3.26).

4.1.2 Univariate and Multivariate Time Series Forecasting

Time series forecasting can also be categorized into univariate time series prediction and multivariate time series prediction. Univariate time series forecasting [29] predicts the future values of a variable $[x_{l+1\ldots l+u}]^\top$ by building an auto-regression model f on its own historical values $[x_{1\ldots l}]^\top$ such as:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-p}) \tag{4.1}$$

As illustrated in Figure 4.2, multivariate time series prediction [138] predicts the future values of the target time series by utilizing not only its own historical values but also information from other related time series. A formal definition of multivariate time series prediction is given as:

Definition 8. (Multivariate Time Series Prediction) *Let $\mathcal{D}_L = \{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$ be a multivariate time series of length l and p predictor variables, the objective of multivariate time series prediction*

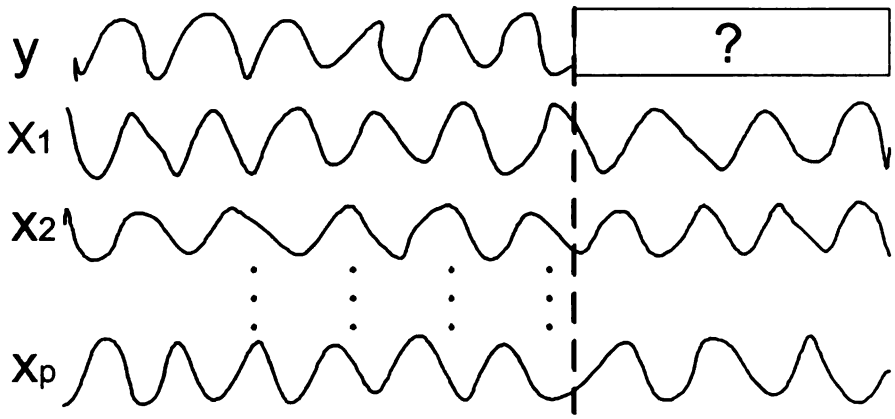


Figure 4.2. Multivariate time series prediction.

is to predict the future values of the response variable $\boldsymbol{X}_U$.

In the above definition, $\boldsymbol{X}_L = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_l]^\top$ is a p-dimensional sequence of values for the predictor variables, which can also be represented as $\boldsymbol{X}_L = [\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_p]$. $\boldsymbol{Y}_L = [y_1, y_2, ..., y_l]^\top$ is the corresponding values for the response variable. Multivariate time series prediction learns a target function f from the historical data $\boldsymbol{\mathcal{D}}_L$ as:

$$y_t = f(x_{1t}, x_{2t}, \ldots, x_{pt}, \cdots) \tag{4.2}$$

The model is applied to the future unlabeled data $\boldsymbol{X}_U = [\boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, ..., \boldsymbol{x}_{l+u}]^\top$ and predicts the future values of the response variable, $\boldsymbol{Y}_U = [y_{l+1}, y_{l+2}, ..., y_{l+u}]^\top$.

4.1.3 Regression Models for Prediction

Before we get into the problem of long term time series forecasting, we first introduce three widely used regression techniques: Multiple Linear Regression, Recurrent Neural Networks and Hidden Markov Regression. These methods can be used for both univariate and multivariate time series prediction.

A. Multiple Linear Regression

Multiple Linear Regression (MLR) models the relationship between the response variable y and the predictor variables $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_p]$ using a linear equation:

$$y = \sum_{i=1}^{p} w_i \boldsymbol{X}_i + w_0 + \epsilon$$

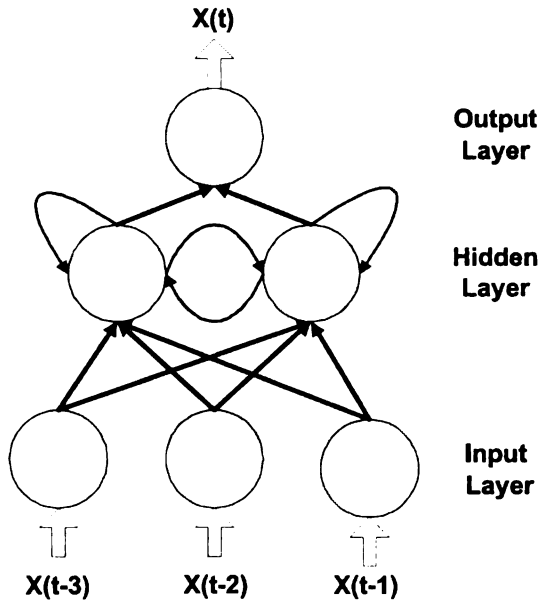


Figure 4.3. A simple example of Elman Network to model AR(3) model for univariate time series.

, where ϵ corresponds to a random noise term with zero mean and variance σ^2. The coefficient vector $\mathbf{w}$ is estimated using the least square method by minimizing the sum of squared error on the training data with solution:

$$\mathbf{w} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

The variance is estimated using SSE/u, where u is the size of the prediction window. For univariate time series prediction, we use x_t as the response variable and $\left[x_{t-1}, x_{t-2}, ..., x_{t-p}\right]^\top$ as the predictor variables, which corresponds to the auto-regression model. For multivariate time series prediction, the predictor variables come from other related time series.

B. Recurrent Neural Networks

Recurrent Neural Networks (RNN) has been successfully applied to noisy and non-stationary time series prediction. In RNN, the temporal relationship of the time series is explicitly modeled using feedback connections [91] to the internal nodes (known as hidden units). An RNN model is trained by presenting the past values of the time series to the input layer of the Elman back propagation network [73]. The weights of the network are then adjusted based on the error between the true output and the output predicted by the network until the algorithm converges. Figure 4.3 illustrates a simple Elman network of AR model with order 3 for modeling a univariate time series. Before

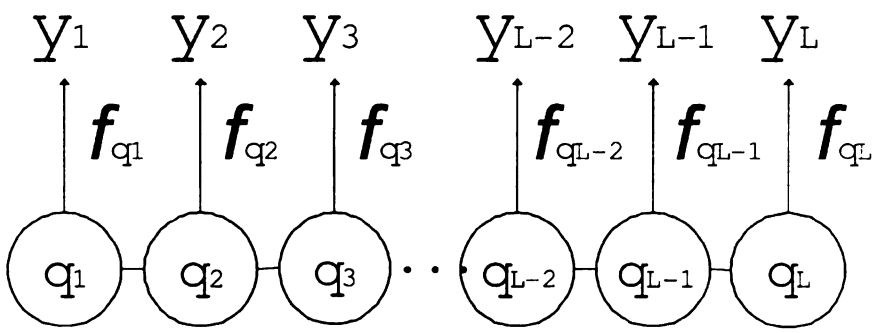


Figure 4.4. Hidden Markov Regression Model.

the network is trained, the user must specify the number of hidden units in the network and the stopping criteria of the learning algorithm. RNN have the strength of modeling complex nonlinear regression models.

C. Hidden Markov Regression

In *Hidden Markov Model Regression*, a time series is assumed to be generated by a doubly stochastic process, where the response variable is conditionally dependent on the values of the predictor variables as well as an underlying unobserved process. The unobserved process is characterized by the state space $\mathbf{\Gamma} = \{s_1, s_2, \cdots, s_N\}$ and is assumed to evolve according to a Markov chain, $\boldsymbol{Q}_L = [q_1, q_2, .., q_l]^\top$, where $q_i \in \mathbf{\Gamma}$ (as shown in Figure 4.4). The transition between states is governed by a $N \times N$ transition probability matrix $\boldsymbol{A} = [a_{ij}]$. Each state $s_i \in \mathbf{\Gamma}$ is associated with an initial probability distribution π_i and a regression function $f_i(\boldsymbol{x}_t)$. For brevity, we consider a multiple linear regression model, $f_i(\boldsymbol{x}_t) = \mathbf{w}_i \boldsymbol{x}_t^\top + \sigma_i \epsilon_t$, where $(\mathbf{w}_i, \sigma_i)$ are the regression parameters, and $\epsilon_t \sim N(0,1)$. Given a set of predictor variables $\boldsymbol{x}_t$ at time t, the response variable y_t is generated based on the following conditional probability:

$$p_{q_t}(y_t|\boldsymbol{x}_t) = (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[-\frac{(y_t - \mathbf{w}_{q_t}\boldsymbol{x}_t^\top)^2}{2\sigma_{q_t}^2}\right] \tag{4.3}$$

The likelihood function for the sequence of labeled observations in $\boldsymbol{\mathcal{D}}_L = \{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$ is given

by

$$
\begin{aligned}
L &= \sum_{\boldsymbol{Q}_L} \prod_{t=1}^{l} p_{q_t}(y_t|\boldsymbol{x}_t) p(q_t|q_{t-1}) \\
&= \sum_{\boldsymbol{Q}_L} \prod_{t=1}^{l} a_{q_{t-1}q_t} (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[-\frac{(y_t - \mathbf{w}_{q_t}\boldsymbol{x}_t^\top)^2}{2\sigma_{q_t}^2} \right]
\end{aligned}
$$

The model parameters

$$
\boldsymbol{\Lambda} = \Big(\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{\Sigma}, \boldsymbol{W} \Big) = \Big(\{\pi_i\}_{i=1}^N, \{a_{ij}\}_{i,j=1}^N, \{\sigma_i\}_{i=1}^N, \{\mathbf{w}_i\}_{i=1}^N \Big)
$$

are estimated by maximizing the above likelihood function using the Baum-Welch (BW) algorithm [14]. The model parameters can be estimated iteratively using the following quantities, which are known as the forward (α) and backward (β) probabilities:

$$
\begin{aligned}
\alpha_t(i) &= p(y_1, y_2, \cdots, y_t, q_t = i|\boldsymbol{X}_L) \\
&= \begin{cases} \pi_i p_i(y_1|\boldsymbol{x}_1), & \text{if } t = 1; \\ \left[\sum_{j=1}^N \alpha_{t-1}(j) a_{ji}\right] p_i(y_t) & \text{otherwise.} \end{cases} \qquad (4.4) \\
\beta_t(i) &= p(y_{t+1}, y_{t+2}, \cdots, y_l|q_t = i, \boldsymbol{X}_L) \\
&= \begin{cases} 1, & \text{if } t = l; \\ \sum_{j=1}^N a_{ij} p_j(y_{t+1}) \beta_{t+1}(j), & \text{otherwise.} \end{cases} \qquad (4.5)
\end{aligned}
$$

Further details on the derivation of the parameter update formula using these quantities can be found in [85]. Upon convergence of the BW algorithm, the HMMR model can be applied to the unlabeled data $\boldsymbol{X}_U$ in the following manner. First, the probability of each hidden state at a given future time step is estimated as follows:

$$
p(q_{l+m} = s_k|\boldsymbol{Y}_L) = \begin{cases} \frac{\sum_i \alpha_l(i) a_{ik}}{\sum_i \alpha_l(i)}, & \text{if } m = 1; \\ \sum_i a_{ik} p(q_{l+m-1} = s_i|\boldsymbol{Y}_L), & 1 < m \leq u. \end{cases}
$$

Next, the future value of the response variable is estimated using the following equation:

$$
f_{q_t}(\boldsymbol{x}_t) = E[y_t] = \sum_i (\mathbf{w}_i \boldsymbol{x}_t^\top) p(q_t = s_i|\boldsymbol{Y}_L) \qquad (4.6)
$$

4.2 Problem Study on Long-Term Time Series Prediction

In this section, we studied the issues for long term time series prediction using traditional prediction methods. Three approaches: multi-stage prediction, independent value prediction, and parameter prediction, are used to solve the multi-step ahead time series prediction problem and compared with each other. These methods are described as follows:

Multi-stage Prediction is a direct extension of single step time series prediction. It predicts the future values of a time series in a step by step manner. We first predict x_{l+1} using the previous p values, $[x_{l+1-p}, ..., x_{l-1}, x_l]^\top$. We then predict x_{l+2} based on its previous p values, which includes the predicted value for x_{l+1}. The procedure is repeated recursively until the last value, x_{l+u}, has been estimated. In this approach, it is sufficient to construct a single model for making the prediction.

Independent Value Prediction predicts the value at each time step using a separate model. Given the initial data set shown in Table 4.1, we first create u training sets, each of which has the same input X, but different output Y. We use $Y(1)$ as the output variable for the first training set, $Y(2)$ as the output variable for the second training set, and so on. By learning each training set independently, we obtain u regression models $\{f_i, i = 1, 2, \cdots, u\}$. The models are then used to predict the next u values as follows: $x_{t+i} = f_i(X), i = 1, 2, ..., u$.

Parameter Prediction transforms the problem of predicting u output values into an equivalent problem of predicting $d + 1$ parameters. For each record in Table 4.1, we fit a parametric function g to the output vector Y. Let $[c_0, c_1, ..., c_d]$ denote the parameters of the function g. We then replace the original output vector $Y = [Y(1), Y(2), ..., Y(u)]$ with a modified output vector $Y' = [c_0, c_1, ..., c_d]^\top$. We now construct $d + 1$ regression models $\{f_i, i = 0, 1, 2, \cdots, d\}$ one for each output column Y'. The models are then applied to predict the $d + 1$ parameters of a test sequence. The test sequence is reconstructed by substituting the predicted parameters into the parametric function g. While this methodology is generally applicable to any family of parametric functions, we use polynomial functions in our experiments.

4.2.1 Error Accumulation

Error accumulation refers to the propagation of past prediction errors into future predictions. To gain a better insight into this problem, we employ the bias-variance decomposition for squared loss functions. Consider a time series generated by the model:

$$x_t = f(x_{t-1}, x_{t-2}, \cdots, x_{t-p}) + e(0, \sigma^2)$$

Let $[x_1, x_2, \ldots, x_l]^\top$ denote the historical values and $[y_{l+1}, y_{l+2}, \ldots, y_{l+u}]^\top$ denote the future observed values, and $\left[y^*_{l+1}, y^*_{l+2}, \ldots, y^*_{l+u}\right]^\top$ be the values generated by the deterministic model f. In other words,

$$y_i = y_i^* + e(0, \sigma^2).$$

We use the notation $[\hat{y}_{l+1}, \hat{y}_{l+2}, \ldots, \hat{y}_{l+u}]^\top$ to denote the values predicted by a regression model g. The MSE (mean square error) at each step can be decomposed into the following three components[26]:

$$\begin{aligned} MSE(j) &= E[(y_j - \hat{y}_j)^2] \\ &= (E(\hat{y}_j) - y_j^*)^2 + E[(\hat{y}_j - E(\hat{y}_j))^2] + E[(y_j - y_j^*)^2] \end{aligned}$$

The first term represents the squared bias of the model. The second term represents the variance of the model. The third term represents the variability due to noise. The following example illustrates the error accumulation problem for the noise term. Consider AR(2) model:

$$x_{t+1} = a_1 x_t + a_2 x_{t-1} + \epsilon,$$

with $\epsilon \sim N(0, \sigma^2)$. Suppose we were able to model accurately the coefficients using MLR. For the multi-stage approach, we have:

$$\begin{aligned} y_{l+1} &= a_1 x_l + a_2 x_{l-1} + \epsilon_1 = \hat{y}_{l+1} + \epsilon_1 \\ y_{l+2} &= a_1 y_{l+1} + a_2 x_l + \epsilon_2 = \hat{y}_{l+2} + (a_1 \epsilon_1 + \epsilon_2) \end{aligned}$$

For independent value prediction, we have

$$\begin{aligned} y_{l+1} &= a_1 x_l + a_2 x_{l-1} + \epsilon_1 \\ y_{l+2} &= (a_1^2 + a_2) x_l + (a_1 a_2) x_{l-1} + (a_1 \epsilon_1 + \epsilon_2) \end{aligned}$$

The preceding example shows the accumulation of errors due to noise for both multi-stage prediction and independent value prediction as the prediction step increases. Thus it is unavoidable.

To analyze the error accumulation due to the bias and variance of a model, we generate the following time series: $X_t = 0.418X_{t-1} + 0.634X_{t-2} + \epsilon$, $\epsilon \sim N(0, 0.1)$. The length of the time series is set to 1000 and the prediction window is $u = 50$. To ensure there is sufficient bias in the model, we set $p = 1$. The bias and variance of the models are estimated by generating 500

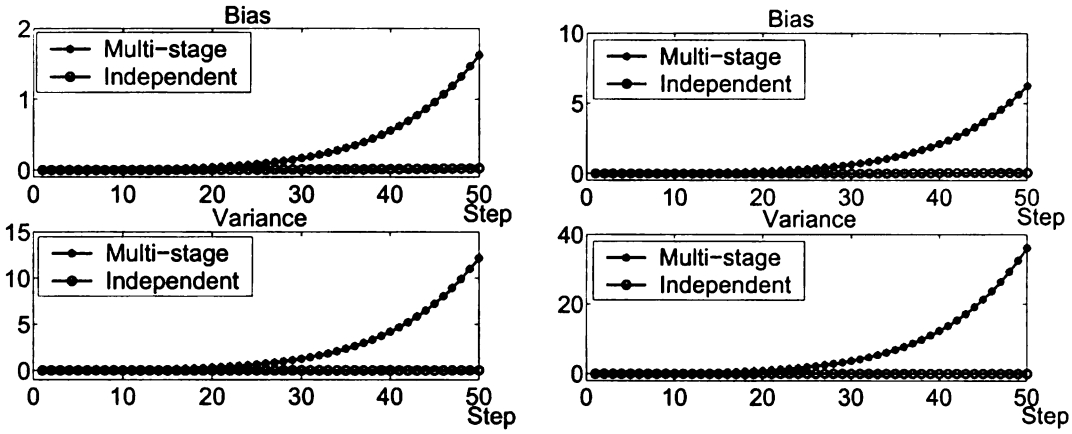


Figure 4.5. Bias and variance for MLR (left) and HMMR (right).

bootstrap replicates of the training set $\mathcal{D}$ and inducing a model g from each bootstrap replicate. The models are then applied to the test sequence to obtain 500 estimated values $\hat{y}_j$ for each prediction step j. The empirical bias is computed by taking the average value of the 500 predictions $E(\hat{y}_j)$ and subtracting it from the value predicted using the deterministic model. The variance of the models can also be estimated as follows: $var(j) = E[(\hat{y}_j - E(\hat{y}_j))^2]$. Figures 4.5 illustrate the bias and variance for multi-stage and independent value predictions (using MLR and a HMMR as the underlying regression methods). Both figures show that the bias and variance for multi-stage prediction grows steadily with increasing time steps, whereas the bias and variance for independent value prediction do not appear to be propagated into future predictions.

4.2.2 Learning Difficulty

Multi-stage prediction builds a single model to fit the entire time series. In contrast, we need to build u models for independent value prediction and $d+1$ models for parameter prediction, which means they are more expensive.

For parameter prediction, the learning difficulty depends on how easy it is to find the appropriate function that fits the output vector. To compare parameter prediction against independent value prediction, we apply them to the monthly milk production data using $u = 12$ and $w = 12$. For parameter prediction, we use a polynomial function to fit the output vector and vary the degree of the polynomial from 0 to 11. We then employ MLR to predict the parameters of the polynomial. The bottom diagram of Figure 4.6 shows the results of $1 - R^2$ for different degree. Observe that

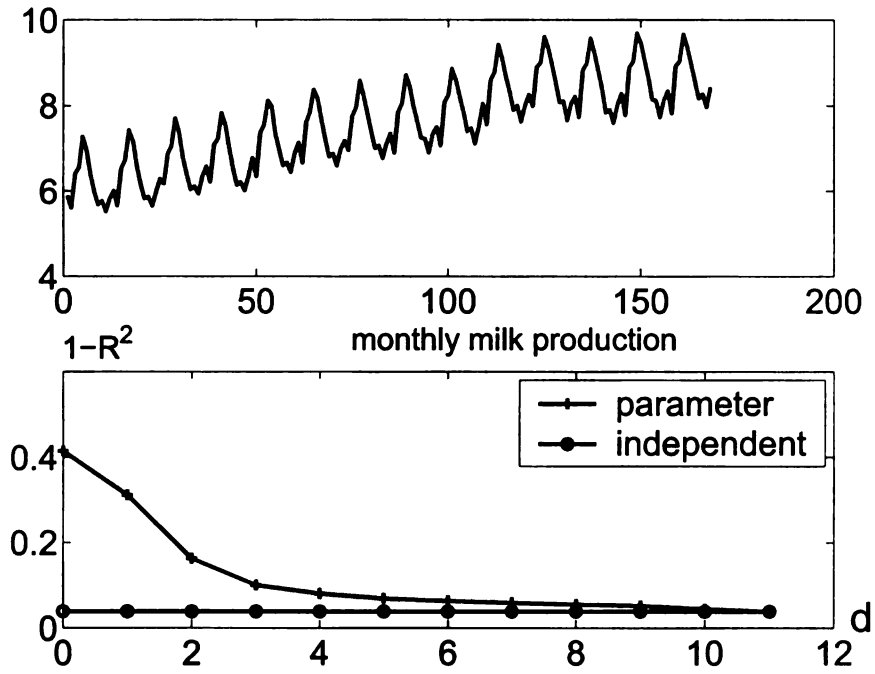


Figure 4.6. Prediction Results (p=12,u=12).

the $1 - R^2$ for parameter prediction drops dramatically when the polynomial degree increases to 3 and decreases slowly thereafter. This results suggest that it is sufficient to fit a polynomial of degree 4 to the output vector and achieves comparable accuracy as independent value prediction.

4.2.3 Smoothness of Prediction

Another factor to consider is the influence of noise on the prediction approaches. To do this, we conduct an experiment using a simple, stationary time series, i.e., white noise, as shown in the left panel of Figure 4.7. The right panel of Figure 4.7 shows that multi-stage prediction tends to smooth out the time series to its mean value after p time steps. Such a smoothing effect is not present in independent value prediction, which makes spurious predictions around the mean, because the prediction at each time step is modeled independently. This method may suffer from overfitting as it tries to capture the fluctuations of the noise time series. For parameter prediction, the best fit model of the data is found to be a polynomial of degree zero. Even though the parameters are predicted independently, the smoothness of the time series is guaranteed by the parametric function used to fit the output vector.

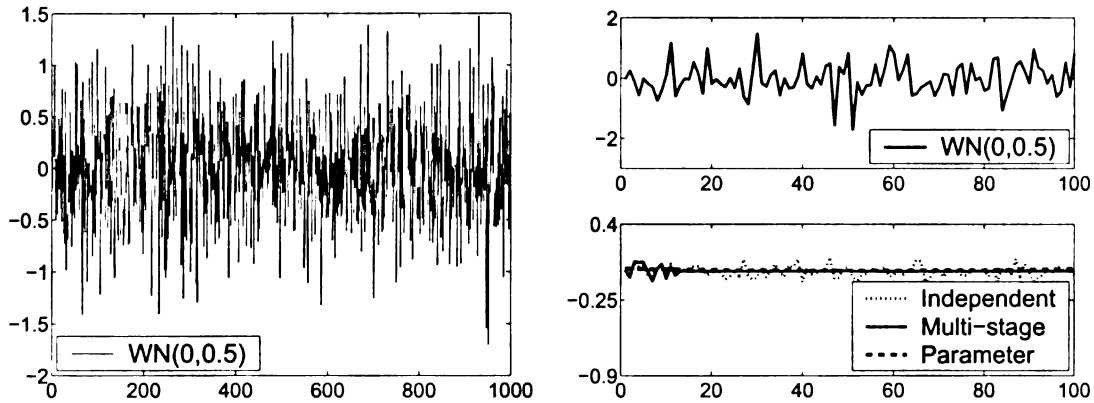


Figure 4.7. White Noise WN(0,0.5) (left) and predicting Results (p=12,u=100,d=6)(right).

4.2.4 Discussion

The study in this section discovered that present multi-stage prediction framework suffers from error accumulation problems when the prediction period is long and the historical data is relatively short. Independent value prediction and parameter prediction can slightly alleviate this problem; however, they have their own issues such as difficulty in learning and sensitive to noise. Their performances are further compared in Section 4.5.2 on a number of real data sets.

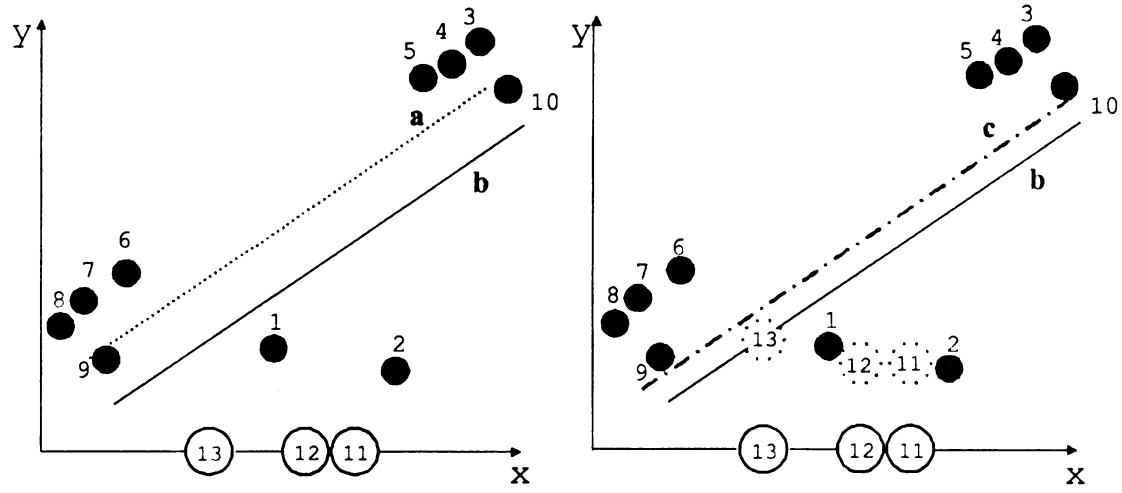
Fortunately, there are alternative ways to improve long term time series forecasting by multi-variate prediction that utilizes information from other related time series. Future data of predictor variables can be generated by simulating potential scenarios using computer-driven models, such as global climate models. However, the data generated by these models are currently utilized in a supervised learning setting, where a predictive model trained on past observations is used to estimate the future values. In the rest of this chapter, we present a semi-supervised learning framework for long-term time series forecasting based on Hidden Markov Model Regression. A covariance alignment method is also developed to deal with the issue of inconsistencies between historical and model simulation data. We evaluated our approach on data sets from a variety of domains, including climate modeling. We start by discussing the value of unlabeled data on regression in the next Section.

4.3 Values of Unlabeled Data on Regression

There have been extensive studies on the effect of incorporating unlabeled data to supervised classification problems, including those based on generative models [56], transductive SVM [112], co-training [25], self-training [229] and graph-based methods [24][230]. Some studies concluded that significant improvements in classification performance can be achieved when unlabeled examples are used, while others have indicated otherwise [25][51][57][199]. Blum and Mitchell [25] and Cozman et al. [51] suggested that unlabeled data can help to reduce variance of the estimator as long as the modeling assumptions match the ground truth data. Otherwise, unlabeled data may either improve or degrade the classification performance, depending on the complexity of the classifier compared to the training set size [57]. Tian et al. [199] showed the ill effects of using different distributions of labeled and unlabeled data on semi-supervised learning.

Recently, there has been growing interest on applying semi-supervised learning to regression problems [228][30][53][231]. Some of these approaches are direct extensions from their semi-supervised classification counterparts. For example, transductive support vector regression is proposed in [53] as an extension to transductive SVM classifier. Zhou and Li developed a co-training approach for semi-supervised regression in [228]. Their algorithm employs two KNN regressors, each using a different distance metric. Another extension of co-training to regression problems was developed by Brefeld et al. [30]. Graph based semi-supervised algorithms [24][230] utilize a label propagation process to ensure that the smoothness assumption holds for both labeled and unlabeled data. An extension of the algorithm to regression problems were proposed by Wang et al. in [210]. Zhu and Goldberg [231] developed a semi-supervised regression method that incorporates additional domain knowledge to improve model performance. Since all of the previous approaches ignore the temporal dependencies between observations, they are not well-suited for time series prediction problems.

This section provides an example to illustrate the value of unlabeled data for regression analysis. Consider the diagram shown in Figure 4.8, where the x-axis corresponds to a predictor variable and the y-axis corresponds to the response variable. The data set contains 10 training examples (labeled $1 - 10$) and 3 unlabeled examples (labeled $11 - 13$). The diagram on the left shows the results of applying supervised linear regression while the diagram on the right shows the results



(a) Supervised learning (b) Semi-supervised learning

Figure 4.8. The value of unlabeled data for regression analysis

of applying semi-supervised linear regression. The solid line b indicates the true function from which the data is generated.

The dashed line a in the left diagram corresponds to the target function estimated from training examples. In the right diagram, the response values for the unlabeled data are initially computed using the values of their nearest neighbors. The target function, represented by the dashed line c, is then estimated from the combined training and previously unlabeled examples. Clearly, augmenting unlabeled data in this situation helps to produce a target function that lies closer to the true function. This example also illustrates the importance of using local prediction methods (such as nearest neighbor estimation) to compute the response values of the unlabeled examples. If the unlabeled examples were estimated using the initial regression function instead, then adding unlabeled data will not change the initial model.

It is worth noting that current research in semi-supervised learning suggests unlabeled data are valuable under two situations. First, the model assumptions should match well with the underlying data. Second, the labeled and unlabeled data must be generated from the same distribution. Otherwise, incorporating unlabeled data may actually degrade the performance of a predictive model [199][57].

4.4 Semi-supervised Multivariate Time Series Prediction

This section describes our proposed semi-supervised learning algorithm for HMMR and a data calibration approach to deal with inconsistencies between the distributions of labeled and unlabeled data.

4.4.1 Semi-Supervised HMMR for Long Term Time Series Forecasting

The basic idea behind our proposed approach is as follows. First, an initial HMMR model is trained from the historical data using the BW algorithm described in Section 4.1.3. The model is then used, in conjunction with a local prediction model, to estimate the response values for future observations. The local model is used to ensure that the target function is sufficiently smooth. The estimated future values, weighted by their confidence in estimation, are then combined with the historical data to re-train the model. This procedure is repeated to gradually refine the HMMR model.

We now describe the details of each step of our algorithm. Let Λ^0 be the initial set of model parameters trained from the historical observations $\mathcal{D}_L$. We use Λ^0 to compute the initial estimate of each response value in $\boldsymbol{Y}_U$:

$$\overline{y}_t = \sum_i p(q_t = s_i|\boldsymbol{Y}_L)\mathbf{w}_i\boldsymbol{x}_t^\top,\ t = l+1, ..., l+u,$$

which is similar to the formula given in Equation (4.6).

A principled way to incorporate unlabeled data is to require that the resulting target function must be sufficiently smooth with respect to its intrinsic structure [227]. For regression problems, this requirement implies that the response values for nearby examples should be close to each other to ensure the smoothness of the target function. We obtain an estimate of the local prediction of the target variable y for a future time step t as follows:

$$\widetilde{y}_t = \frac{\sum\limits_{\boldsymbol{x}_j \in \Omega_k(\boldsymbol{x}_t)} \boldsymbol{S}(\boldsymbol{x}_t, \boldsymbol{x}_j) y_j}{\sum\limits_{\boldsymbol{x}_j \in \Omega_k(\boldsymbol{x}_t)} \boldsymbol{S}(\boldsymbol{x}_t, \boldsymbol{x}_j)},\ t = l+1, \ldots, l+u$$

where $\Omega_K(\boldsymbol{x}_t)$ is a subset of observations in $\boldsymbol{X}_L$ that correspond to the K-nearest neighbors of the unlabeled observation $\boldsymbol{x}_t$ and $\boldsymbol{S}(\boldsymbol{x}_t, \boldsymbol{x}_j)$ is the similarity measure between two observations.

Based on their global and local estimations, we compute the weighted average of the response value at each future time step t as follows:

$$\hat{y}_t = \mu\overline{y}_t + (1-\mu)\widetilde{y}_t, t = l+1, \ldots, l+u \tag{4.7}$$

The parameter μ controls the smoothness of the target function; a smaller weight means preference will given to the local estimation.

The newly labeled observations from the future time period will be augmented to the training set in order to rebuild the model. As some of the predicted values $\hat{y}_t$ may deviate quite significantly from either the local or global estimations (depending on μ), incorporating such examples may degrade the performance of semi-supervised HMMR. To overcome this problem, we compute the confidence value for each prediction and use it to weigh the influence of the unlabeled examples during model rebuilding. Let c_t denote the weight assigned to the value of y_t:

$$c_t = \begin{cases} 1, & t = 1, 2, \cdots, l; \\ \exp[-\delta_t], & t = l+1, l+2, \cdots, l+u. \end{cases} \tag{4.8}$$

where $\delta_t = |\widetilde{y}_t - \overline{y}_t|/(\widetilde{y}_t + \overline{y}_t)$. Equation (4.8) assigns a weight of 1 to each historical observation $\boldsymbol{Y}_L$. This is based on the assumption that there is no noise in the historical data. Even if such an assumption is violated, our framework may accommodate noisy observations by applying Equation (4.8) to both training and future observations. The weight formula reduces the influence of future observations whose predictions remain uncertain. After the model has been revised, it is used to re-estimate the response values for the future observations. This procedure is repeated until the changes in the model parameters become insignificant.

We now describe the procedure for estimating the parameters of the semi-supervised HMMR model. Let $\hat{\boldsymbol{Y}}_{\boldsymbol{u}}$ denote the estimated response values for the unlabeled data obtained from Equation (4.7). The likelihood function for the combined data is:

$$\begin{aligned} L &= \sum_{q} P_{\boldsymbol{\Lambda}}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, \boldsymbol{Q} | \boldsymbol{X}_L, \boldsymbol{X}_U) \\ &= \sum_{q} \Bigg(\prod_{t=1}^{l} a_{q_{t-1}q_t} p_{q_t}(y_t | \boldsymbol{x}_t) \prod_{t=l+1}^{l+u} a_{q_{t-1}q_t} p_{q_t}(\hat{y}_t; c_t | \boldsymbol{x}_t) \Bigg) \\ &= \sum_{q} \Bigg(\prod_{t=1}^{l} a_{q_{t-1}q_t} (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[-\frac{(y_t - \mathbf{w}_{q_t}\boldsymbol{x}_t^\top)^2}{2\sigma_{q_t}^2} \right] \\ &\times \prod_{t=l+1}^{l+u} a_{q_{t-1}q_t} (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[-\frac{c_t(\hat{y}_t - \mathbf{w}_{q_t}\boldsymbol{x}_t^\top)^2}{2\sigma_{q_t}^2} \right] \Bigg) \end{aligned}$$

Algorithm 5 Semi-Supervised Hidden Markov Regression for Time Series Prediction

Input: Labeled multivariate time series for the historical period, $\mathcal{D}_L = \{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$ and unlabeled multivariate time series for the future period $\boldsymbol{X}_U$.

Output: Future response $\boldsymbol{Y}_U$

1: Learn the initial HMMR model $\Lambda_0 = (\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{\Sigma}, \boldsymbol{W})$ using the training data $\mathcal{D}_L$.

2: Perform local estimation of $\boldsymbol{Y}_U$ as in Equation (4.7)

3: **repeat**

4: Perform global estimation of $\boldsymbol{Y}_U$ using the current parameters in $\boldsymbol{\Lambda}$ as in Equation (4.6)

5: Calculate the final estimation of $\boldsymbol{Y}_U$ as in Equation (4.7)

6: Calculate the confidence of the predicted values in $\boldsymbol{Y}_U$ as in Equation (4.8)

7: Combine $(\hat{y}_t; c_t)$ estimated in steps 4 and 5 with the training data to re-train the HMMR model $\boldsymbol{\Lambda}' = (\boldsymbol{\Pi}', \boldsymbol{A}', \boldsymbol{\Sigma}', \boldsymbol{W})'$.

8: **until** Convergence ($\|\boldsymbol{\Lambda}' - \boldsymbol{\Lambda}\| \ll \epsilon$)

where

$$p_{q_t}(y_t; c_t|\boldsymbol{x}_t) = (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[-\frac{c_t(y_t - \mathbf{w}_{q_t}x_t^\top)^2}{2\sigma_{q_t}^2}\right]$$

with $c_t = 1$ for historical observations. Unlike the supervised learning case, the weights are used to determine the least square error $(\hat{y}_t - \mathbf{w}_{q_t}\boldsymbol{x}_t^\top)$ for each future observation. To maximize the likelihood function, observations with large weights will incur higher penalty if their response values disagree with the predictions made by the current HMMR model. Such observations are therefore more influential in rebuilding the HMMR model.

To determine the model parameters that maximize the likelihood function, we introduce the

following auxiliary function:

$$
\begin{aligned}
& O(\mathbf{\Lambda}, \mathbf{\Lambda}') \\
= & \sum_{q} P_{\mathbf{\Lambda}}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, Q|\boldsymbol{X}) \ln P_{\mathbf{\Lambda}'}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, Q|\boldsymbol{X}) \\
= & \sum_{q} \sum_{i,j=1}^{N} (\sum_{t=2}^{l+u} \mathbf{1}_{q_{t-1}=i, q_t=j}) P_{\mathbf{\Lambda}}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, Q|\boldsymbol{X}) \ln a'_{ij} \\
+ & \sum_{q} \sum_{i=1}^{N} (\mathbf{1}_{q_1=i}) P_{\mathbf{\Lambda}}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, Q|\boldsymbol{X}) \ln \pi'_i \qquad (4.9) \\
+ & \sum_{q} \sum_{i=1}^{N} \sum_{t=1}^{l+u} (\mathbf{1}_{q_t=i}) P_{\mathbf{\Lambda}}(\boldsymbol{Y}_L, \hat{\boldsymbol{Y}}_u, Q|\boldsymbol{X}) \ln p_{(\sigma'_i, \mathbf{w}'_i)}(y'_t; c_t|\boldsymbol{x}_t)
\end{aligned}
$$

where $\boldsymbol{X} = [\boldsymbol{X}_L; \boldsymbol{X}_U]$ and

$$
y'_t = \begin{cases} y_t, & \text{if } t = 1, \cdots, l; \\ \hat{y}_t, & \text{if } t = l+1, \cdots, l+u. \end{cases}
$$

It can be shown that maximizing the auxiliary function will produce a sequence of model parameters with increasing likelihood values. Taking the derivative of $O(\mathbf{\Lambda}, \mathbf{\Lambda}')$ in Equation (4.9) with respect to each model parameter in $\mathbf{\Lambda}'$, we obtain the following update formula:

$$
\begin{aligned}
\mathbf{w}'_i &= \begin{bmatrix} r_{x_0,x_0} & r_{x_0,x_1} & \cdots & r_{x_0,x_p} \\ \vdots & \vdots & & \vdots \\ r_{x_p,x_0} & r_{x_p,x_1} & \cdots & r_{x_p,x_p} \end{bmatrix} \times \begin{bmatrix} r_{x_0,\hat{y}} \\ \vdots \\ r_{x_p,\hat{y}} \end{bmatrix} \\
\sigma'^2_i &= \frac{1}{\sum_{t=1}^{l+u} \alpha_t(i)\beta_t(i)} \Big\{ \sum_{t=1}^{l} \alpha_t(i)\beta_t(i) c_t (y_t - \mathbf{w}_i \boldsymbol{x}_t^\top)^2 \\
&\quad + \sum_{t=l+1}^{l+u} \alpha_t(i)\beta_t(i)(\hat{y}_t - \mathbf{w}_i \boldsymbol{x}_t^\top)^2 \Big\} \\
a'_{ij} &= \frac{1}{\sum_{t=1}^{l+u-1} \alpha_t(i)\beta_t(i)} \Big\{ \sum_{t=1}^{l-1} \alpha_t(i) a_{ij} p_j(c_{t+1}, y_{t+1}) \beta_{t+1}(j) \\
&\quad + \sum_{t=l+1}^{l+u-1} \alpha_t(i) a_{ij} p_j(c_{t+1}, \hat{y}_{t+1}) \beta_{t+1}(j) \Big\} \\
\pi'_i &= \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{N} \alpha_l(j)}
\end{aligned}
$$

where:

$$
\begin{aligned}
\alpha_t(i) &= \begin{cases} \pi_i p_i(w_1, y_1), \text{if } t = 1 \\ \left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji}\right] p_i(c_t, y_t), \text{if } t > 1 \end{cases} \\
\beta_t(i) &= \begin{cases} 1 \; \forall i, \text{if } t = 1 \\ \sum_{j=1}^{N} a_{ij} p_j(c_{t+1}, y_{t+1}) \beta_{t+1}(j), \text{if } t > 1 \end{cases} \\
r_{x_i, x_j} &= \sum_{t=1}^{l+u} c_t \alpha_t(i) \beta_t(i) \boldsymbol{x}_{ti} \boldsymbol{x}_{tj} \\
r_{x_i, y} &= \sum_{t=1}^{l} c_t \alpha_t(i) \beta_t(i) \boldsymbol{x}_{ti} y_t + \sum_{t=l+1}^{l+u} c_t \alpha_t(i) \beta_t(i) \boldsymbol{x}_{ti} \hat{y}_t \\
p_j(c_t, y_t) &= (2\pi\sigma_{q_t}^2)^{-\frac{1}{2}} \exp\left[- \frac{c_t (y_t - \mathbf{w}_{q_t} \boldsymbol{x}_t^\top)^2}{2\sigma_{q_t}^2} \right]
\end{aligned}
$$

The overall procedure for our proposed semi-supervised algorithm is summarized in Algorithm 4.4.1. After estimating the model parameters $\boldsymbol{\Lambda}$, the response values for future observations are predicted using Equation (4.6).

4.4.2 Data Calibration

The previous section described our proposed algorithm for semi-supervised time series prediction. The underlying assumption behind the algorithm is that the predictor variables for labeled and unlabeled data have the same distribution. This may not be true in real-world applications such as climate modeling or urban growth planning, where the unlabeled data are obtained from a different source (e.g. model simulations) or their distributions may have been perturbed by changes in the modeling domain (e.g. increase of population growth or greenhouse gas concentration).

Several recent studies on semi-supervised classification have suggested the negative effect of unlabeled data, especially when a classifier assumes an incorrect structure of the data [57] or when the labeled and unlabeled data have different distributions [199]. None of these studies, however, have been devoted to regression or time series problems. Our experimental results demonstrate that, while in most cases, semi-supervised HMMR indeed outperforms its supervised counterpart, for the climate modeling domain, where the historical and future observations come from different sources, semi-supervised learning do not significantly improve the performance of HMMR. To overcome this problem, we propose a data calibration technique to deal with the inconsistencies between historical and future data.

A straightforward way to calibrate $\boldsymbol{X}_L$ and $\boldsymbol{X}_U$ is to standardize the time series of each predictor variable by subtracting their means and dividing by their respective standard deviations. The drawback of this approach is that the covariance structures for $\boldsymbol{X}_L$ and $\boldsymbol{X}_U$ are not preserved by the standardization procedure. As a result, a model trained on the historical data may still not accurately predict the future data since the relationship between the predictor variables may have changed. In this paper, we propose a new data calibration approach to align the covariance structure of the historical data and future data. Our approach seeks to find a linear transformation matrix β that is applicable to the future unlabeled data $\bar{\boldsymbol{X}}_U$ such that the difference between their covariance matrices is minimized.

Let A denote the covariance matrix of $\boldsymbol{X}_U$ and B denote the covariance matrix of $\boldsymbol{X}_L$:

$$\begin{aligned} A &= E\Big[(\boldsymbol{X}_L - E(\boldsymbol{X}_L))^\top (\boldsymbol{X}_L - E(\boldsymbol{X}_L)\Big] \\ B &= E\Big[(\boldsymbol{X}_U - E(\boldsymbol{X}_U))^\top (\boldsymbol{X}_U - E(\boldsymbol{X}_U))\Big] \end{aligned}$$

The covariance matrix after transforming $\boldsymbol{X}_U$ to $\boldsymbol{X}_U\beta$ is

$$\begin{aligned} B' &= E\Big[(\boldsymbol{X}_U\beta - E(\boldsymbol{X}_U\beta))^\top (\boldsymbol{X}_U\beta - E(\boldsymbol{X}_U\beta))\Big] \\ &= \beta^\top B\beta \end{aligned}$$

The transformation matrix β can be estimated using a least-square approach:

$$\begin{aligned} \arg\min_{\beta} \boldsymbol{J} &= \arg\min_{\beta} \|A - B'\|_F^2 \\ &= \arg\min_{\beta} \|A - \beta^\top B\beta\|_F^2 \end{aligned} \tag{4.10}$$

The optimization problem can be solved using a gradient descent algorithm:

$$\beta^{i+1} = \beta^i - \eta \frac{\partial \boldsymbol{J}}{\partial \beta}$$

where:

$$\frac{\partial \boldsymbol{J}}{\partial \beta} = 4(B\beta\beta^\top B^\top \beta - B\beta A^\top)$$

and $\eta > 0$ is the learning rate.

Although the preceding data calibration approach helps to align the covariance matrices of the data, our experimental results show that the transformation tends to significantly distort the neighborhood structure of the observations. Since our semi-supervised HMMR framework performs

local estimation based on the nearest neighbor approach, such a transformation leads to unreliable local predictions and degrades the overall performance of the algorithm. An ideal transformation should preserve the neighborhood information while aligning the covariance structure. To accomplish this, we create a combined matrix $\boldsymbol{X} = [\boldsymbol{X}_L; \boldsymbol{X}_U]$, and compute the covariance matrix B using the matrix, i.e.,

$$B = E\Big[(\boldsymbol{X} - E(\boldsymbol{X}))^\top (\boldsymbol{X} - E(\boldsymbol{X}))\Big]$$

After calibration using the gradient descent method described previously, both $\boldsymbol{X}_L$ and $\boldsymbol{X}_U$ will be transformed as follows:

$$\boldsymbol{X}'_L = \boldsymbol{X}_L\boldsymbol{\beta} \quad \boldsymbol{X}'_U = \boldsymbol{X}_U\boldsymbol{\beta}.$$

The transformed data $\boldsymbol{X}'_L$ and $\boldsymbol{X}'_U$ will serve as the new training and test data for the semi-supervised HMMR algorithm.

4.5 Experimental Result

We have conducted several experiments to evaluate the performance of proposed algorithms for long term time series prediction. All the experiments were performed on a Windows XP machine with 3.0GHz CPU and 1.0GB RAM.

4.5.1 Experimental Setup

First, we compared the relative performance among multi-stage prediction, independent value prediction and parameter prediction on 21 real data sets. The real datasets are obtained from the UCI Machine Learning Repository [154] and the Time Series Data Library [110]. All the time series used are univariate time series with length recored in the last column of Table 4.3. The parameters for the prediction approaches include the order of regression model p and the degree of polynomial fit d (for parameter prediction). We use Akaike's final prediction error (FPE) [5] as our criterion for determining the right order for p in the MLR model. The same criterion is applicable to estimate the degree of the polynomial function used in parameter prediction. To determine the correct order for RNN, we employ the method described by Kennel in [119] by choosing a value for p such that the number of false nearest neighbors is close to zero.

Table 4.2. Description of the UCR time series data sets for Semi-HMMR

Data Sets	Length	# Variables
Dryer	867	6
Foetal	2500	9
Glass	1247	9
Greatlake	984	5
Steam	9600	4
Twopat	5000	129
Logistic	1000	101
Leaf	442	151
Cl2full	4310	166

Second, we conducted several experiments to evaluate the performance of our proposed semi-supervised algorithm. Table 4.2 summarizes characteristics of the time series used in this experiment. The time series are obtained from the UC Riverside time series data repository [120]. We divide each time series into 10 disjoint segments. To simulate this as a long term forecasting problem, for the k-th run, we use segment k as training data and segments $k+1$ to $k+5$ as test data. The future period is therefore five times longer than the training period for each run. The results reported for each data set are based on the average R^2 for 5 different runs. We also consider three other competing algorithms for our experiments: (1) univariate autoregressive (AR) model, (2) multiple linear regression (MLR), and (3) supervised HMMR.

There are also several parameters that must be determined for our semi-supervised HMMR, such as the number of nearest neighbors K, the number of hidden states N, and smoothness parameter μ. To determine the number of nearest neighbors, we perform 10-fold cross validation on the training data using the K-nearest neighbor regression method [99]. There are various methods available to determine the number of hidden states N. These methods can be divided into two classes—modified cross-validation and penalized likelihood methods (such as AIC, BIC, and ICL). In this work, we employ the modified 10-fold cross validation with missing value approach [36] to select the best N. For each fold, we randomly select one-tenth of the observations to be removed from the training data. The likelihood function will be estimated from the remaining nine-tenth of the training data (while treating the removed data as missing values). The number of states N that produces the highest R^2 will be chosen as our parameter. To ensure smoothness in the target function, μ should be biased more towards the local prediction. Our experience shows

Table 4.3. R^2 of three methods on 21 data sets using Multiple Linear Regression

	Multi-stage	Independent	Parameter	length
Milk	0.9267	0.9295	0.9172	168
Temp.	0.7224	0.7041	0.7064	216
PET	0.9581	0.9586	0.9381	216
PREC	0.9690	0.9683	0.9466	216
Solar	0.8782	0.8764	0.7868	216
Appb	0.7026	0.6196	0.6197	78
Appd	0.7848	0.7605	0.7234	64
Appf	0.6853	0.7555	0.7009	108
Appg	0.1358	0.0657	0.0782	114
Deaths	0.2691	0.4440	0.4367	72
Lead	0.5805	0.5793	0.5794	150
Sales	0.6813	0.6363	0.6363	150
Wine	0.7262	0.7098	0.6791	142
Seriesc	0.0641	0.0155	0.0155	224
Odono	0.5774	0.5269	0.5288	114
Qbirth	0.4550	0.5207	0.4769	300
Bond2	0.4774	0.4114	0.4116	309
Daily	0.7994	0.7863	0.7863	500
Food	0.8005	0.8071	0.8050	117
Treerin	0.1071	0.1193	0.1182	948
Pork	0.0538	0.2052	0.2082	99

that this can be accomplished by setting μ somewhere between 0.1 to 0.3. We fix the smoothness parameter $\mu = 0.1$ throughout our experiments.

4.5.2 Comparative Study on Multi-step Prediction

A general comparative study on the three prediction approaches using real datasets is performed in this section. A Win-Draw-Loss Table is created to compare the relative performance between two prediction approaches when applied to a number of data sets. We use the criterion of 0.01 difference in R^2 to determine whether one approach wins or loses against another approach. For a stricter evaluation, we also apply the paired t significance test to determine whether the observed difference in R^2 is statistically significant. To do this, we first calculate the difference (ΔR) in the R^2 obtained from two prediction approaches on each data set. The means $\overline{\Delta R}$ and standard deviation $s_{\Delta R}$ of the observed differences are also calculated. To determine whether the difference

Table 4.4. R^2 of three methods on 21 data sets using Recurrent Neural Network

	Multi-stage	Independent	Parameter
Milk	0.9267	0.9295	0.9082
Temp.	0.7224	0.7041	0.7064
PET	0.9581	0.9586	0.9381
PREC	0.9690	0.9683	0.9466
Solar	0.8782	0.8764	0.7868
Appb	0.7026	0.6196	0.6197
Appd	0.7848	0.7605	0.7234
Appf	0.6853	0.7555	0.7009
Appg	0.1358	0.0657	0.0782
Deaths	0.2691	0.4440	0.4367
Lead	0.5805	0.5793	0.5794
Sales	0.6813	0.6363	0.6363
Wine	0.7262	0.7098	0.6791
Seriesc	0.0641	0.0155	0.0155
Odono	0.5774	0.5269	0.5288
Qbirth	0.4550	0.5207	0.4769
Bond2	0.4774	0.4114	0.4116
Daily	0.7994	0.7863	0.7863
Food	0.8005	0.8071	0.8050
Treerin	0.1071	0.1193	0.1182
Pork	0.0538	0.2052	0.2082

is significant, we compute their T-statistic:

$$\mathbf{t} = \frac{\overline{\Delta R}}{s_{\Delta R}/\sqrt{u}}$$

which follows a t-distribution with $u - 1$ degrees of freedom. Under the null hypothesis that the two prediction approaches are comparable in performance, we expect the value of $\mathbf{t}$ should be close to zero. From the computed value for $\mathbf{t}$, we estimate the p-value of the difference, which corresponds to the probability of rejecting the null hypothesis. We say the difference in performance is statistically significant if $\mathbf{p} < 0.05$ and highly statistically significant if $\mathbf{p} < 0.001$.

We apply the three prediction approaches to 21 real data sets to compare their relative performance. The R^2 value for each data set is obtained by 10-fold cross validation. The size of the prediction window is set to $u = 24$. Table 4.3 summarizes the R^2 for the three prediction approaches using MLR as the underlying regression method. Their relative performance is summarized in Table 4.5 in terms of the number of wins, draws and losses. We also test the significance of the

Table 4.5. Win-Draw-Loss results for MLP

	Multi vs Indep	Multi vs Param	Indep vs Param
0.01 diff	10-6-5	14-2-5	8-12-1
t value	0.1496	0.8761	2.7299
p value	0.8826	0.3914	0.0129

Table 4.6. Win-Draw-Loss results for RNN

	Multi vs Indep	Multi vs Param	Indep vs Param
0.01 diff	7-0-14	5-2-14	13-1-7
t value	2.6396	3.3884	0.3012
p value	0.0157	0.0029	0.7664

difference using paired t-significance test. The results show that the observed difference between the R^2 of multi-stage and independent value prediction is not that significant. However, the performance of parameter prediction is significantly worse than independent value prediction. This is because MLR may not be suitable to fit the parameters of the function, which have nonlinear relationships with the time series values.

Tables 4.4 and 4.6 show the results using RNN as the underlying regression method. Observe that the independent value and parameter prediction approaches perform significantly better than multi-stage prediction ($\mathbf{p} < 0.05$). For multi-stage prediction, the R^2 for RNN is lower than the R^2 of MLR in 10 out of 21 data sets, which suggests the possibility of model overfitting when using a flexible regression method such as RNN. Nevertheless, we still find 17 data sets in which independent value prediction with RNN outperforms all the prediction approaches using MLR and 12 data sets in which parameter prediction with RNN outperforms all the prediction approaches using MLR. The results suggest that, using nonlinear regression methods such as RNN, the independent value and parameter prediction approaches may achieve better performance than multi-stage prediction. Moreover, for parameter prediction, most of the data sets require $d < 5$, which makes it more efficient to build compared to independent value prediction (which requires building $u = 24$ models).

4.5.3 Performance Comparison on Semi-Supervised HMMR

Table 4.7 compares the R^2 of our proposed framework (semiHMMR) against the univariate AR (UAR), multiple linear regression(MLR), and supervised Hidden Markov Model Regression

Table 4.7. Average R^2 for UAR, MLR, HMMR and SemiHMMR on UCR time series data sets.

Data Sets	UAR	MLR	HMMR	SemiHMMR
Dryer	0.4347	0.6253	0.6797	0.7105
Foetal	0.1221	0.7762	0.7782	0.8014
Glass	0.0019	0.2319	0.2543	0.5170
Greatlake	0.1367	0.5840	0.6047	0.6846
Steam	0.2850	0.5027	0.5237	0.6808
Twopat	0.0877	0.1739	0.1465	0.1774
Logistic	0.0195	0.0212	0.0232	0.0246
Leaf	0.2063	0.7382	0.7421	0.7897
Cl2full	0.1621	0.6840	0.7031	0.8263

(HMMR). First, observe that the performance of univariate AR is significantly worse than multivariate MLR and supervised HMMR model. This is consistent with the prevailing consensus that multivariate prediction approaches are often more effective than univariate approaches because they may utilize information in the predictor variables to improve its prediction. Second, we observe that HMMR generally performs better than MLR on most of the data sets. This result suggests the importance of learning models that take into consideration the dependencies between observations. Finally, the results also show that semi-supervised HMMR is significantly better than HMMR on the majority of the data sets. The improvements achieved using semi-supervised HMMR exceed 10% on data sets such as "Glass", "Greatlake", "Steam", and "Cl2full".

4.5.4 Value of Unlabeled Data for Semi-Supervised HMMR

We further show the value of incorporating unlabeled data into the model retraining phase of the semi-supervised HMMR algorithm. The average R^2 for each iteration before convergence is plotted in Figure 4.9 when applying the semiHMMR algorithm to the "Foetal" data set. The R^2 of semiHMMR algorithm increases from 0.7782, which is the same to the R^2 achieved by HMMR, to 0.8014 after several iterations. The performance of local KNNR estimation algorithm is 0.6951, which is considerably worse than HMMR and semiHMMR algorithm. The R^2 obtained by a linear combination of KNNR and HMMR with $\mu = 0.1$ is 0.7216, which is also much lower than the R^2 of the semiHMMR algorithm. This experiment demonstrates the effectiveness of retraining the model using unlabeled data in the semiHMMR algorithm.

Another experiment is conducted to show the value of unlabeled data when there is limited

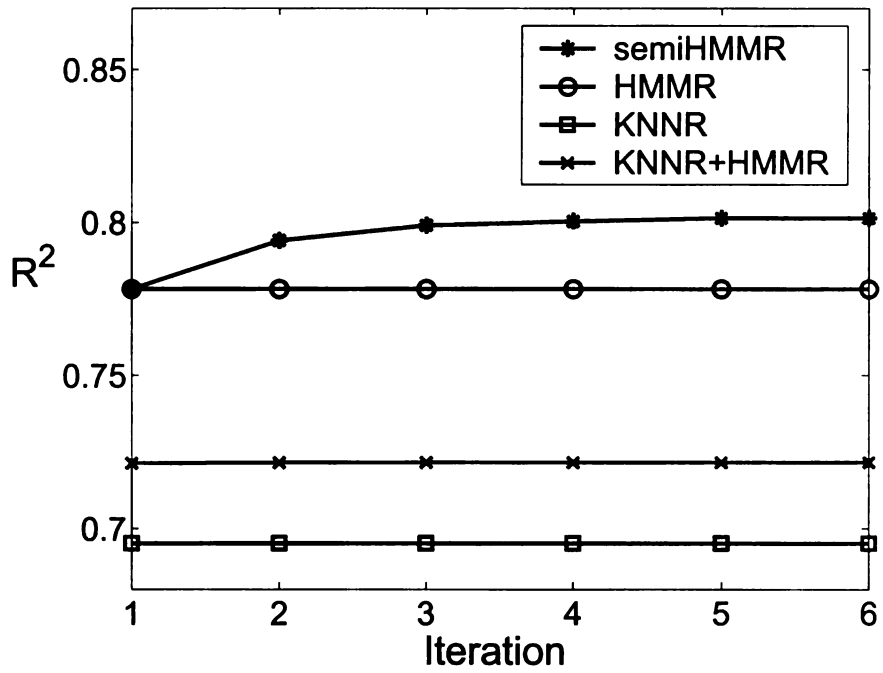


Figure 4.9. The performance of SemiHMMR by each iteration compared with KNNR, HMMR and KNNR+HMMR on the data set "Foetal".

training data. We have used the "Steam" time series for this experiment and vary the ratio of unlabeled to labeled data from 1 to 5 by changing the size of labeled data and report its average R^2. As shown in Figure 4.10, when the ratio is 1, the R^2 for MLR and supervised HMMR are nearly the same as that for semi-supervised HMMR. However, when the ratio increases to 5, the performance of both MLR and supervised HMMR degrades rapidly (R^2 from 0.67726 to 0.5027 for MLR and from 0.6884 to 0.5229 for HMMR) whereas the R^2 for semi-supervised HMMR decreases only slightly, from 0.6924 to 0.6807. The results of this experiment show that semi-supervised HMMR can effectively utilize information in the unlabeled data to improve its prediction, especially when labeled data is scarce.

4.5.5 Semi-Supervised HMMR with Covariance Alignment on Climate Prediction

Here we apply the proposed semi-supervised HMMR algorithm to the climate projection problem by combining historical observations with GCM data. The GCM models are developed based on the basic principles of physics, chemistry, and fluid mechanics, taking into account the coupling

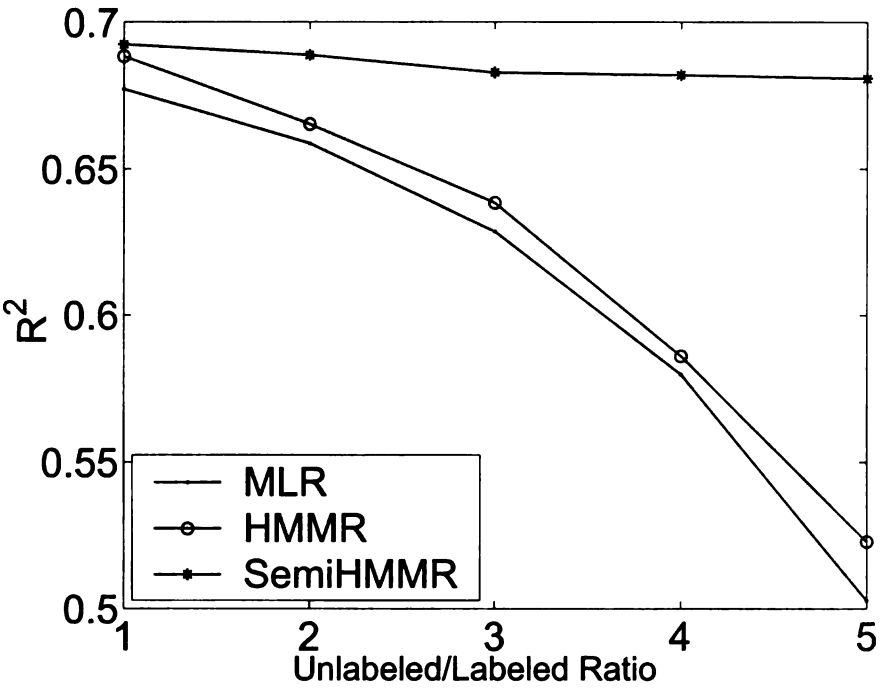


Figure 4.10. The performance of MLR, HMMR and SemiHMMR when varying the ratio of unlabeled to labeled data

between land, ocean, and atmospheric processes. Although the outputs from these models sufficiently capture many of the large-scale ($\approx$ 150-300 km spatial resolution) circulation patterns of the observed climate system, they may not accurately model the response variable at the local or regional scales ($\approx$ 1-50 km) needed for climate impact assessment studies [214]. As a result, the coarse-scale GCM outputs need to be mapped into finer scale predictions, a process that is known as "downscaling" in the Earth Science literature. Regression is a popular technique for downscaling, where the GCM outputs are used as predictor variables and the response variable corresponds to the local climate variable of interest (e.g. precipitation or temperature).

For this experiment, we downloaded climate data from the Canadian Climate Change Scenarios Network Web site [35]. The data consists of daily observations for 26 climate predictor variables including sea-level pressure, wind direction, vorticity, humidity, etc. The response variable corresponds to the observed mean temperature at a meteorological station. In short, there are three sources of data for this experiment: (1) Mean temperature observations from 1961 to 2001 to be used as response variable, (2) Reanalysis data from NCEP (National Center for Environmental Prediction) reanalysis project from 1961 to 2001 to be used as predictor variables for training

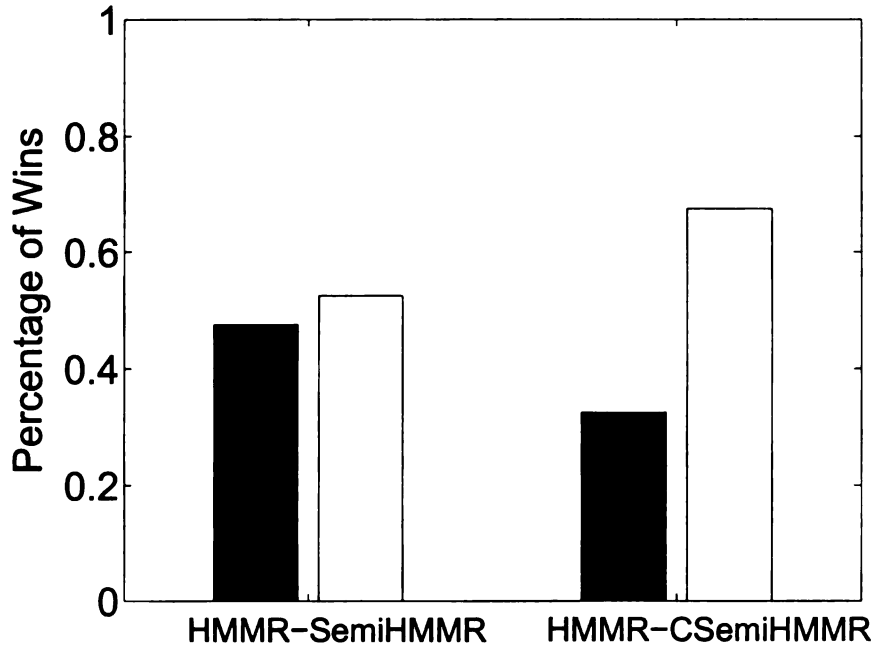


Figure 4.11. Performance comparison between HMMR, SemiHMMR, and CSemiHMMR on Canadian climate data

data, and (3) Simulation data from HADCM3 global climate model from 1961 to 2099 to be used as predictor variables for future data. Since the mean temperature for the future time period is unavailable, we conducted our experiment using NCEP reanalysis and the observed mean temperature data from 1961 to 1965 for training and HADCM3 simulation data from 1966 to 1991 for testing.

The main purpose of this experiment is to investigate the effect of data calibration on the performance of semi-supervised HMMR when applied to climate prediction. We show that data calibration is useful when the historical observations and future data come from different sources. To compare the relative performance of supervised HMMR, semi-supervised HMMR (SemiHMMR) and semi-supervised HMMR with data calibration (CSemiHMMR), we applied the algorithms to predict the mean daily temperature for 40 randomly selected meteorological stations (another criterion for choosing a station is that the time series must be complete, i.e., it has no missing values). in Canada. The bar chart shown in Figure 4.11 indicates the fraction of locations in which one algorithm has a higher R^2 than the other. Unlike the results reported in Section 4.5.3, the bar chart seems to suggest that the performance of semi-supervised HMMR is only comparable

Table 4.8. Comparing the average R^2 values for MLR, HMMR, SemiHMMR, and CSemiHMMR on the Canadian climate data.

(Lat°,Lon°)	MLR	HMMR	SemiHMMR	CSemiHMMR
(48.65N,123.43W)	0.6038	0.6144	0.6392	0.6356
(48.95N,54.58W)	0.3276	0.3447	0.4187	0.4350
(52.18N,113.89W)	0.3555	0.4591	0.4538	0.4732
(48.33N,71W)	0.4843	0.4842	0.4776	0.4982
(82.52N,62.28W)	0.4982	0.5456	0.5875	0.6277

to supervised HMMR (55% versus 45%). This is actually consistent with the conclusions drawn in [57, 199] for semi-supervised classification in which it was suggested that unlabeled data with different distribution may not improve the performance of a semi-supervised algorithm. However, with the calibration method developed in Section 4.4.2, CSemiHMMR actually performs better than HMMR on 72% of the data sets. This result confirmed the effectiveness of incorporating the covariance alignment technique to our semi-supervised learning framework. We also illustrate the R^2 values for five of the selected stations in Table 4.8. The latitude and longitude for each station is recorded in the first column of Table 4.8. Though the performance of semiHMMR appears to be worse than supervised HMMR at (52.2°N, 113.9°W) and (48.3°N 71°W), the R^2 values for semi-supervised HMMR with data calibration is clearly superior for both locations.

In Section 4.4.2, we argued that aligning $\boldsymbol{X}_L$ with $\boldsymbol{X}_U$ (we call this calibration technique 1) may not be as effective as calibrating $\boldsymbol{X}_L$ with the combined matrix $\boldsymbol{X}_C = [\boldsymbol{X}_L, \boldsymbol{X}_U]$ (we call this calibration technique 2). This is because the former may result in significant loss of nearest neighbor information, thus degrading the performance of semi-supervised HMMR (note that the results shown in Figure 4.11 are based on calibration technique 2). To measure the degree of alignment and loss of neighborhood information using the calibration techniques, we define the following two measures: RCovDiff and NNLoss. Let $\Delta_0(\boldsymbol{X}_A, \boldsymbol{X}_B)$ denote the difference between the covariance matrices constructed from $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ before alignment, and $\Delta_1(\boldsymbol{X}_A, \boldsymbol{X}_B)$ denote the corresponding difference after alignment. RCovDiff measures the reduction of the covariance difference before and after alignment, i.e.:

$$\text{RCovDiff1} = \frac{|\Delta_0(\boldsymbol{X}_L, \boldsymbol{X}_U) - \Delta_1(\boldsymbol{X}_L, \boldsymbol{X}_U)|}{\Delta_0(\boldsymbol{X}_L, \boldsymbol{X}_U)}$$

$$\text{RCovDiff2} = \frac{|\Delta_0(\boldsymbol{X}_L, \boldsymbol{X}_C) - \Delta_1(\boldsymbol{X}_L, \boldsymbol{X}_C)|}{\Delta_0(\boldsymbol{X}_L, \boldsymbol{X}_C)}$$

Table 4.9. Comparing the degree of alignment and loss of neighborhood structure information using calibration techniques 1 and 2.

(Lat°,Lon°)	RCovDiff1	RCovDiff2	NNLoss1	NNLoss2
(48.6N,123.4W)	0.998	0.785	0.994	0.186
(48.9N,54.6W)	0.993	0.714	0.998	0.116
(52.2N,113.9W)	0.989	0.728	0.995	0.108
(48.3N,71W)	0.992	0.703	0.996	0.086
(82.5N,62.3W)	0.993	0.702	0.991	0.096

A calibration technique with larger RCovDiff will produce covariance matrices that are better aligned with each other. We also measure the loss of neighborhood structure due to data calibration in the following way. Let $M_0(\boldsymbol{X}_A, \boldsymbol{X}_B)$ denote a 0/1 matrix computed based on the 1-nearest neighbor of each example in $\boldsymbol{X}_B$ to the examples in $\boldsymbol{X}_A$ before alignment and $M_1(\boldsymbol{X}_A, \boldsymbol{X}_B)$ denote the corresponding matrix after alignment. The NNLoss measure is defined as follows:

$$\text{NNLoss} = \frac{|M_0(\boldsymbol{X}_L, \boldsymbol{X}_U) - M_1(\boldsymbol{X}_L, \boldsymbol{X}_U)|}{M_0(\boldsymbol{X}_L, \boldsymbol{X}_U)}$$

Unlike RCovDiff, the same equation is applied to both calibration techniques 1 and 2. Table 4.9 compares the results of both calibration techniques. Although calibration technique 1 produces more well-aligned covariance matrices, it loses more information about its neighborhood structure. This explains our rationale for using calibration technique 2 for semi-supervised HMMR.

4.6 Summary

Time series forecasting is an important task and has been studied for a long time in statistics, data mining and machine learning literature. However, most previous work on time series forecasting focused on single step or short term prediction problems, which are not sufficient for many applications. Although there has been some work extending single step time series forecasting methods to long term prediction, it suffers from problems such as error accumulation. In this chapter, we first investigated the issues in the long term time series forecasting and then proposed a semi-supervised multivariate time series prediction algorithm to improve the prediction accuracy. The main contributions of this work are:

- We first investigated the issues associated with long term time series forecasting problem by studying the advantages and disadvantages of three prediction approaches both theoretically

and empirically. Using the bias-variance decomposition framework, our analysis shows that multi-stage prediction suffers from the error accumulation problem, especially when the prediction window is long. Independent value prediction alleviates this problem by making predictions independently at each time step. However, it has its own problems. First it is more expensive since it has to learn a separate prediction model for each step in the prediction window. Second, it has difficulty in learning the appropriate function when the prediction window is long because the true function becomes more complex with increasing prediction time steps. Third, this approach does not smooth out the effect of noise, unlike multi-stage prediction. Parameter prediction smooths the effect of noise by fitting a function over the entire output sequence and avoids the error accumulation problem by making independent predictions. It is more efficient than independent value prediction when the parameter set is small. However, finding the appropriate parameter function to fit the output values can still be quite a challenging task.

- We further investigated a multivariate time series prediction method to improve the prediction accuracy for long term forecasting by utilizing information from other related time series. The future data of the predictor variables can be collected by simulation in many domains such as climate modeling; however, it is used in supervised fashion in traditional methods. In this work, a semi-supervised multivariate time series prediction framework based on the Hidden Markov Regression model was proposed to obtain a better model by utilizing both historical and future data in training. The experimental results on benchmark time series data sets demonstrate the superior performance of the semi-supervised HMMR algorithm compared with other supervised methods.

- However, applying the semi-supervised HMMR to climate projection problem by combining the historical reanalysis data and GCM future data did not show much improvement, which demonstrated that the inconsistencies between training and test data may not improve the model's performance. To compensate for this problem, we proposed a covariance alignment data calibration method that will transform the training and test data into a new space before applying the semi-supervised HMMR algorithm. The semi-supervised HMMR with data calibration technique clearly achieves considerable improvement according to our

evaluation using the climate data sets.

CHAPTER 5

Anomaly Detection, Characterization, and Visualization

In this chapter, we discuss methods that are used for the detection, characterization, and visualization of anomalies in time series data. Give time series data $\mathcal{D}$ with length l, the goal of anomaly detection is to discover the timestamps at which the measurement values for one or more variables have deviated significantly from their nominal behavior. The nominal behavior represents the expected value of a time series based on its its own history as well as its relationship to other time in $\mathcal{D}$. The major innovations and contributions of this chapter are listed as follows.

- We present a graph-based algorithm and some extensions to detect different types of anomalies in time series data. The graph-based method is applied to Earth Science data to detect ecosystem disturbance.
- A kernel alignment technique is further proposed for the detection and characterization of anomalies in noisy multivariate time series data, which is used to discriminate the ecosystem disturbances in Earth Science applications.
- A clustering-based framework is proposed to aid the visual exploration and detection of anomalies from large scale time series data in high spatial resolution, e.g. to explore the ecosystem disturbances in global FPAR time series data.

The rest of this chapter is organized as follows. Section 5.1 introduces some preliminaries on time series anomaly detection. Section 5.2 presents the graph-based based time series anomaly

detection algorithm. Section 5.3 proposes the kernel alignment based method for multivariate time series anomaly detection and characterization. Two extensions of the graph-based anomaly detection algorithms are discussed in 5.4. Section 5.5 introduces a clustering-based visualization system for the exploratory analysis of anomalies in Earth Science data. The evaluation of our algorithms and system is given in Section 5.5.3.

5.1 Preliminaries

Here we present some background for the problem of time series anomaly detection.

5.1.1 Anomaly Detection in Time Series Data

Anomaly detection in time series data can be categorized into point-wise anomaly detection and subsequence anomaly detection . A formal definition of point-wise anomaly and subsequence anomaly in time series data is given as follows:

Definition 9. (Point-wise Anomaly) *Point-wise anomaly refers to the timestamps at which the observed values are significantly different than the rest of the time series.*

Examples of point-wise anomalies includes peak or valley anomaly whose values are significantly larger or smaller than the rest of time series.

Definition 10. (Subsequence Anomaly) *Subsequence anomaly is defined as a segment of length* $d \ll l$ *that does not match any other other segments in the time series.*

An example of such type of anomaly is shown in Figure 5.9(a), where there is an unusual shape of subsequence in the time series.

In both cases, the anomalies can be characterized as *global* or *local* anomalies, depending on how the nominal behavior is determined. The nominal behavior for global anomalies is computed from the entire time series whereas for local anomalies, the nominal behavior is determined with respect to some local neighborhood, which can be defined in several ways using temporal information. One way is to consider the time-based neighborhood:

Definition 11. (Time-based Local Anomaly) *A data point at time t is considered a time-based local anomaly if its value is significantly different than the values within the time interval $[t - k, t + k]$. An example of such anomaly is shown in Figure 5.10(a).*

Another category of local anomaly for time series with periodical patterns is cycle-based defined as:

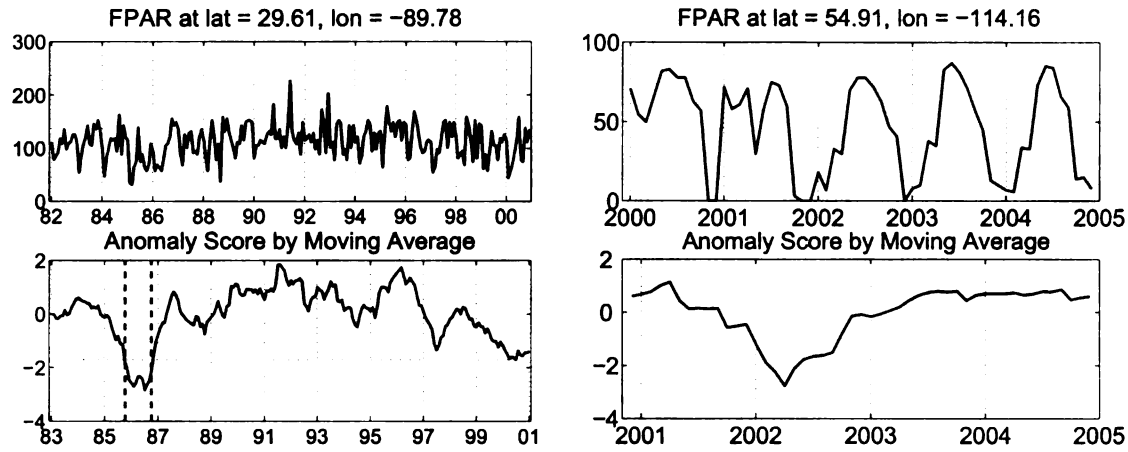
Definition 12. (Cycle-based Local Anomaly) *Let k be the periodicity of a time series. A data point at time t is considered a cycle-based local anomaly if its value is significantly different than the values measured at times $t - k, t - 2k, \cdots$ and $t + k, t + 2k, \cdots$.*

The above discussion applied to both univariate and multivariate time series.

5.1.2 Moving Average Time Series Anomaly Detection

Moving average is the simplest anomaly detection method for time series data. In this approach, a time series was first detrended using a linear adjustment. For example, this is necessary to minimize the possibility that, in cases where there is gradual but marked increase in monthly FPAR over the time series data, any potential disturbance events occurring relatively near the end of the series are not overlooked. Then the detrended time series was subsequently deseasonalized by computing the moving average time series to remove the dominant seasonal oscillations.

When applied to detect ecosystem disturbance in FPAR time series, the domain experts hypothesize that a "sustained" disturbance event could be defined as any decline in average annual FPAR levels (at an assigned significance level) that lasts for a temporal threshold value of at least 12 consecutive monthly observations at any specific location [173]. The logic used here is that an actual disturbance involves a sustained decline in FPAR because the structure of the vegetation cover has been severely altered or destroyed during the disturbance event, to a magnitude that lowers FPAR significantly for at least one seasonal growing cycle, after which time regrowth and recovery of the former vegetation structure may permit FPAR to increase again. Significant declines in average annual FPAR levels can be defined to be greater than 1.7 standard deviations (SD) below the 19-year average FPAR computed for any specific location. The threshold of 1.7 SD was chosen based on the 95% confidence level in rejecting the null hypothesis using a one-sided statistical t-test.



(a) Detection of ecosystem disturbance by moving average due to Hurricane Elena (during September 1985) from FPAR time series data.

(b) Detection failure of ecosystem disturbance by moving average due to Chisholm wildfire.

Figure 5.1. FPAR time series at two different locations and their anomaly scores learned by moving average.

Figure 5.1(a) shows an example of an ecosystem disturbance event recorded at 29.61°N and 89.78°W. The event coincides with the timing of Hurricane Elena (September 1985) while its location is near the vicinity of the landfall points of the storm. In addition to this example, this method successfully detects documented landfall points of other tropical storms including Hurricane Alicia, Gloria, Gilbert, and Hugo. For an extended discussion that includes coverage of droughts, heat waves, cold waves, and blizzards, see [172].

While this approach can successfully detect many large-scale disturbance events in Earth Science data, it also has several limitations. First, the timing of the event may vary since it uses 12-month moving average to deseasonalize the time series. Second, it may not be able to detect small scale anomaly events in which the vegetation structure of the region recovers in less than one growing seasonal cycle. Third, it has problems detecting certain special kind of anomalies such as local anomalies. Finally, it fails to explore the relationship between different variables, e.g. the correlation between climate and FPAR time series. Figure 5.1(b) shows the failure of this approach to detect the disturbance event in Chisholm, Alberta where a major wildfire had occurred in May 2001.

5.1.3 Anomaly Detection and Characterization in Multivariate Time Series Data

A multivariate time series $\mathcal{D} = \{\boldsymbol{X}_i\}_{i=1}^{p}$ is a collection of time series that corresponds to the measurements of p real-valued variables spanning the same time interval. For some application domains, one of the variables $Y \in \mathcal{D}$ may be designated as the target variable of interest, while the remaining $\mathcal{D} - \{Y\}$ variables are known as the predictor variables. In this study, we consider two types of multivariate time series anomaly detection problems: *general* and *target specific*.

For *general multivariate time series* anomaly detection, all variables in $\mathcal{D}$ are considered equally important when detecting anomalies. For example, in network intrusion detection, anomalies can be detected based on the deviations observed in one or more time series—e.g. the unusually high number of connections originating from the same IP address (for a denial of service attack), the wide range of port numbers used (for a port scan attack), or the abnormally large number of ICMP packets sent (for a ping flood attack).

In contrast, *target specific anomaly detection* attempts to find anomalies in the time series for a target variable that are correlated with the deviations observed in the predictor time series. An example application can be found in the Earth Science domain, where scientists are interested in identifying ecosystem disturbances such as wildfires from satellite observations of the global vegetation cover data (i.e., the target variable). By correlating the anomalies found in the vegetation cover data with those found in climate variables such as temperature and precipitation, this may help scientists to distinguish between climate-induced anomalies from human-induced anomalies. As a result, target specific anomaly detection can help characterize the anomalies in a target variable using the anomalies observed in predictor time series.

5.1.4 Anomaly Exploration in Large Scale Time Series Repository

Anomaly detection in a very large time series database is challenging because of the computation cost for process all the data. For example, huge amounts of time series such as FPAR remote sensing data are available in high spatial resolution in Earth Science application. Scanning all the time series across all the locations for anomalies becomes really expensive, not to mention that most anomaly detection algorithms require the adjustment of one or more thresholds. Visual data

mining is defined as the use of interactive visual representation of the abstract data to amplify the pattern discovery process. The goal of the visualization is to reduce the complexity of a given data set by presenting only small portion of the massive time series at a time, while at the same time minimizing the loss of information. Interaction is a fundamental component of visualization that permits the user to modify the visualization parameters, browsing the data or querying for results. Anomalies are discovered interactively between the user and the visualized anomaly exploration system.

5.2 Graph-based Time Series Anomaly Detection

This section presents the graph-based anomaly detection algorithm for time series anomaly detection. This method first constructs a kernel matrix from the given time series data and then run a random walk to obtain the anomaly scores for each data point or subsequence.

5.2.1 Kernel Matrix for Time Series Data

Here we introduce the kernel matrix to measure the similarity between data points in a time series. Let i and j denote a pair of timestamps in the time series. Each timestamp is associated with a set of measurements for p variables (Note that $p = 1$ for univariate time series and $p > 1$ for multivariate time series). Although there are numerous similarity measures that have been proposed in the literature (e.g. cosine and correlation), few of them are applicable to both univariate and multivariate time series. In this study, we measure the similarity between two timestamps using the RBF function:

$$\boldsymbol{K}(i,j) = \exp\left[-\frac{\sum_{k=1}^{p}(x_{ki} - x_{kj})^2}{\boldsymbol{\sigma}^2}\right] \tag{5.1}$$

where x_{ik} is a data point at time i for time series variable k. The RBF function can also be generalized to measure the similarity between two subsequences of width d as follows:

$$\boldsymbol{K}(i,j) = \exp\left[-\frac{\sum_{k=1}^{p}\sum_{s=0}^{d-1}(x_{k,i+s} - x_{k,j+s})^2}{\boldsymbol{\sigma}^2}\right], \tag{5.2}$$

where i and j are the starting timestamps for each subsequence. Based on the similarity measure, we may construct a non-negative, symmetric matrix $\boldsymbol{K}$ to capture the pairwise similarity between

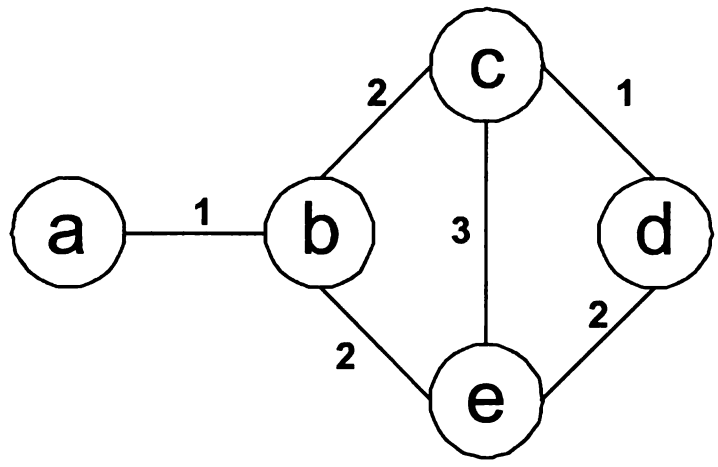


Figure 5.2. A weighted graph representation of a kernel matrix.

every pair of timestamps (or subsequences) in a given time series. $\boldsymbol{K}$ is also known as a kernel matrix (or a Gram matrix). The next section describes a graph-based algorithm to detect anomalies by utilizing the information in a kernel matrix.

5.2.2 Random Walk on Graph for Anomaly Detection

A kernel matrix can be transformed into a weighted graph representation, $\mathcal{G} = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Each node in the graph corresponds to a data point (or a subsequence) in the time series whereas each edge is weighted by the similarity value encoded in the kernel matrix $\boldsymbol{K}$. Because an anomalous node is highly dissimilar to other nodes, the weights of its connections to the rest of the nodes in the graph are generally small. Such a node will be rarely visited when performing a random walk traversal on the graph. This is the fundamental idea behind our graph-based anomaly detection algorithm. For example, node a in Figure 5.2 is an anomalous node due to its low connectivity compared to other nodes in the graph.

To estimate the connectivity value of a node, we view the graph as a Markov chain on the state space V with a transition matrix $\boldsymbol{S}$, whose $(i,j)_{th}$ element denote the transition probability from node i to node j. The transition matrix is obtained by normalizing each column of the kernel matrix.

$$\boldsymbol{S}(i,j) = \frac{\boldsymbol{K}(i,j)}{\sum_{i=1}^{n} \boldsymbol{K}(i,j)} \tag{5.3}$$

The connectivity value of each node is computed iteratively by applying the following recursive

Table 5.1. Connectivity values obtained by applying a random walk algorithm on the graph shown in Figure 5.2. The lower the connectivity value, the more anomalous a node is.

a	b	c	d	e
0.05	0.23	0.28	0.14	0.32

equation:

$$\boldsymbol{c} = d/n + (1-d)\boldsymbol{S}\boldsymbol{c}, \tag{5.4}$$

where d is the damping factor and $\boldsymbol{c}$ is a connectivity vector for all the nodes in the graph. This iterative procedure can be viewed as performing a random walk on the Markov chain, where given the current node u, there is a probability $1-d$ of visiting one of its neighboring nodes according to the transition matrix $\boldsymbol{S}$ and a probability d of visiting any random node in the graph. This approach is equivalent to the the formulation used by the PageRank algorithm [163]. Upon convergence, nodes with high connectivity $\boldsymbol{c}$ are considered normal whereas those with low connectivity are declared as anomalous. As an example, Table 5.1 shows the connectivity values of the nodes given in Figure 5.2. Node a is successfully detected as an anomaly because it has the lowest connectivity value.

5.3 Anomaly Detection and Characterization in Noisy Multivariate Time Series

This section investigates the problem of anomaly detection and characterization in noisy multivariate time series. We start with an introduction of multivariate kernel alignment method.

5.3.1 Multivariate Kernel Alignment

As previously noted in Section 5.1.1, the nominal behavior of a multivariate time series can be determined based on the historical evolution of each time series as well as their relationships with each other. A kernel matrix may capture the similarity between measurements at different timestamps but not the relationship between the different time series. To apply the graph-based algorithm described in the previous section to multivariate time series, the kernel matrix must

be adjusted to capture the dependence relationships among the time series in $\mathcal{D}$. This can be accomplished by "aligning" their corresponding kernel matrices.

We begin with a discussion of how kernel alignment works when one of the time series corresponds to a target variable of interest. An extension of the framework to more general multivariate time series problems, where all the time series are equally important, is described in Section 5.3.2. Let $\boldsymbol{K}$ be the initial kernel matrix constructed from the set of predictor variables $\boldsymbol{X} \in \mathcal{D}$ (using Equation (3.1)) and $\boldsymbol{K_Y}$ be the corresponding kernel matrix computed from the target time series. The objective of kernel alignment is to derive an adjusted kernel matrix $\widehat{\boldsymbol{K}}_\alpha$ from $\boldsymbol{K}$ that maximizes its correlation to the target kernel matrix $\boldsymbol{K_Y}$, i.e.:

$$\max_{\boldsymbol{\alpha}} \frac{< \widehat{\boldsymbol{K}}_\alpha, \boldsymbol{K_Y} >_F}{\sqrt{< \widehat{\boldsymbol{K}}_\alpha, \widehat{\boldsymbol{K}}_\alpha >_F < \boldsymbol{K_Y}, \boldsymbol{K_Y} >_F}} \tag{5.5}$$

where $< \boldsymbol{A}, \boldsymbol{B} >_F = \sum_{ij} A_{ij} B_{ij}$ is the Frobenius product between two matrices. The aligned matrix $\widehat{\boldsymbol{K}}_\alpha$ is obtained by first decomposing $\boldsymbol{K}$ into a set of basis vectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n\}$ and learning the corresponding weights α_i associated with each basis vector:

$$\max_{\boldsymbol{\alpha}} \frac{< \sum_{i=1}^{p} \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F}{m\sqrt{< \sum_{i=1}^{p} \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i', \sum_{i=1}^{p} \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i' >_F}}, \tag{5.6}$$

where $m =< \boldsymbol{K_Y}, \boldsymbol{K_Y} >_F$. The resulting aligned matrix is:

$$\widehat{\boldsymbol{K}}_\alpha = \sum_{i=1}^{p} \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i'$$

We consider two ways to decompose $\boldsymbol{K}$ into its basis vectors. The first approach extracts the normalized eigenvectors of $\boldsymbol{K}$ by solving the eigenvalue equation $\boldsymbol{K}\boldsymbol{v} = \lambda \boldsymbol{v}$, subject to the constraint $\boldsymbol{v}_i' \boldsymbol{v}_j = 1$ if $i = j$ and zero otherwise. As a result, kernel alignment reduces to the following optimization problem:

$$\max_{\boldsymbol{\alpha}} \frac{\sum_{i=1}^{n} \alpha_i < \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F}{m\sqrt{\sum_{i=1}^{n} \alpha_i^2}} \tag{5.7}$$

or equivalently,

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i < \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F - \mu \left[\sum_{i=1}^{n} \alpha_i^2 - 1 \right] \tag{5.8}$$

which has the following closed form solution [60]:

$$\alpha_i = \frac{< \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F}{\sqrt{\sum_i < \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F^2}} \tag{5.9}$$

One potential problem with this formulation is that the constraint $\sum_i \alpha_i^2 = 1$ may overfit $\boldsymbol{K_Y}$ and loses its similarity to the original kernel matrix $\boldsymbol{K}$. To overcome this problem, we propose a new objective function:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i < \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F - \mu \left[\sum_{i=1}^{n} (\alpha_i - \lambda_i)^2 \right] \tag{5.10}$$

where $\lambda_i, \boldsymbol{v}_i$ are the corresponding eigenvalues and eigenvectors of the kernel matrix $\boldsymbol{K}$. With this modification, the aligned kernel matrix $\widehat{\boldsymbol{K}}$ will preserve as much information in $\boldsymbol{K}$ as possible. The weight parameters α_i can be computed in closed form as follows:

$$\alpha_i = \lambda_i + \frac{< \boldsymbol{v}_i \boldsymbol{v}_i', \boldsymbol{K_Y} >_F}{2\mu} \tag{5.11}$$

The preceding equation shows that α_i increases when there is strong positive correlation between $\boldsymbol{v}_i \boldsymbol{v}_i'$ and the target kernel $\boldsymbol{K_Y}$. Furthermore, if $\mu \rightarrow \infty$, $\alpha_i \rightarrow \lambda_i$, which reduces $\widehat{\boldsymbol{K}}$ back to the original kernel matrix $\boldsymbol{K}$.

An alternative approach to using eigenvectors is to simply replace $\boldsymbol{v}$ by the time series for each predictor variable, i.e., $\boldsymbol{v}_i = \boldsymbol{X}_i$. In this case, the objective function to be maximized is

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{p} \alpha_i < \boldsymbol{X}_i \boldsymbol{X}_i', \boldsymbol{K_Y} >_F$$
$$-\mu \left[\sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j < \boldsymbol{X}_i \boldsymbol{X}_i', \boldsymbol{X}_j \boldsymbol{X}_j' >_F - 1 \right]$$

For brevity, we choose $\mu = 1$ for our experiments. Taking its derivative with respect to α_i, we obtain:

$$< \boldsymbol{X}_i \boldsymbol{X}_i', \boldsymbol{K_Y} >_F = \sum_{j=1}^{p} \alpha_j < \boldsymbol{X}_i \boldsymbol{X}_i', \boldsymbol{X}_j \boldsymbol{X}_j' >_F \tag{5.12}$$

The weight parameters α_j can be estimated by solving a linear regression problem. Since α_j is associated with the original variable $\boldsymbol{X}_j$, a large weight of α_j indicates a strong correlation between $\boldsymbol{X}_j$ and the target variable $\boldsymbol{Y}$.

Once the weight parameters have been determined, the aligned kernel matrix $\widehat{\boldsymbol{K}}_\alpha = \sum \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i'$ is constructed. Note that elements of $\widehat{\boldsymbol{K}}_\alpha$ may become negative when there are negative values in the basis vector. Although the random walk approach is still applicable in such a case, the kernel matrix $\widehat{\boldsymbol{K}}_\alpha$ can no longer be interpreted as a transition matrix. Our strategy is to make all

Algorithm 6 Multivariate Time Series Anomaly Detection

Input: Multivariate time series $\mathcal{D}$.

Output: Connectivity vector $\boldsymbol{c}$.

1: $\widehat{\boldsymbol{K}}_\alpha \leftarrow$ KernelAlign($\mathcal{D}$)

2: $\boldsymbol{S} \leftarrow$ Normalize($\widehat{\boldsymbol{K}}_\alpha$)

3: $\boldsymbol{c} \leftarrow$ RandomWalk($\boldsymbol{S}$)

the elements of $\widehat{\boldsymbol{K}}_\alpha$ non-negative by adding a sufficiently large number to each element of the matrix. This is equivalently to adding a constant weight to every edge in the graph. However, if the added weight is too large, then the weights for all the edges will become close to uniform values after normalization. To avoid this problem, the elements of the matrix are shifted such that the minimum value of the matrix becomes zero:

$$\widehat{\boldsymbol{K}}_\alpha \leftarrow \widehat{\boldsymbol{K}}_\alpha + |\min(\widehat{\boldsymbol{K}}_\alpha)|, \text{if } \min(\widehat{\boldsymbol{K}}_\alpha) < 0$$

The graph-based algorithm described in Section 5.2.2 will applied to the shifted aligned kernel matrix to detect anomalies.

5.3.2 Multivariate Time Series Anomaly Detection and Characterization Algorithms

Here we propose our multivariate time series anomaly detection algorithms. A high-level summary of the algorithm is shown in Algorithm 5.3.2. There are two variations to the proposed algorithm. The first variation is designed to identify anomalies in a target time series whereas the second variation is designed to identify anomalies in a general multivariate time series, where each variable is equally important. Their difference lies in the way $\widehat{\boldsymbol{K}}_\alpha$ is computed.

For target specific anomaly detection, we first align the kernel matrix derived from the predictor variables with the kernel matrix of the target variable, as described in Section 5.3.1. To perform the alignment, the original kernel matrix $\boldsymbol{K}$ can be decomposed into its basis vectors before applying Equations (5.11) or (5.12) to determine the weight parameters α. We then construct a transition matrix $\boldsymbol{S}$ from the aligned kernel $\widehat{\boldsymbol{K}}$ by normalizing the columns of the matrix. Finally, a Markov chain random walk algorithm is performed on the graph associated with the aligned kernel matrix to obtain the connectivity values of each node (which may represent a timestamp or

subsequence of the time series). Since the connectivity values depend on the similarity measure, number of nodes, and other factors, we normalize the connectivity scores by subtracting their means and dividing their standard deviations. Nodes that have the connectivity scores below certain threshold are declared as anomalies. The impact of kernel alignment step for target specific anomaly detection in multivariate time series is two-fold:

- It tends to extract the components in the predictor time series that are most related to the anomalies in target time series at the same time period. In this way, the anomalies found in the predictor time series after alignment can be used to characterize the corresponding anomalies in target time series.

- The components in predictor time series that are not related to target time series will be removed after kernel alignment, which leads to two effects: first, the anomalies in predictor time series which are independent of the target time series will be degraded; second, the impact of noise in the multivariate time series will be reduced.

In such a way, the anomalies found in predictor time series can be used to characterize the anomalies discovered in target time series.

For general multivariate time series anomaly detection, where $\mathcal{D} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_p]$, we consider each variable $\boldsymbol{X}_i$ as a target and learn an aligned kernel matrix $\widehat{\boldsymbol{K}}_i$ between the target and all the remaining variables in $\mathcal{D}$. After alignment, this produces p aligned kernel matrices. The overall aligned kernel matrix $\widehat{\boldsymbol{K}}_\alpha$ is computed by taking the Hadamard product of the individually aligned kernel matrices $\widehat{\boldsymbol{K}}_i$:

$$\widehat{\boldsymbol{K}}_\alpha = \widehat{\boldsymbol{K}}_1 \circ \widehat{\boldsymbol{K}}_2 \cdots \circ \widehat{\boldsymbol{K}}_p \tag{5.13}$$

Once the overall aligned kernel has been computed, the remaining steps are the same as those taken for target-specific anomaly detection. Again, the noise in multivariate time series is removed after kernel alignment such that false alarm rate can be reduced.

5.4 Extensions of the Proposed Algorithm

This section describes two extensions of the proposed algorithms: (1) to detect local anomalies in time series data, and (2) to discover subsequence-based anomalies. Although we discuss it in the

framework of multivariate time series anomaly detection, it is also applicable to univariate time series.

5.4.1 Sparse Kernel for Local Anomaly Detection

Our aligned kernel matrix is used to construct a graph $\mathcal{G}$ upon which a random walk algorithm is applied to identify anomalous nodes in the graph. $\mathcal{G}$ is a completely connected graph whose edge weights depend on the values of the kernel matrix $\widehat{\boldsymbol{K}}_\alpha$. As a result, the anomalies found correspond to global anomalies whose values are significantly different than the rest of the time series. Our proposed algorithm can be extended to detect local anomalies by sparsifying the kernel matrix prior to constructing the transition matrix $\boldsymbol{S}$.

As mentioned in Section 5.1.1, we have implemented two ways to define the neighborhood of a node for local anomaly detection. The first approach, time-based neighborhood, is implemented by removing the edges between all pairs of nodes i and j in which $|i-j| > k$. Thus, a local anomaly observed at time t has significantly different values than other observations within the time interval $[t-k, t+k]$. For the second approach, i.e., cycle-based neighborhood, we sparsify the graph by removing all edges in $\mathcal{G}$ in which $|i-j| \mod k > \tau$. For example, setting $\tau = 0$ would compare an observation at time t to other observations at times $t-k, t-2k, \cdots$ and $t+k, t+2k, \cdots$, where k is the known periodicity of the time series. Cycle-based approach has great applications in non-stationary time series data with periodical patterns. For example, it can be used to detect anomalies in the monthly FPAR or climate time series by comparing the values for the same month in different years.

5.4.2 Subsequence Anomaly Detection

In addition to point-wise anomalies, our algorithm can also be extended to discover subsequence anomalies (also known as discords). Let $\boldsymbol{Y}$ be the target variable and $\boldsymbol{X}$ be the set of predictor variables. Assume T is the length of all the time series. A sliding window of predefined size d is used to extract all the subsequences of length d from the multivariate time series. The corresponding predictor and target variable subsequences are represented as $\overline{\boldsymbol{X}}$ and $\overline{\boldsymbol{Y}}$, respectively. The number of subsequence windows created from the time series data is $T-d+1$. Each element

of $\overline{X}$ and $\overline{Y}$ is a subsequence with length d, which is denoted as:

$$\begin{aligned} \overline{x}_{ij} &= \{x_{i,j}, x_{i,j+1}, \ldots, x_{i,j+d-1}\} \\ \overline{y}_j &= \{y_j, y_{j+1}, \ldots, y_{j+d-1}\} \end{aligned}$$

The kernel matrices $\boldsymbol{K}$ and $\boldsymbol{K}_Y$ are constructed by applying the following Equations:

$$\begin{aligned} \boldsymbol{K}(i,j) &= \exp(-\frac{\sum_{k=1}^{p} ||\overline{x}_{ki} - \overline{x}_{kj}||^2}{\sigma^2}) \\ &= \exp(-\frac{\sum_{k=1}^{p} \sum_{s=0}^{d-1} (x_{k,i+s} - x_{k,j+s})^2}{\sigma^2}) \end{aligned}$$

and

$$\boldsymbol{K}_Y(i,j) = \overline{y}_i \overline{y}_j' = \sum_{s=0}^{d-1} y_{i+s} y_{j+s}$$

The (i,j)th element in the kernel matrix represents the similarity between two subsequences (i.e., sliding windows) of length d starting from the timestamps i and j. Similar to the approach described for point-wise anomaly detection, the kernel matrices can be aligned before applying the random walk algorithm to detect the anomalous subsequences. While the approach presented in this section assumes a fixed window size d, it can be easily extended to deal with variable length time windows as well as other similarity measures for subsequences.

5.5 Visual Exploration of Ecosystem Disturbances

In previous sections, we have introduced a graph-based method for anomaly detection and characterization in multivariate time series data. However, applying these methods for ecosystem disturbance detection in large scale Earth Science data directly is very expensive because of the massive data size and the requirement for parameter tuning. Visual exploratory analysis provides a way to alleviate this problem by presenting only small portion of the data each time and enable user interaction during the mining process. Here we present a visualization system for the mining and exploration of ecosystem disturbance in Earth Science data. We first present a system to cluster the regions of similar disturbance events. Then a multi-level indexing framework is proposed to aid the expiatory analysis of ecosystem disturbance in the global FPAR time series repository.

Table 5.2. Features for clustering disturbance events

Features	Description
Location	Latitude and longitude of a location
Period	Number of months of consecutively low FPAR values
First month	Month in which disturbance event was detected
FPAR (During)	Minimum and range of FPAR values during period of disturbance event
FPAR (After)	Minimum and range of FPAR values after period of disturbance event
Climate	Precipitation and temperature values at the onset of disturbance event
Land cover	Land cover type at the location

5.5.1 Clustering of Ecosystem Disturbance Events

Clustering is the task of partitioning data into groups of similar objects. In Earth Science studies, clustering has been applied for various applications including land cover classification [202, 207], classification of synoptic-scale circulation patterns [59], and discovery of climate indices [193]. Clustering can also be used to aggregate related (and possibly nearby) data points that have similar characteristics (e.g. climate conditions, landscape variability, etc). Since it does not require any labeled examples, clustering is considered an unsupervised learning task, as opposed to supervised learning tasks such as classification and regression.

In this work, we apply clustering to group together locations that exhibit similar types of disturbance events. The clusters may help users to automatically categorize the different types of disturbance events (e.g. wildfires, insects, deforestation, etc) based on characteristics of the eco-climatic time series during or after the event has occurred. Since there are no labeled examples available, a key challenge is to determine which attributes are appropriate to characterize the different types of disturbance events. Table 5.2 shows some of the attributes available to describe the characteristics of disturbance events. The user may cluster the data points using any subset of these attributes. Since each attribute can have a different scale, we need to normalize the attribute values before applying the clustering algorithm. Otherwise, the similarity computation between data points will be dominated by attributes that have a large range of possible values. Continuous-valued attribute are normalized by subtracting each attribute with its minimum possible value and dividing it by the range of its possible values. For each discrete attribute, we normalize it by creating a binary attribute for each attribute-value pair. Normalization allows each attribute selected for the clustering task to be treated equally important during similarity computation.

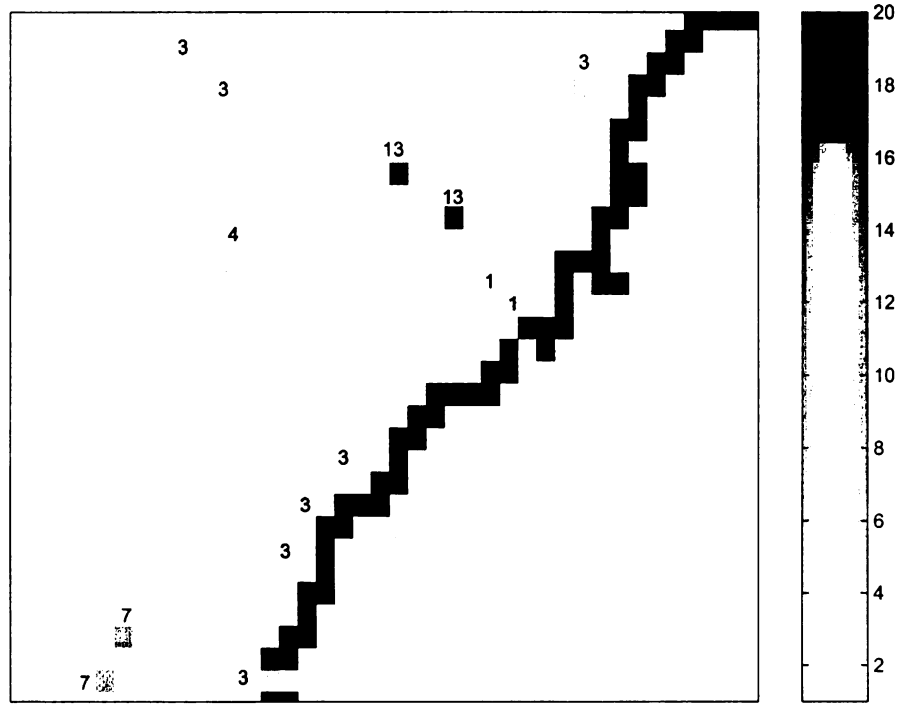
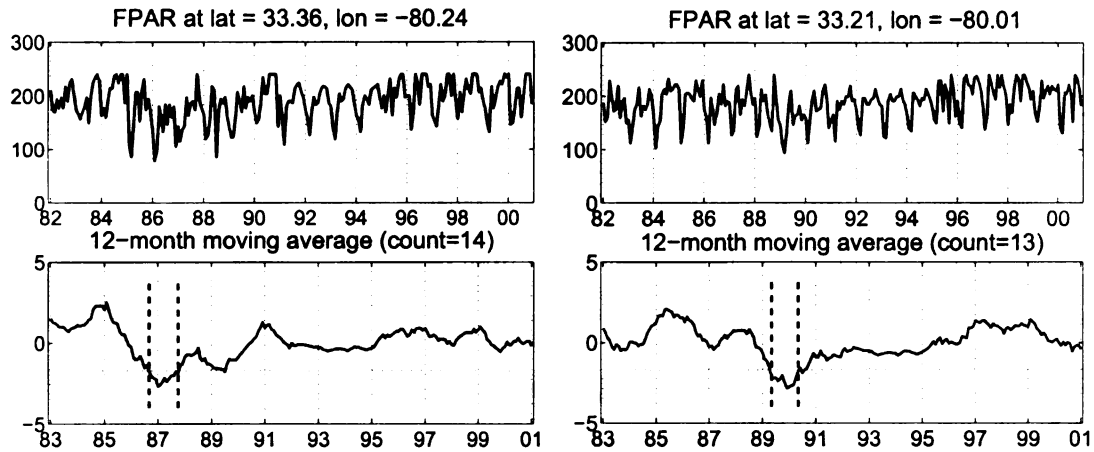


Figure 5.3. Clusters of ecosystem disturbance events detected in the North Carolina region (each symbol corresponds to a different cluster).

5.5.2 Clustering for Exploratory Data Analysis

The methods described in the previous section had been successfully applied to the FPAR time series data from AVHRR. With the availability of the higher resolution MODIS data, applying the disturbance event detection algorithm on all the data points can be very expensive. For example, with the 4 km × 4 km MODIS data, there are more than 1.3 million data points for North America alone. The total number of FPAR time series for the entire world is more than 10.1 million. Although the disturbance event detection algorithm scales linearly with the size of the data, it still takes more than a few minutes to process the data for North America. Repeating the analysis using different thresholds will take a considerable amount of time, making it infeasible for real-time exploration of the data. The runtime will increase considerably if the analysis is to be performed on the 1 km × 1 km MODIS data or using more sophisticated (and expensive) anomaly detection algorithms. Techniques must therefore be developed to reduce the amount of processing time.

For this experiment, we have performed K-means clustering (with k=20) on all disturbance event locations in North America using precipitation and temperature as data attributes. Figure



(a) FPAR at 32.85°N and 79.84°S (cluster 1)

(b) FPAR at 33.21°N and 80.01°S (cluster 13)

Figure 5.4. FPAR time series at different locations.

5.3 shows clusters of disturbance events found near the region of North Carolina. Each colored pixel corresponds to a location where disturbance event has been previously detected. We have also labeled each colored pixel according to its cluster ID. Our preliminary results suggest the possibility of distinguishing the different types of ecosystem disturbance events in the region according to their cluster assignment. For example, Figure 5.4 shows two FPAR time series for one data point assigned to each cluster #1 and #13, respectively. Further investigation revealed that the data points assigned to cluster #1 are associated with disturbance events due to Hurricane Hugo (which occurred in September 1989). Although the disturbance event found for the data point shown in Figure 5.4(b) also occurs in 1989, it appears earlier in the year, compared to the disturbances found in cluster #1.

A. Proposed Clustering Framework

In this work, we propose to develop a clustering-based framework to reduce the number of time series that needs to be processed in order to facilitate interactive exploration of the large-scale data. Specifically, our strategy is to build a multi-level index on the MODIS FPAR time series by applying clustering on the time series and choosing representative samples from the clusters at each level of the index hierarchy. Therefore, instead of applying the moving average or graph-based disturbance detection algorithm on the entire time series, we would approximate the frequency distribution of disturbance events in each region by applying the algorithm to the selected sam-

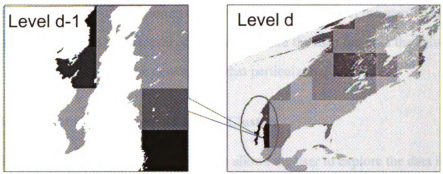


Figure 5.5. A multi-level approach for visual exploration of ecosystem disturbances.

ples. This helps to reduce the amount of processing time considerably, thus allowing the user to tune the thresholds of their algorithm and to observe the changes in the results in real-time. It will also help the user to focus on a particular region of interest during exploratory data analysis.

Figure 5.5 shows an example of how the multi-level indexing scheme works. Specifically, we use the multi-level index to monitor the frequency distribution of ecosystem disturbance events in North America. We use North America for illustrative purposes only. Multi-level indices for other continents are also created. At the top level, the entire continent is partitioned into a 7×10 grid box, where each box contains 256×256 pixel locations. Next, each of the 7×10 boxes is further divided into smaller 4×4 grid boxes, where each of the smaller box now contains 64×64 pixel locations. At the next level, each of these smaller boxes is partitioned into another 4×4 grid boxes, each of which now contains 16×16 pixel locations. A final partitioning of each grid box produces a square region that contains 4×4 pixel locations. A map displayed at this level will show the actual locations where disturbance events have been observed. At all other higher levels, the map displays the frequency distribution of disturbance events in each grid box (region).

For example, in Figure 5.5, the map at level d displays the frequency distribution of disturbance events for each region in North America. As can be seen from the map, the Baja California peninsula, the midwest region, and Canada appear to have highest concentration of disturbance event locations. The user may decide to focus only on these regions and may click on the grid box that corresponds to one of these regions (say California) for further analysis. The map for the California peninsula will now be displayed (level $d - 1$). The region is now divided into 4×4 smaller regions; this allows the user to examine the regions where there is a dense concentration of

disturbance events (e.g. the dark region in the lower right hand corner of the map). This process is repeated until it reaches the lowest level of the index, where the map displays the actual locations where disturbance events have been detected for that particular region.

B. Sampling Representative Time Series from Clusters

The multi-level indexing scheme described above allows the user to explore the data interactively in a hierarchical fashion. Computation time can still be very expensive because the number of time series in each grid box grows quadratically from one level (m) to the next level ($m + 1$). For example, there are 256 pixel locations in each grid box at level 1 and 65,536 pixel locations in each grid box at level 3. To reduce the computation time at higher levels of the index, instead of applying the disturbance event detection algorithm to all the time series in the grid box, we apply the algorithm on samples of time series for that region. The key challenge here is to determine which time series should be selected to obtain a representative sample. A better representative sample must capture more disturbance events in the region. Another issue to consider is the tradeoff between coverage and processing time, which depends on the sample size. If the sample size is large, this will provide a good coverage for the region but at the expense of increasing the processing time. Our experiments on the 4km $\times$ 4km MODIS FPAR data suggest that a sample size containing 100 points per grid box will produce reasonable coverage and processing time.

If N time series must be sampled from a given grid box, we perform clustering on all the time series located in the grid box to obtain k clusters. We then select $\lceil N/k \rceil$ representative time series from each cluster to form the sample for the region. We investigate several sampling approaches in this study. For example, two basic sampling technique is considered:

- **Closest Sampling** would sample time series that are closest to the cluster centroid.
- **Farthest Sampling** would sample time series that are farthest away from the cluster centroid.

Since we are interested in the detection of anomalous events, farthest sampling may be useful to focus on time series that deviate significantly from others in the cluster.

During the construction of the multi-level index, the pixel locations of the sampled time series are stored in the index structure. The samples are retrieved when the grid box is selected by the

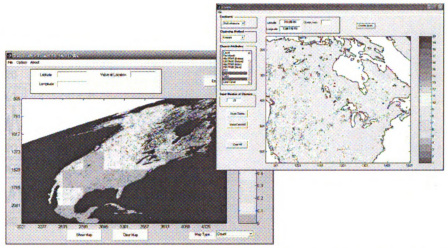


Figure 5.6. An interactive visual data mining tool for detection and characterization of ecosystem disturbances.

user during exploratory data analysis. The disturbance event detection algorithm is then applied to the samples and their results will be displayed on the map (as shown in Figure 5.5.

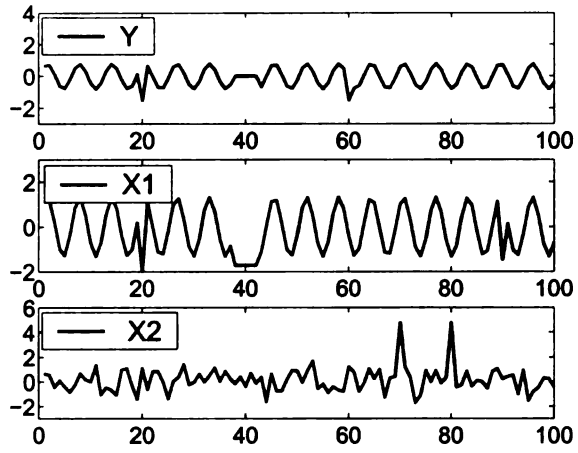
To evaluate the effectiveness of the sampling strategy, we compute the fraction of disturbance events missed due to sampling:

$$\text{Loss} = \frac{1}{d} \sum_{l=1}^{d} \sum_{b_i \in \Omega_l} \frac{N_a(b_i;\theta) - N_s(b_i;\theta)}{N_a(b_i;\theta)} \tag{5.14}$$

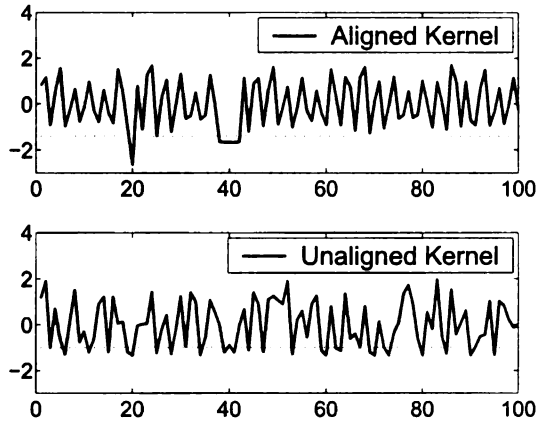
where d is the number of levels, Ω_l is the set of grid boxes at level l, θ is the event definition threshold, $N_a(b_i;\theta)$ is the actual number of disturbance events in the grid box, and $N_s(b_i;\theta)$ is the corresponding number of disturbance events in the sampled time series.

5.5.3 A User-Interactive System for Disturbance Event Detection

Figure 5.6 shows a snapshot of the interactive system that integrates the disturbance event detection and clustering algorithms developed in this study. The algorithms and user interface are developed using the Matlab programming language. Initially, the map will display the frequency distribution of disturbance events for the selected continent (e.g. North America, as shown in Figure 5.6). The continent is initially divided into a 7×10 grid box. The color of the pixels on the map represents the fraction of locations in each grid box that contain disturbance events. The user may decide to select a grid box that contains a high percentage of disturbance events for further exploration. The viewer will zoom in to the selected grid box and scanned all the subgrids for



(a) Three simulated time series: target Y with two predictor variables X_1 and X_2,for measuring the effectiveness of target specific anomaly detection.



(b) Anomaly scores obtained by applying a random walk on the aligned (top) and unaligned (bottom) kernels for time series in Figure 5.7(a).

Figure 5.7. Target specific anomaly detection.

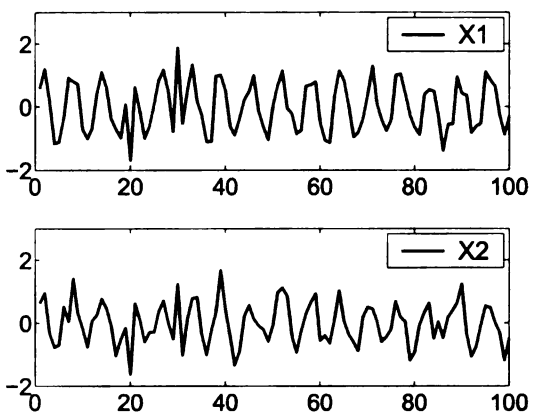
that region in the next higher resolution level. Also, our viewer enable users to return to the last level by right clicking the map and choosing a different grid at a lower resolution level. Users can repeat this process until the highest level is reached where each grid represents a true location with latitude and longitude. At the highest resolution level, users can click a single location and the viewer will plot the time series and return anomaly score for each temporal data point.

5.6 Experimental Result

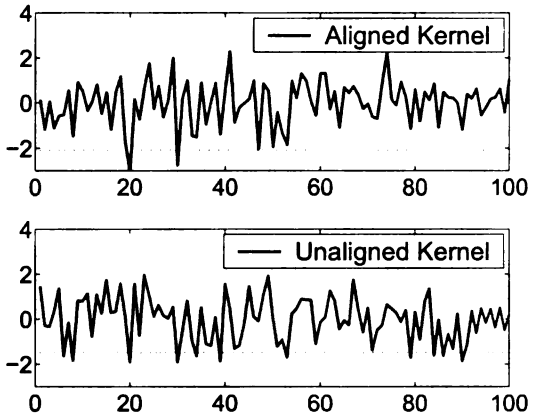
We perform several experiments to evaluate the performance of our proposed techniques in this work.

5.6.1 Target Specific Anomaly Detection

The objective of this experiment is to illustrate the effectiveness of applying kernel alignment for target specific anomaly detection in multivariate time series. We simulated three equal length time series as shown in Figure 5.7(a). The top diagram represents the target variable Y whereas the bottom two time series correspond to the predictor variables X_1 and X_2. Both X_1 and Y are generated by interjecting large amplitude pulses to a periodic sinusoidal time series. The time



(a) Two simulated time series $X1$ and $X2$ with noise for measuring effectiveness of general multivariate time series anomaly detection.



(b) Anomaly scores obtained by applying a random walk to the aligned (top) and unaligned (bottom) kernels for time series in Figure 5.8(a).

Figure 5.8. General multivariate noisy time series anomaly detection.

series for X_2 is generated by interjecting large amplitude pulses and modifying the local shape of a random time series generated from the normal distribution $N(t, 0.8)$. The diagram clearly shows that two of the anomalies in the target variable Y (at timestamps 20 and 40, respectively) can be explained by the anomalies found in X_1. The third anomaly in the target variable Y (at timestamp 60) is unrelated to any anomalies observed in the predictor variables X_1 and X_2. Furthermore, the anomalies at timestamp 90 of the predictor variable X_1 and at timestamps 70 and 80 of the predictor variable X_2 do not correspond to any anomalies in Y.

Figure 5.7(b) shows the results of applying our target-specific anomaly detection algorithm on the aligned ($\widehat{\boldsymbol{K}}_\alpha$) and unaligned ($\boldsymbol{K}_{\boldsymbol{X}}$) kernel matrices derived from the combined predictor variables, X_1 and X_2. Anomalies are detected based on the timestamps at which the connectivity values are below a certain threshold (after standardization). The results clearly demonstrate the effectiveness of applying kernel alignment to learn the dependence relationship between the predictor and target variables. Without kernel alignment, the correlated anomalies at timestamps 20 and 40 are indistinguishable from other timestamps in the time series, which may lead to high false positive or false negative rates (depending on the threshold chosen). By applying kernel alignment, anomalies that are correlated with those observed in Y (at timestamps 20 and 40) are amplified, which suggests that they can be used to characterize the anomalies found in the target variable. The anomalies found at timestamps 70 and 80 in the predictor variable X_1 and the

anomaly found at timestamp 90 in predictor variable X_2 are degraded since they do not align with the target variable Y. Similarly, the anomaly at timestamp 60 in the target variable is also not detected because it cannot be explained by any anomalies in the predictor variables.

This experiment uses the original predictor time series as the basis vectors for the aligned kernel (see Section 5.3.1). The weights associated with the predictor variables are $\alpha_1 = 0.9964$ and $\alpha_2 = 0.0844$, respectively. These parameters suggest that X_1 is more correlated to the target variable Y than X_2, which is consistent with our observation.

5.6.2 General Multivariate Noisy Time Series Anomaly Detection

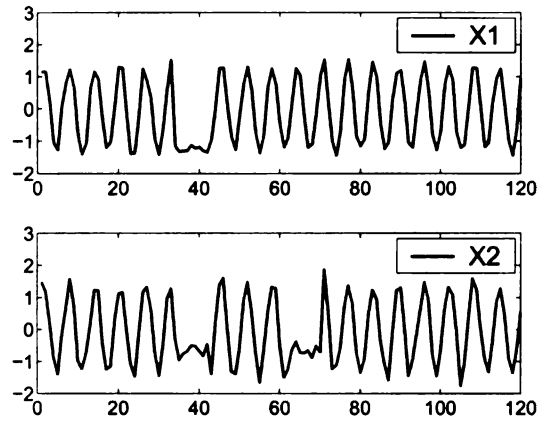
The purpose of this experiment is to demonstrate the effectiveness of applying kernel alignment on anomaly detection for multivariate time series without a target variable. Figure 5.8(a) shows two simulated time series X_1 and X_2 with a pair of anomalies at timestamps 20 and 30, respectively. A Gaussian noise ($N(0, 0.3)$) is also added to each time series to distort the signals.

The resulting anomaly score is plotted in Figure 5.8(b). Without kernel alignment, the random walk algorithm has difficulty in discriminating the anomalies at timestamps 20 and 30 from other noisy signals in the time series. By applying kernel alignment (using Equation (5.13)), the random walk algorithm can successfully detect both anomalies in the noisy time series. This experiment suggests that kernel alignment helps to improve the kernel matrix used by the anomaly detection algorithm by reducing the effect of noise.

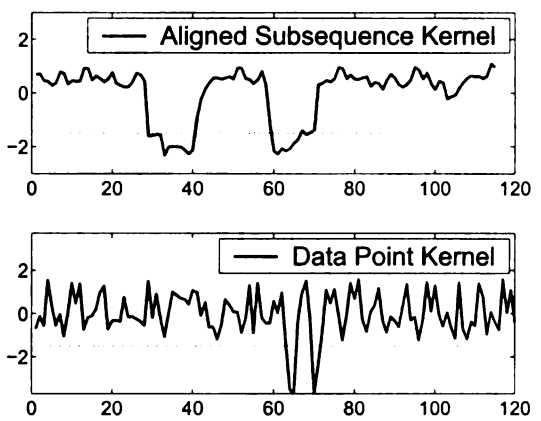
5.6.3 Subsequence Anomaly Detection

The purpose of this experiment is to demonstrate the effectiveness of our algorithm for finding unusual subsequences in a multivariate time series. We simulated two time series X_1 and X_2 as shown in Figure 5.9(a). A Gaussian noise (with mean zero and standard deviation 0.1) is used to distort both time series. Anomalous subsequences are then added to the time series two time series.

We compare the results of applying random walk anomaly detection algorithm on kernel matrices constructed from the subsequences to those constructed from individual time points. In both cases, the kernel matrices are initially aligned before applying the random walk algorithm. The



(a) Two simulated time series $X1$ and $X2$ with unusual subsequences.



(b) Anomaly scores obtained by applying a random walk to the subsequence (top) and data point (bottom) kernels for time series in Figure 5.9(a).

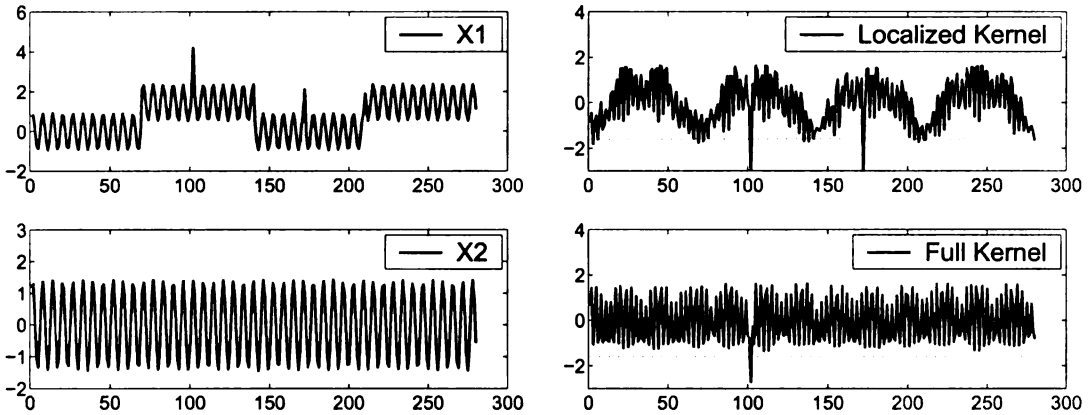
Figure 5.9. Unusual subsequences detection.

subsequence-based kernel matrices are constructed using a sliding window of length 6. Figure 5.9(b) compares the anomaly scores obtained using the aligned subsequence-based kernel matrix (top diagram) to those obtained using the aligned point-wise kernel matrix (bottom diagram). The results obtained using subsequence-based kernel are clearly more superior.

5.6.4 Local Anomaly Detection

This experiment compares the results of applying the full versus sparse kernel matrices for detecting local anomalies. First, we illustrate the results of detecting time-based local anomalies. Figure 5.10(a) shows the two simulated time series used in our experiment. One of the time series, X_1, contains both a global anomaly (at timestamp 100) and a local anomaly (at timestamp 170). We apply the random walk anomaly detection algorithm on the graph induced by the sparse kernel matrix (using time-based neighborhood with $k = 20$), as described in Section 5.4.1. The resulting anomaly scores are plotted in Figure 5.10(b) and compared against those obtained by using the full (global) kernel. The results show that a full kernel may miss the local anomaly because its value at timestamp 170 is similar to other observations in X_1. In contrast, the sparse kernel is capable of detecting both global and local anomalies.

Next, we illustrate the results of detecting cycle-based local anomalies. The two simulated time series used for this experiment are shown in Figure 5.11(a). Both of the simulated time series



(a) Two simulated time series $X1$ and $X2$ with local and global anomalies.

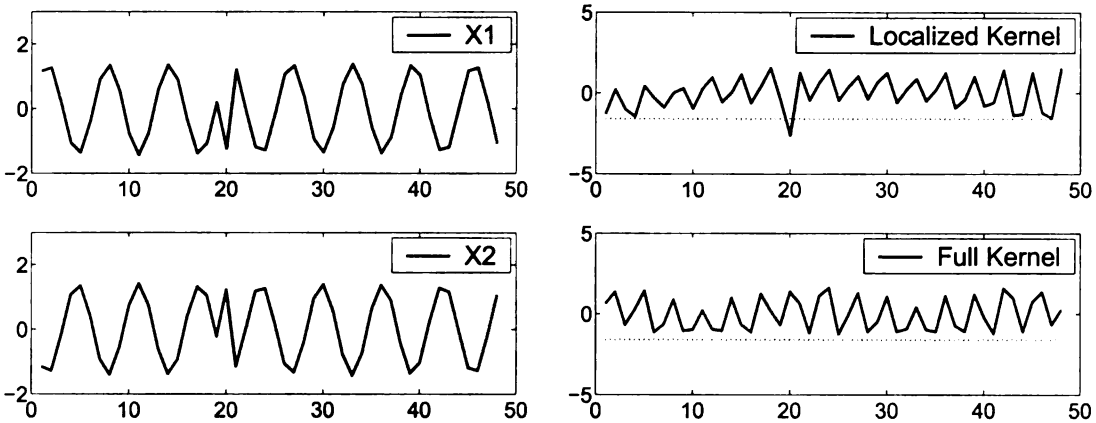
(b) Anomaly scores obtained by applying a random walk to the time-based local (top) and full (bottom) kernel matrices.

Figure 5.10. Time-based local anomaly detection.

contain a local anomaly at timestamp 20 and have a periodicity equals to 6. Clearly, the values of the variables at timestamp 20 are not unusual from a global perspective. As a result, standard distance-based and density-based anomaly detection methods fail to detect such anomalies. Our random walk algorithm using the full kernel matrix also fails as shown in the bottom panel of Figure 5.11(b). However, by sparsifying the kernel matrix based on the cycle-based neighborhood approach, the anomaly at timestamp 20 is successfully detected as shown in the top panel of Figure 5.11(b).

5.6.5 General Performance Comparison

In this experiment, we compare the performance of our general multivariate time series anomaly detection algorithm against several baseline algorithms using benchmark data (the data was originally created by Dasgupta et al. [111] and obtained via a CD from Eamonn Keogh). from the University of California Riverside time series data mining archive [120]. The benchmark data, which is used to test the performance of anomaly detection algorithms, contain 38 distinct time series data. Each time series data has a separate training and test sets. The training set contains a pair of "normal" time series of length 1000. The test set also has a pair of time series of length 1000 in addition to a class label vector that indicates whether a data point in the bivariate time series is anomalous.



(a) Two simulated time series $X1$ and $X2$ with cycle-based anomalies.

(b) Anomaly scores obtained by applying a random walk to the cycle-based local (top) and full (bottom) kernel matrices.

Figure 5.11. Cycle-based local anomaly detection.

We compared our general multivariate time series algorithm (denoted as `Align`) against 4 other competing methods—random walk without alignment [148], probability based [220], LOF [31], and K-distance based methods [176]. Note that the probability based method trains a classifier from the training set and thus is a supervised methods. All other methods, including `Align`, are unsupervised and do not use the training set. The performance of the algorithms is measured in terms of their area under ROC curve (AUC) [82]. Figure 5.12 shows the AUC values for all the methods on the 38 data sets. The results suggest that `Align` significantly outperforms other existing methods for the majority of the data sets. As an example, Figure 5.13 shows the ROC curves for all the methods on one of the data sets in which our random walk algorithm on the aligned kernel matrix has the largest area.

5.6.6 Ecosystem Disturbance Detection

This section describes the results of applying our algorithm to the problem of detecting ecosystem disturbances such as wildfires in Earth Science data. Specifically, ecosystem disturbances are detected by monitoring changes in the vegetation cover data (called FPAR [200]) obtained from satellite observations. The monthly FPAR data is available at 4km $\times$ 4km spatial resolution covering the time period between 2000 and 2005. The objective of this study is to detect ecosystem disturbances using our proposed algorithm and to correlate them against the anomalies observed

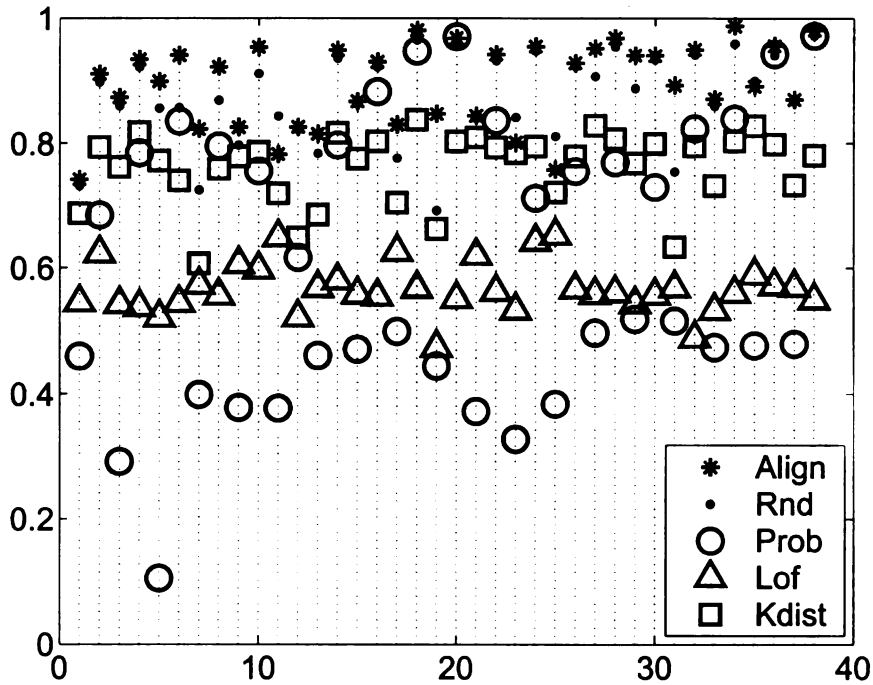


Figure 5.12. Performance comparison using 38 data sets from UC Riverside data repository.

in climate data (such as precipitation, temperature, and sea-level pressure). The FPAR time series is treated as the target variable, whereas the climate time series are used as the predictor variables.

To detect anomalies, we align the kernel matrix constructed from the climate variables against the kernel matrix for FPAR. The aligned kernel is then sparsified using the cycle-based neighborhood approach with $k = 12$ and $\tau = 1$. The anomaly scores for the multivariate time series are then computed by performing a random walk on the graph induced by the kernel matrix.

Using this approach, we were able to detect several incidents of large-scale FPAR disturbance events that correlate with extreme climate conditions. One example is given in Figure 5.14(a), which shows the FPAR time series at a location in Alberta, Canada (latitude=54.85N, longitude=114.20W) along with its anomaly scores obtained by applying the random walk algorithm to the kernel matrix constructed from the FPAR time series. The timing and location of the FPAR disturbance event coincide with the Chisholm wildfire event on May, 2001 as reported [194]. It has also been reported that dry condition was one of the factors causing the severity of the wildfire. Figure 5.14(b) shows the monthly precipitation for the location along with the anomaly scores

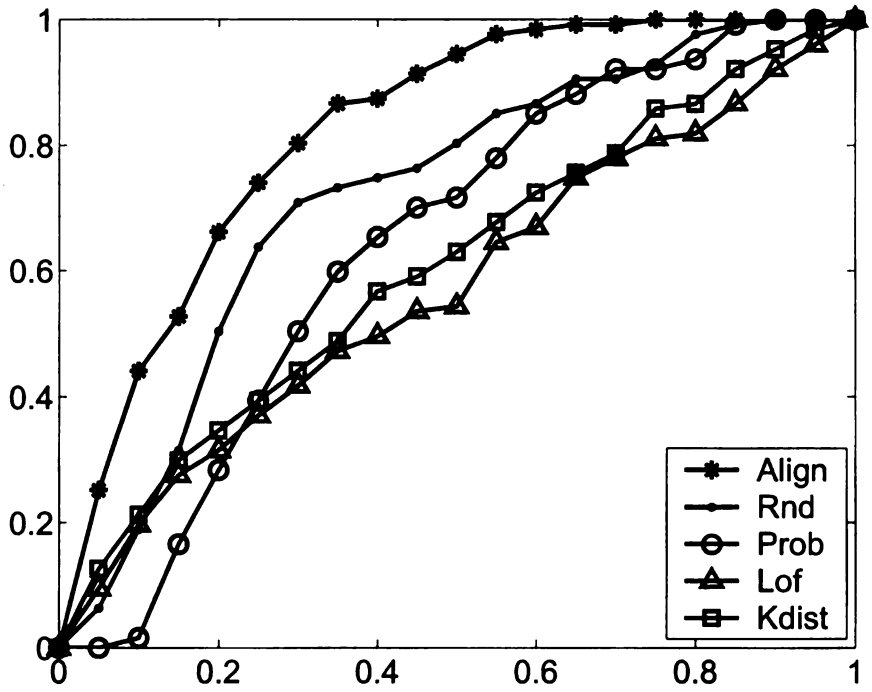
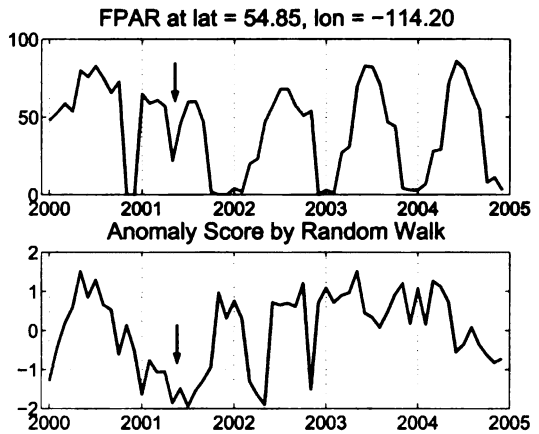


Figure 5.13. An example of ROC curves for all the methods on the 7_{th} data set.

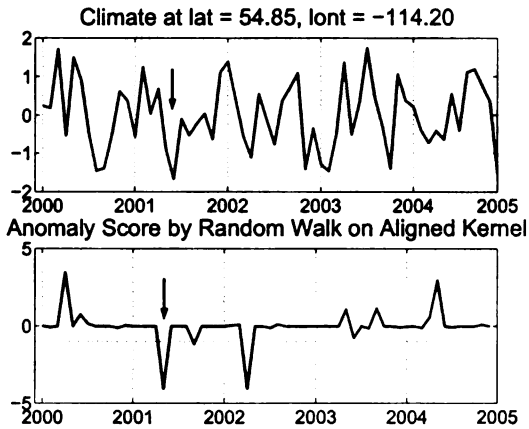
computed from the aligned kernel. Observe that the FPAR disturbance event is well-aligned with the occurrence of a low precipitation event. Our analysis also suggests the possibility of another FPAR disturbance and low precipitation event in Spring 2002, however, more analysis is needed to validate this event.

As another example, Figure 5.15(a) shows the FPAR time series and anomaly score for a location in South Dakota (latitude=35.31N, longitude=111.21W). The timing and location of the disturbance event co-incide with the Antelope wildfire [197] reported in August, 2002. The FPAR anomaly is also associated with a low precipitation event, as shown in Figure 5.15(b). Although there are other unusual precipitation events during this period, only one of them coincides with the FPAR disturbance.

A third example is given in Figure 5.16(a) for the FPAR time series at a location in Honduras (latitude=15.68N, longitude=87.57W). A sudden drop in FPAR was observed at the end of 2001, which correlates with a sudden drop in sea level pressure (see Figure 5.16(b)). According to a report by the National Oceanic and Atmospheric Administration (NOAA) [134], the FPAR disturbance event coincides with the timing of Hurricane Iris.



(a) FPAR and its anomaly score.



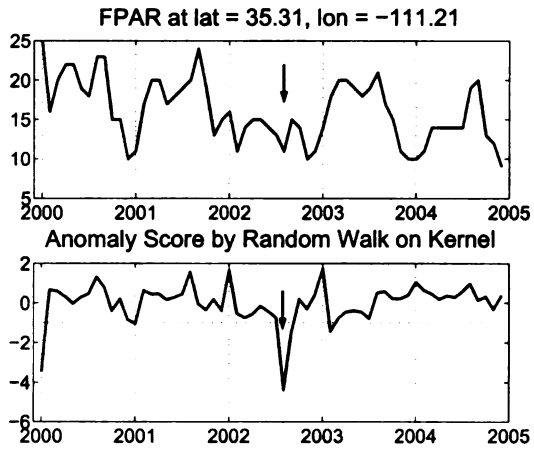
(b) Precipitation and its anomaly score.

Figure 5.14. FPAR and Precipitation time series at $(latitude = 54.85N, longitude = 114.20W)$ and corresponding anomaly score by random walk.

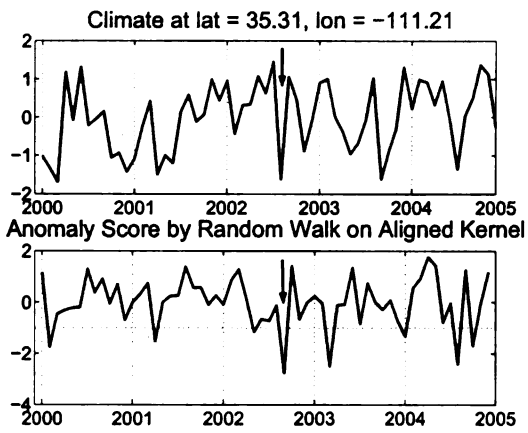
5.6.7 Ecosystem Disturbance Exploratory Analysis

We verify our viewer with several examples of fire events that happened between 2000 to 2005. Figure 5.17(a) shows a subregion in the highest resolution level, which is close to the Pike-San Isabel National Forest, 30 miles southwest of Denver, Colorado. According to a report published in [1], a forest fire hit this region around June 2002. Figure 5.17(b) shows one example location picked from dark color locations in Figure 5.17(a). The raw time series and the anomaly score by moving average method is plotted. Continuous low FPAR-Low events lasting 12 months is successfully detected at those two locations with latitude and longitude $(39.45°N, 105.00°S)$. Figure 5.18(a) shows another subregion in the highest resolution level close to the Alberta, Canada. According to a report published in [194], the Chisholm Fire hit this region around May 2001. Figure 5.18(b) shows an example location picked from dark color locations in Figure 5.18(a). The raw time series and the anomaly score by graph-based method is plotted. Continuous low FPAR-Low events lasting 3 months is successfully detected at those two locations with latitude and longitude $(54.88°N, 114.18°S)$.

Since all our results are computed on the fly, users can adjust the parameters whenever they want to. For example, he can change the length of the sliding window or the threshold in moving average method accord to his/her own purpose as well as the parameters in random walk approach.



(a) FPAR and its anomaly score.



(b) Precipitation and its anomaly score.

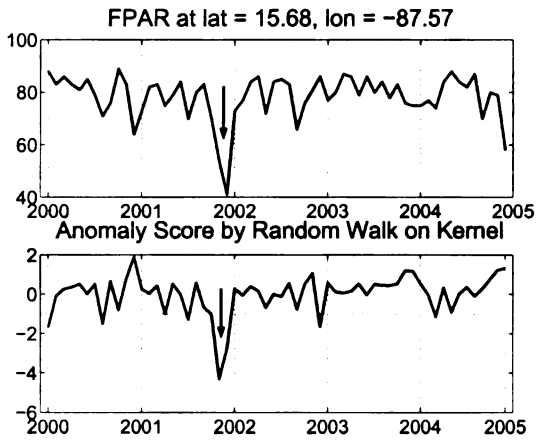
Figure 5.15. FPAR and Precipitation time series at (latitude=35.31N, longitude=111.21W) and corresponding anomaly score by random walk.

5.6.8 Multi-Level Indexing Performance Evaluation

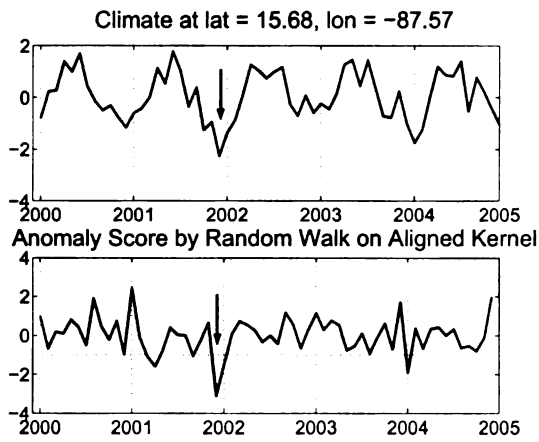
Here we would like to evaluate how different aggregation methods will affect the results during the exploratory analysis on the high resolution FPAR data. The configuration of the system in this evaluation is based on 3 level indexing and 50, 100, 100 samples for level 1, 2, 3. We first evaluate how the system performs when aggregating data from high resolution level to low resolution level using different definitions of samples. Two types of samples are used for K-means clusters: 1. data points closet to the cluster centroid, 2. data points farthest away from the cluster centroid. The metric in formula 5.14 is used to calculate the sample effect when data is aggregated from high resolution level to lower resolution level. We include the results for these methods as well as the random sampling and uniform sampling in Table 5.3. It shows that the loss for random and uniform sampling is more than 77%, which is considerably higher than farthest sampling. This result suggests that, by sampling the time series that are considerably different than the

Table 5.3. Comparison of sampling loss for different strategies.

	Level 1	Level 2	Level 3	avg
Random	0.7679	0.7319	0.8356	0.7784
Uniform	0.7699	0.7196	0.8213	0.7703
Closest	0.7595	0.7008	0.7994	0.7532
Farthest	0.7527	0.6956	0.4243	0.6242



(a) FPAR and its anomaly score.



(b) Sea level pressure and its anomaly score.

Figure 5.16. FPAR and Sea level pressure time series at (latitude=15.68N, longitude=87.57W) and corresponding anomaly score by random walk.

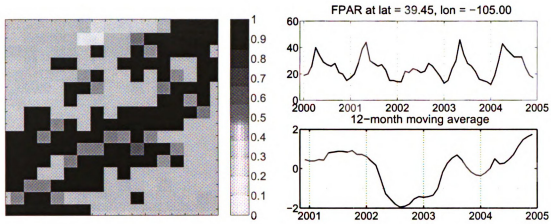
cluster centroid, we are more likely to sample time series that contain anomalies (i.e., disturbance events).

5.6.9 Multi-Level Indexing System Efficiency Evaluation

Table 5.4 shows the total runtime for computing all the disturbance events at different levels using the moving average and random walk methods. The number of grid boxes and the number of time series to scan at each level are also recorded. Level 0, which represents the highest resolution where each grid box is a 4km × 4km location, requires nearly 1.5 hours to scan the entire land points in North America for moving average method. It is even more expensive for random walk method, which requires around 5 hours. Clearly, this is infeasible for exploratory data analysis. After the data is aggregated to level 1 with 50 clusters in each grid box, the time needed to scan all the sampled time series at this level reduces significantly to only 10 minutes for moving average

Table 5.4. Runtime (in seconds) needed to compute disturbance events at each level for North America.

Level	0	1	2	3
Total Time (MV)	5418.8	661.7	98.4	10.9
Total Time (RW)	17710.0	1839.6	254.6	56.0
#Grids (70×)	4^8	4^4	4^2	1
#Points (70×)	4^8	$4^4 \times 50$	$4^2 \times 100$	100



(a) Disturbance locations distributed in the regions of Hayman Fire Event at the highest resolution level.

(b) Examples of FPAR time series for the Hayman Fire Event at 39.45°N and 105.00°S and the anomaly score obtained by moving average (First month = 6/2002, Count = 12).

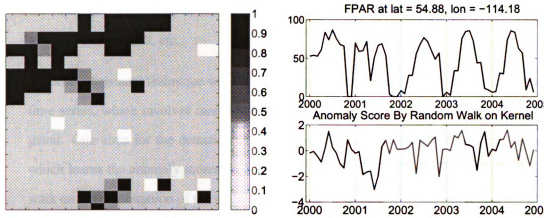
Figure 5.17. Hayman Fire Event.

method. This was further reduced to as few as 10 seconds at the lowest resolution level 3, which makes exploratory analysis feasible.

The preceding table shows the amount of time needed to process all the time series at each level. However, during exploratory data analysis, a user selects only one grid box to process as he/she drills down from one level to another. The run time needed to compute disturbance events for each grid box is shown in Table 5.5. Since all the grid points needed to be scanned at the lowest resolution when the explorer first appears, the time consumed is equal to scan all the data points at level 3, which is about 10 seconds for moving average method. When a region at level 3 is selected, the sample time series for that region will be computed for disturbance event detection. The response time at this level takes only about 1.4 seconds to compute. In turn, it takes another 0.59 seconds to drill down from level 2 to level 1 and 0.3 from level 1 to the highest resolution level 0. The same effect is also observed with the random walk method.

Table 5.5. Response Time For Drilling Down From Low Resolution To High Resolution.

	1-0	2-1	3-2	3
Avg Time(MV)	0.3	0.59	1.4	10.9
Avg Time(RW)	3.24	10.04	18.1	56.0
#Points	4^4	$4^2 \times 50$	$4^2 \times 100$	70×100



(a) Disturbance locations distributed in the regions of Chisholm Fire Event at the highest resolution level.

(b) Examples of FPAR time series for the Chisholm Fire Event at 54.88°N and 114.18°S and its anomaly score

Figure 5.18. Chisholm Fire Event.

5.7 Summary

Anomaly detection in time series data has been studied extensively in the literature. However, there are still many challenging problems for existing methods especially when applying to Earth Science data, which consists of a large number of multivariate time series in high spatial resolution. In this chapter, we discussed various techniques to detect and characterize different types of anomalies in multivariate time series data and developed a visualization system for exploratory analysis of ecosystem disturbances in Earth Science data. The main contributions of this work are:

- A graph-based method was first proposed for anomaly detection in multivariate time series data, which learns the connectivity score for each node by running a random walk on the graph constructed from the kernel matrix of the time series data. This method was further extended to detect different types of anomalies in time series data. First, local anomalies such as time-based or cycle-based anomalies can be detected by sparsifying the full kernel matrix to local kernel matrix, which consists of similarity values between data points in time-based or cycle-based local neighborhood only. Second, subsequence anomalies can also be detected by constructing kernel matrix between subsequences of the time series. Experimental results on both real FPAR time series and syntactic data sets demonstrated

the superior performance of this technique compared with other methods such as moving average, LOF, K-dist, etc.

- A kernel alignment technique was further discussed for anomaly discovery in multivariate time series, which involves more than one related variables. Two problems were investigated. One aims for the detection of anomalies in general noisy multivariate time series, which learns the anomaly scores for every data point or subsequence by running a random walk on the combination of the aligned kernel matrix among different variables. The alignment procedure will reduce the impact of noise significantly and thus decrease the false alarm rate. The effectiveness of this technique is demonstrated by experiments on a number of data sets. Another problem aims for the detection of target specific anomalies in multivariate time series, which learns the anomaly scores for every data point or subsequence in target time series and predictor time series by running a random walk on the their aligned kernel matrices. Anomalies in predictor time series that are most related to corresponding anomalies in target time series are detected and thus used to characterize the anomalies in target time series. This method is successfully applied to associate anomalies in climate events such as low precipitation with disturbance events in FPAR time series such as forest fires.

- Despite the success of our proposed graph-based technique for anomalies detection in time series data, it is very expensive to apply to large amounts of Earth Science time series in high spatial resolution, not to mention the requirement of tuning thresholds or parameters frequently in real applications. A clustering-based framework is proposed to assist the exploratory analysis of the FPAR time series repository for ecosystem disturbance detection. The clustering approach was first applied to cluster regions with similar ecosystem disturbance events to help the user in categorizing different types of disturbance events. Then it was used to build a multi-level indexing system, which is integrated into an interactive viewer that enables scientists to explore the data in near real-time fashion. The effectiveness and efficiency of our system are demonstrated by our evaluation.

CHAPTER 6

Conclusions and Future Work

The primary objective of the work presented in this dissertation is to develop new spatial and temporal data mining techniques to address the challenging problems in mining Earth Science data.

6.1 Contributions

We focused on three interesting but challenging research problems related to spatial and temporal data mining: (1) localized prediction in the presence of spatial and temporal dependencies; (2) long term time series forecasting; and (3) detection, characterization, and visualization of anomalies. Preliminary results of this thesis have appeared in several conference proceedings [46][47][45][48][42][44]. In this section, we summarize the main contributions of this thesis.

The localized prediction work described in Chapter 3 has made the following contributions:

- **Integrated Localized Support Vector Machine Framework**:

  An integrated framework for localized Support Vector Machine (including LSVC and LSVR) is proposed, which incorporates neighborhood information between training and testing examples into the SVM (including SVC and SVR) objective function. We tested the proposed framework on a number of real-world data sets and showed its superior performance over both KNN and nonlinear global SVM.

- **Supervised Clustering Algorithms and Profile Support Vector Machine**:

LSVM is expensive to compute since it requires training a separate SVM model for each test example. To address this limitation, we developed a more efficient implementation of the algorithm called Profile Support Vector Machine. PSVM reduces the computational time by extracting a small number of clusters using supervised algorithms called MagKmeans and VarKmeans, for classification and regression respectively. Our analysis showed that PSVM achieved comparable accuracy as LSVM but is much more computationally efficient.

- **LSVM and PSVM with Spatial and Temporal Dependencies**:

  We demonstrated the effectiveness of the proposed algorithms on data with spatial only, temporal only, or both spatio-temporal dependencies. The spatial, temporal or spatio-temporal neighborhood information is incorporated into the LSVM and PSVM framework. Our algorithms achieved significant improvement when applied to a number of prediction tasks on Earth Science data sets with spatial and temporal dependencies.

The long term time series forecasting work described in Chapter 4 has made the following contributions:

- **Empirical Study on Multi-step Time Series Prediction**:

  An empirical study is conducted on three prediction approaches for multi-step ahead time series prediction. Our theoretical and empirical analysis showed that current multi-stage prediction tends to suffer from error accumulation problem when the prediction period is long. Independent value prediction is less susceptible to this problem because its predictions are made independently at each time step. However, it has difficulty in learning the true model and filtering the effect of noise. Parameter prediction handles noisy data by fitting a function over the entire output sequence and also tends to be more efficient than independent value prediction when the number of parameters to be fitted is small. However, finding the appropriate parameter function to fit the time series is quite a challenging task.

- **Semi-supervised Hidden Markov Regression Model for Multivariate Time Series Prediction**:

  To alleviate the error accumulation problem in long term time series forecasting, an alternative strategy is adapted to perform multivariate time series prediction by utilizing informa-

tion from other related time series. A combination of local prediction and global prediction methods is developed to estimate the response values for the future data. A semi-supervised time series prediction approach is developed based on Hidden Markov Model Regression (HMMR) to learn from both historical and future data. Experimental results on a number of benchmark data sets shows the superior performance of our proposed semi-supervised algorithm compared with supervised algorithms.

- **Covariance Alignment for Data Calibration**:

  We showed that the inconsistencies between historical and future data may actually hurt the model's performance when we apply our semi-supervised HMMR for long term climate prediction. To compensate for this problem, we proposed a covariance alignment data calibration method that will transform the training and test data into a new space before applying the semi-supervised HMMR algorithm. Experimental results on the climate data sets clearly demonstrate the improvement of the semi-supervised HMMR algorithms when the proposed data calibration technique is used.

The anomaly detection, characterization, and visualization techniques described in Chapter 5 have made the following contributions:

- **Graph-based Time Series Anomaly Detection Algorithm and Its Extensions**:

  A robust graph-based method is proposed for anomaly detection in multivariate time series data, which performs better than other methods such as moving average, LOF, K-dist, etc. Our algorithm is also very flexible in that it can be extended to detect unusual subsequences or local anomalies. Our method was successfully applied to detect ecosystem disturbance in Earth Science data.

- **Kernel Alignment for Multivariate Time Series Anomaly Detection and Characterization**:

  A kernel alignment method is proposed to improve the anomaly detection in noisy multivariate time series data. This method can also be utilized to associate the anomalies in predictor time series with anomalies in target time series, e.g. to characterize the ecosystem disturbance such as forest fires detected from FPAR time series with climate anomalies such as

dry weather detected from precipitation time series. Extensive experiments have been conducted on synthetic and real Earth Science data sets to demonstrate the effectiveness of our algorithm.

- **Visualization System for Exploratory Analysis of Global Ecosystem Disturbances**:

  We also developed an interactive viewer that enables scientists to display the FPAR time series at locations where disturbance events have been detected. A clustering-based technique is adapted to group together regions with similar disturbance events to be presented to the user. The clustering method is further used to build a multi-level indexing structure of the large scale high resolution FPAR time series data such that the user can explore the data in near real-time fashion. The effectiveness and efficiency of the system are demonstrated by our evaluation.

In addition to the Earth Science domain, the algorithms and techniques developed in this thesis would be beneficial for many other real-world applications including traffic analysis, network monitoring, geographical information system, and stock market analysis, etc.

6.2 Future Research

In this section, we discuss some of the future research directions on spatial and temporal data mining motivated by this thesis.

- **Prediction of Extreme Events**:

  Most time series prediction predicts smooth future values first by constructing a regression function and then adds possible noise. The forecasts will exhibit less variability than the actual values, which implies a regression to the mean problem [23]. In many applications, people are interested in predicting extreme events. For example, Earth scientists are interested in knowing the distribution of droughts in the future which may have substantial impact on the ecosystem and agriculture of the affected region. Prediction of such extreme events is much more difficult compared with traditional time series prediction problems. First, although there has been some work on extreme value theory [74], modeling extreme events is still a challenging task, not mention to predict their future occurrences. Second,

extreme events usually happen more rarely than normal events and thus require much more data for model training. Third, because of pattern evolvement in long term time series such as climate change, the behavior of extreme events may change substantially compared with the history. Extending current methods or developing new techniques for extreme events forecasting is a very interesting problem to be addressed in future work.

- **Pattern Discovery in Spatio-temporal Database**:

  It this thesis, we have examined the detection and characterization of anomalous patterns in time series data. However, the method we proposed focused on temporal patterns only. There are still many interesting pattern discovery problems remaining for future research in Earth Science data. For instance, to detect anomalies in spatio-temporal data or to discover spatio-temporal associations between anomaly patterns in spatio-temporal data. Mining other interesting patterns such as frequent patterns or periodical patterns and their spatio-temporal association is also an interesting and challenging problem for future research. Developing new techniques to solve these problems for Earth Science applications would definitely help people better understand the interaction between the global climate, ecological, and economic systems underlying different observed phenomenons.

- **Distributed and Parallel Data Mining**:

  In this thesis, we have proposed a multi-level indexing system by sampling from high resolution spatio-temporal Earth Science data for exploratory analysis of the ecosystem disturbances. One limitation of our visualization system for mining ecosystem disturbance in large scale spatio-temporal data is that only parts of the data are processed each time. It was able to facilitate the task of exploratory analysis but did not solve the scalability issue directly when we need to process a very large data set. Recently, large scale computing systems such as cloud computing have been developed, which utilizes a large number of machines to solve a problem efficiently. Thus another interesting research direction that may worth future exploration is to develop parallelized algorithms in large scale distributed computing system such as map-reduce framework [65]. For example, to develop the parallelized version of the graph-based anomaly detection algorithm for ecosystem disturbance detection in large scale Earth Science data.

- **Learning from Heterogeneous Data**:

  As discussed in Chapter 4, data in Earth Science may come from different sources such as climate observations and global climate simulation. Heterogeneous data sets are usually generated from different distributions, which may degrade the performance of most standard supervised or semi-supervised learning algorithms. We have proposed a covariance alignment method in our work to calibrate the data from different sources. However, our solution only aligns the first and second order of the data. Higher order alignment of the data using kernel statistics would be a possible research direction. Another interesting research problem would be to apply transfer learning methodology [217], which learns from small amounts of training data and large amounts of low quality auxiliary data with different distribution. However, transfer learning typically requires the auxiliary data with different data distribution to be labeled, which is not true for the future data in time series forecasting problem. Recently, Raina et al. [175] and Ando et al. [8] proposed transfer learning framework to learn from unlabeled data, which may serve as the foundation for our future work.

- **Multi-scale Anomaly Detection**:

  The proposed anomaly detection techniques in Chapter 5 only consider anomalies at the same spatial and temporal scales. In the Earth Science domain, ecosystem disturbance events may happen at different spatial or temporal scales. For example, ecosystem disturbance events such as wildfires may happen at a large spatial scale whereas other events caused by human activities such as construction of a housing subdivision or a golf course may happen at a small spatial scale; some disturbance events may last several months whereas others may last only a short period of time. Detecting anomalies at different spatial and temporal scales may help reveal the intrinsic spatial and temporal structure of the anomaly patterns and can be used to characterize different types of anomalies.

APPENDIX A

Proof of the Dual Form for Localized Support Vector Machine

Let $\mathcal{D}_U = [\boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, ..., \boldsymbol{x}_{l+u}]^\top$ be a set of u test examples and $\boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})$ be the similarity between the training example $\boldsymbol{x}_i$ and the test example $\boldsymbol{x}_{l+s}$. For each $\boldsymbol{x}_{l+s} \in \mathcal{D}_U$, we construct its primal local SVM model by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^{l} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})\xi_i \\ \text{s. t} \quad & y_i(< \mathbf{w}, \varphi(\boldsymbol{x}_i) > -b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, 2, \ldots, l \end{aligned} \tag{A.1}$$

We introduce the Lagrangian multiplier α_i for each inequality condition $\mathbf{y}_i(< \mathbf{w}, \varphi(\boldsymbol{x}_i) > -b) \geq 1 - \xi_i$, and $\mathbf{u}_i$ for $\xi_i \geq 0$. As a result, the Lagrangian for the optimization problem can be written as:

$$\begin{aligned} \mathcal{L} \quad = \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^{l} \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s})\xi_i \\ & - \sum_{i=1}^{l} \alpha_i(\mathbf{y}_i(< \mathbf{w}, \varphi(\boldsymbol{x}_i) > -b) - 1 + \xi_i) - \sum_{i=1}^{l} \mathbf{u}_i \xi_i \end{aligned} \tag{A.2}$$

Then, taking deritives of $\mathcal{L}$ with regard to $\mathbf{w}$, $\mathbf{b}$, ξ_i, we obtain the KKT conditions for the primal problem:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i \mathbf{y}_i \tag{A.3}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = 0 \Rightarrow -\sum_{i=1}^{l} \alpha_i \mathbf{y}_i = 0 \tag{A.4}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow \beta \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s}) - \alpha_i - \mathbf{u}_i = 0 \tag{A.5}$$

$$\mathbf{y}_i(< \mathbf{w}, \varphi(\boldsymbol{x}_i) > -b) - 1 + \xi_i \geq 0 \tag{A.6}$$

$$\xi_i \geq 0 \tag{A.7}$$

$$\alpha_i \geq 0 \tag{A.8}$$

$$\mathbf{u}_i \geq 0 \tag{A.9}$$

$$\alpha_i\{\mathbf{y}_i(< \mathbf{w}, \varphi(\boldsymbol{x}_i) > -b) - 1 + \xi_i\} = 0 \tag{A.10}$$

$$\mathbf{u}_i \xi_i = 0 \tag{A.11}$$

From the last two KKT complementarity conditions,we can determine the threshold b. Furthermore, if $\alpha_i < \beta \boldsymbol{S}(\boldsymbol{x}_i, x)$, then $\xi_i = 0$. So any training point in which $0 < \alpha_i < \beta \boldsymbol{S}(\boldsymbol{x}_i, x)$ can be used to compute b. By replacing Equations (A.3)−(A.5) into the objective function in (A.2), the optimization problem can be reduced to:

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j + \sum_{i=1}^{l}(\alpha_i + \mathbf{u}_i)\xi_i - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \\ &\quad - \sum_{i=1}^{l} \alpha_i(-b\mathbf{y}_i - 1 + \xi_i) - \sum_{i=1}^{l} \mathbf{u}_i \xi_i \\ &= -\frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j + \sum_{i=1}^{l} \alpha_i \end{aligned} \tag{A.12}$$

Thus, we obtain the dual form of the LSVM optimization problem, which is given as follows:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \\ \text{s. t.} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta \boldsymbol{S}(\boldsymbol{x}_i, \boldsymbol{x}_{l+s}),\ i = 1, 2, \ldots, l \end{aligned}$$

BIBLIOGRAPHY

[1] Hayman fire, baer report. http://www.wilderness.org/Library/Documents/WildfireSummary_Hayman.cfm, 2002.

[2] J. Adhikary. Knowledge discovery in spatial databases: Progress and challenges. Technical report, University of British, 1996.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, Santiago de Chile, Chile, 1994.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995.

[5] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21(1):243–247, 1969.

[6] E. L. Allwein, R. E. Schapire, Y. Singer, and P. Kaelbling. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[7] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 375–381, Madison, Wisconsin, USA, 2003.

[8] R. K. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 1–9, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[9] L. Anselin. Local indicators of spatial association-lisa. *Geographical Analysis*, 27:93–115, 1995.

[10] M. Atallah, R. Gwadera, and W. Szpankowski. Detection of significant sets of episodes in event sequences. In *Proceedings of the 4th IEEE International Conference on Data Mining*, volume 00, pages 3–10, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[11] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, April 1997.

[12] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. *Fast Kernel Learning using Sequential Minimal Optimization*. EECS Department, University of California, Berkeley, 2004.

[13] R. A. Baeza-Yates. Searching subsequences. *Theorem of Computer Science*, 78(2):363–376, 1991.

[14] P. M. Baggenstos. A modified baum-welch algorithm for hidden markov models with multiple observation spaces. *IEEE Transactions on Speech Audio Processing*, 2:411–416, 2001.

[15] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden markov models of biological primary sequence information. In *Proceedings of the National Academy of Science*, volume 91, pages 1059–1063, feb 1994.

[16] S. Bay, K. Saito, N. Ueda, and P. Langley. A framework for discovering anomalous regimes in multivariate time-series data with local models. In *Symposium on Machine Learning for Anomaly Detection*, 2004.

[17] B. Belhouari, S. Bermak, and Amine. Gaussian process for nonstationary time series prediction. *Computational Statistics and Data Analysis*, 47(41):705–712, 2004.

[18] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[19] M. Bertolotto, T. Kechadi, S. Di Martino, and F. Ferrucci. Scalable 2-pass data mining technique for large scale spatio-temporal datasets. In *Proceedings of the 11th Knowledge based and Intelligent Information and Engineering Systems*, pages 785–792, Vietri sul Mare, Italy, 2007.

[20] S. K. Bethi, V. V. Phoha, and Y. B. Reddy. Clique clustering approach to detect denial-of-service attacks. In *Proceedings of the 5th Annual IEEE SMC Workshop on Information Assurance*, pages 447–448, June 2004.

[21] D. Birant and A. Kut. Spatio-temporal outlier detection in large databases. *Computing and Information Technology*, 14(4):291–298, 2006.

[22] J. A. Blackard and J. D. Denis. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24:131–151, 2000.

[23] J. M. Bland and D. G. Altman. Statistic notes: Regression towards the mean. *British Medical Journal*, 308:1522, 1994.

[24] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings 18th International Conference on Machine Learning*, pages 19–26, San Francisco, CA, USA, 2001. Morgan Kaufmann.

[25] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

[26] Y. L. Borgne. Bias variance trade-off characterization in a classification. what differences with regression? Technical report, University Libre de Bruxelles, 2005.

[27] Y. L. Borgne, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Process*, 87(12):3010–3020, 2007.

[28] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[29] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis, Forecasting, and Control.* Prentice Hall, 1970.

[30] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *Proceedings of the 23rd international conference on Machine learning*, pages 137–144, Pittsburgh, Pennsylvania, 2006. ACM Press.

[31] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 19th ACM SIGMOD international conference on Management of Data*, pages 93–104, Dallas, Texas, USA, 2000.

[32] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2:121–167, 1998.

[33] J. Caiado and N. Crato. A garch-based method for clustering of financial time series: International stock markets evidence. MPRA Paper 2074, University Library of Munich, Germany, 2007.

[34] E. Camossi, M. Bertolotto, and T. Kechadi. Mining spatio-temporal data at different levels of detail. In *Lecture Notes in Geoinformation and Cartography*, pages 225–240. Springer Berlin Heidelberg, 2008.

[35] CCCSN. Canadian climate change scenarios network, environment canada. `http://www.ccsn.ca/`.

[36] G. Celeux and J. B. Durand. Selecting hidden markov model state number with cross-validated likelihood. *Computational Statistics*, 2007.

[37] C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[38] C. Y. Chang, M. S. Chen, and C. H. Lee. Mining general temporal association rules for items with different exhibition periods. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, volume 00, page 59, Los Alamitos, California, USA, 2002. IEEE Computer Society.

[39] N. Chapados and Y. Bengio. Augmented functional time series representation and forecasting with gaussian processes. 2007.

[40] S. P. Charles, B.C. Bates, I.N. Smith, and J. P. Hughes. Statistical downscaling of daily precipitation from observed and modelled atmospheric fields. *Hydrological Processes*, 18(8):1373–1394, 2004.

[41] C. Chatfield. *The analysis of time series.* New York, NY: Chapman and Hall, 1996.

[42] H. F. Chen, H. B. Cheng, G. F. Jiang, and K. J. Yoshihira. Exploiting local and global invariants for the management of large scale information systems. In *Proceedings of the 8th IEEE International Conference on Data Mining*, Pisa, Italy, 2008.

[43] M. Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.

[44] H. B. Cheng and P. N. Tan. Semi-supervised learning with data calibration for long-term time series forecasting. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–141, Las Vegas, NV, 2008.

[45] H. B. Cheng, P. N. Tan, J. Gao, and J. Scripps. Multistep-ahead time series prediction. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774, Singapore, 2006.

[46] H. B. Cheng, P. N. Tan, and R. Jin. Localized support vector machine and its efficient algorithm. In *Proceedings of the 7th SIAM Conference on Data Mining*, 2007.

[47] H. B. Cheng, P. N. Tan, C. Potter, and S. Klooster. Data mining for visual exploration and detection of ecosystem disturbaces. In *Proceedings of 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Irvine, CA, 2008.

[48] H. B. Cheng, P. N. Tan, C. Potter, and S. Klooster. A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series. In *Proceedings of the 8th IEEE International Conference on Data Mining Workshop on Spatial and Spatiotemporal Data Mining*, Irvine, CA, 2008.

[49] T. Cheng and Z. Li. A hybrid approach to detect spatial-temporal outliers. *GeoInformaticas*, pages 173–178, 2004.

[50] S. Chu, E. K., D. Hart, and Michael. Iterative deepening dynamic time warping for time series. In *Proceedings of the 2nd SIAM international conference on Data Mining*, 2002.

[51] I. Cohen, N. Sebe, F. G. Cozman, M. C. Cirelo, and T. S. Huang. Semi-supervised learning of classifiers: Theory and algorithms for bayesian network classifiers and applications to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1553–1566, Dec 2004.

[52] P. Compieta, S. Di Martino, M. Bertolotto, F. Ferrucci, and T. Kechadi. Exploratory spatio-temporal data mining and visualization. *Journal of Visual Languages and Computing*, 18(3):255–279, 2007.

[53] C. Cortes and M. Mohri. On transductive regression. In *Proeedings of the 22th Annual Conference on Neural Information Processing Systems*, British Columbia, Canada, 2006.

[54] P. Cortez and A. Morais. A data mining approach to predict forest fires using meteorological data. In *Proceedings of the 13th Portuguese Conference on Artificial Intelligence*, pages 512–523, Guimaraes, Portugal, 2007.

[55] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions in Information Theory*, 36:21–27, 1967.

[56] F. G. Cozman, I. Cohen, and M. Cirelo. Semi-supervised learning of mixture models. In *Proceedings of the 20th International Conference on Machine Learning*, pages 99–106, Washington, DC, USA, 2003.

[57] F.G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the 15th International Florida Artificial Intelligence Society Conference*, pages 327–331, Pensacola Beach, Florida, USA, 2002.

[58] N. Cressie. *Statistics for spatial data.* Wiley, New York, 1993.

[59] M. A. Crimmins. Synoptic climatology of extreme fire-weather conditions across the southwest united states. *International Journal of Climatology*, 26:1001–1016, 2006.

[60] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pages 367–373, British Columbia, CA, 2002.

[61] T. Darrell and A. Pentland. Space-time gestures. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 335–340, Jun 1993.

[62] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 88–100, London, UK, 1997. Springer-Verlag.

[63] D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the 5th International Conference on Intelligent Systems*, pages 82–87, Reno, Nevada, USA, 1996.

[64] M. D'Auria, M. Nanni, and D. Pedreschi. Time-focused density-based clustering of trajectories of moving objects. In *Proceedings of the Workshop on Mining Spatio-Temporal Data*, 2005.

[65] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, pages 137–150, San Francisco, CA, December 2004.

[66] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1997.

[67] Denny, G. J. Williams, and P. Christen1. Exploratory hot spot profile analysis using interactive visual drill-down self-organizing maps. In *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 536–543, Osaka, Japan, 2008. Springer Berlin Heidelberg.

[68] M. Dillon. Search for efficient local kriging algorithm. *The Statistician*, 41(3):383, 1992.

[69] C. Domeniconi, D. Gunopulos, and P. Jing. Large margin nearest neighbor classifiers. *IEEE Transactions on Neural Networks*, 16(4):899–909, 2005.

[70] T. Dusek. Regional income differences in hungary - a multi-level spatio-temporal analysis. ERSA conference papers ersa06p284, European Regional Science Association, 2006.

[71] T. S. Dye, C. P. MacDonald, and C. B. Anderson. Guideline for developing an ozone forecasting program. Technical report, EPA-454/R-99-009, 1999.

[72] D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, and R. Porter andJ. Theiler. Genetic algorithms and support vector machines for time series classification. In *Proceedings of the SPIE*, volume 4787, pages 74–85, 2002.

[73] J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–224, 1991.

[74] P. Embrechts, C. Kluppelberg, and T. Mikosch. *Modeling Extremal Events: for Insurance and Finance*. Spring Verlag, Berlin, German, 1997.

[75] W. Enke and A. Spekat. Downscaling climate model outputs into local and regional weather elements by classification and regression. *Climate Research 8*, 8(3):195–207, 1997.

[76] J. Ernst, G. J. Nau, and Z. Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(1):159–168, 2005.

[77] A. K. Ersbøll and B. K. Ersbøll. On spatio-temporal kriging. In *Proceedings of the 3rd Annual Conference of the International Association for Methematica Geology*, pages 617–622, Barcelona, Spain, 1997. CIMNE.

[78] M. Ester. Data mining tasks and methods: spatial analysis. *Handbook of data mining and knowledge discovery*, pages 409–418, 2002.

[79] M. Ester, H. P. Kriegel, J. Sander, and X. W. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, USA, 1996.

[80] M. Ester, H.P. Kriegel, and J. Sander. Algorithms and applications for spatial data mining. *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, 2001.

[81] Martin Ester, Hans peter Kriegel, and Xiaowei Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In *Proceedings of 4th International Symposium on Large Spatial Databases*, pages 67–82, Portland, Maine, 1995. Springer.

[82] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[83] A. S. Fotheringham, C. Brunsdon, and M. E. Charlton. *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. Chichester: Wiley, 2002.

[84] H. Frohlich and A. Zell. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. *Proceedings of IEEE International Joint Conference on Neural Networks*, 3:1431–1436, July 2005.

[85] K. Fujinaga, M. Nakai, H. Shimodaira, and S. Sagayama. Multiple-regression hidden markov model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 513–516, Salt Lake City, Utah, USA, 2001.

[86] S. J. Gaffney, A. W. Robertson, P. Smyth, S. J. Camargo, and M. Ghil. Probabilistic clustering of extratropical cyclones using regression mixture models. *Climate Dynamics*, 29:423–440, 2007.

[87] P. Galeano, D. Pena, and R. S. Tsay. Outlier detection in multivariate time series via projection pursuit. Statistics and Econometrics Working Papers ws044211, Universidad Carlos III, Departamento de Estadłstica y Econometrła, 2004.

[88] M. Garofalakis, R. Rastogi, and K. Shim. Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):530–552, 2002.

[89] N. Gilardi and S. Bengio. Local machine learning models for spatial data analysis. *Geographic Information and Decision Analysis*, 4(1):11–28, 2000.

[90] C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy time series prediction using a recurrent neural network and grammatical inference. *Machine Learning*, 44(1-2):161–183, 2001.

[91] C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy time series prediction using a recurrent neural network and grammatical inference. *Machine Learning*, 44(1/2):161–183, July/August 2001.

[92] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[93] J. G. Goldammer. Global fire monitoring center, an overview on forest fires in cuba. `http://www.fire.uni-freiburg.de/iffn/country/cu/cu_1.htm#top`.

[94] T. H. Grubesic and A. T. Murray. Detecting hot spots using cluster analysis and gis. In *Proceedings of the 5th Annual Conference of the Crime Mapping Research Center*, Dallas, TX, 2001.

[95] R. Haining. *Spatial Data Analysis in the Social and Environmental Sciences*. Cambridge University Press, 1993.

[96] R. Haining. *Spatial Data Analysis: Theory and Practice*. Cambridge University Press, 2003.

[97] D. J. Hand and V. Vinciotti. Choosing k for two-class nearest neighbor classifiers with unbalanced classes. *Pattern Recognition Letters*, 24(9-10):1555–1562, 2003.

[98] J. Haslett, R. Bradley, P. Craig, A. Unwin, and G. Wills. Dynamic graphics for exploring spatial data with application to locating global and local anomalies. *The American Statistician*, 45(3):234–242, 1991.

[99] T. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statistical Science*, 8(2):120–143, 1993.

[100] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

[101] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference and prediction*. New York: Springer-Verlag, 2001.

[102] K. Hechenbichler and K. P. Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. Technical report, Ludwig-Maximilians University, Munich, 2006.

[103] T. Hengl, G. B. M. Heuvelink, and D. G. Rossiter. About regression-kriging: From equations to case studies. *Computers and Geosciences*, 33:1301–1315, Oct 2007.

[104] L. J. Herrera, H. Pomares, I. Rojas, A. Guillén, A. Prieto, and O. Valenzuela. Recursive prediction for long term time series forecasting using advanced models. *Neurocomputinig*, 70(16-18):2870–2880, 2007.

[105] B. C. Hewitson and R. G. Crane. Climate downscaling techniques and applications. *Climate Research*, 7:85–95, 1996.

[106] F. M. Hoffman, W. W. Hargrove, and A. D. Del Genio. Multivariate spatio-temporal clustering of time-series data: An approach for diagnosing cloud processes and understanding arm site representativeness. In *Proceedings of the 13th Atmospheric Radiation Measurement (ARM) Science Team Meeting, Extended Abstracts*, Broomsfield, Colorado, USA, 2003.

[107] W.C. Hong, P. F. Pai, S. L. Yang, and R. Theng. Highway traffic forecasting by support vector regression model with tabu search algorithms. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1617–1621, Vancouver, British Columbia, Canada, 2006.

[108] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2001.

[109] W. Hsu, J. Dai, and M. L. Lee. Mining viewpoint patterns in image databases. In *Proceeding of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 553–558, Washington, DC, USA, 2003.

[110] R. Hyndman. Time series data library. `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/`.

[111] Z. Ji and D. Dasgupta. Applicability issues of the real-valued negative selection algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 111–118, New York, NY, USA, 2006. ACM.

[112] T. Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[113] P. D. Jones and A. Moberg. Hemispheric and large-scale surface air temperature variations: An extensive revision and an update to 2001. *Journal of Climate*, 16:206–223, 2003.

[114] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[115] G. Karypis, E. H. Han, and V. Kumar. Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, Aug 1999.

[116] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. New York: Wiley, 1990.

[117] B. Kazar, S. Shekhar, and D. J. Lilja. Parallel formulation of spatial auto-regression. Technical report, Army High-Performance Computing Research Center, 2003.

[118] B. Kedem and K. Fokianos. *Regression Models for Time Series Analysis*. Wiley-Interscience, 2002.

[119] M. B. Kenneland, R. Brown, and H. D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A (Atomic, Molecular, and Optical Physics)*, 45(6):3403–3411, 1992.

[120] E. Keogh and T. Folias. Ucr time series data mining archive. `http://www.cs.ucr.edu/~eamonn/TSDMA/index.html`, 2002.

[121] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 226–233, Houston, Texas, USA, 2005.

[122] E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the 8th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 550–556, Edmonton, Alberta, Canada, 2002.

[123] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1–11, San Diego, California, USA, 1999. Springer.

[124] R. A. Kerr. Climate prediction: Signs of success in forecasting elnino. *Science*, 26:497, 2002.

[125] S. Kiritchenko, S. Matwin, and S. Abu-Hakima. Email classification with temporal features. *Proceedings of Intelligent Information Systems, New Trends in Intelligent Information Processing and Web Mining*, pages 523–534, 2004.

[126] K. Koperski, J. Adhikary, and J. W. Han. Spatial data mining: Progress and challenges. In *Proceedings of the SIGMOD Workshop on Research Issues on data Mining and Knowledge Discovery*, pages 1–10, 1996.

[127] K. Koperski and J. W. Hah. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 47–66, Portland, Maine, 1995. Springer-Verlag.

[128] K. Koperski, J. Han, and N. Stefanovic. An efficient two-step method for classification of spatial data. In *Proceedings of the 8th International Symposium on Spatial Data Handling*, pages 45–54, Vancouver, Canada, 1998.

[129] M. Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.

[130] J.K. Kumar. An application of spatial prediction using a fuzzy-neural network. *Proceedings of the International Joint Conference on Neural Networks*, 6:4241–4246, Jul 1999.

[131] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 201–206, Taormina, Sicily, Italy, 2004. ACM.

[132] K. W. Lau and Q. H. Wu. Local prediction of non-linear time series using support vector regression. *Pattern Recognition*, 41(5):1556–1564, 2008.

[133] M. H. Law and J. T. Kwok. Rival penalized competitive learning for model-based sequence clustering. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 02, page 2195, Los Alamitos, California, USA, 2000. IEEE Computer Society.

[134] M. B. Lawrence. Tropical cyclone report - tropical storm lorenzo,national hurricane center. `http://www.nhc.noaa.gov/2001iris.html`, 2001.

[135] S. Laxman and P. S. Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.

[136] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419, San Jose, California, USA, 2007. ACM.

[137] S. T. Li, S. W. Chou, and J. J. Pan. Multi-resolution spatio-temporal data mining for the study of air pollutant regionalization. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, page 7, Hawaii, USA, 2000.

[138] B. Liu and J. Liu. Multivariate time series prediction via temporal classification. In *Proceedings of the 18th International Conference on Data Engineering*, page 268, San Jose, CA, USA, 2002.

[139] C. T. Lu, D. Chen, and Y. Kou. Algorithms for spatial outlier detection. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, Nov. 2003.

[140] C. T. Lu and L. R. Liang. Wavelet fuzzy classification for detecting and tracking region outliers in meteorological data. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 258–265, New York, NY, USA, 2004. ACM.

[141] M. Mahoney and P. Chan. Trajectory boundary modeling of time series for anomaly detection. In *Proceedings of the KDD Workshop on Data Mining Methods for Anomaly Detection*, pages 32–40, 2005.

[142] D. Malerba, F. Esposito, and F. A. Lisi. Mining spatial association rules in census data: a relational approach. In *Proceedings of the ECML/PKDD workshop on mining official data*, pages 80–93. University Printing House, 2002.

[143] N. Mamoulis, H. P. Cao, G. Kollios, M. Hadjieleftheriou, Y. F. Tao, and D. W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245, Seattle, WA, USA, 2004. ACM.

[144] H. Mannila, , H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[145] F. Melgani and S. B. Serpico. A markov random field approach to spatio-temporal contextual image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 41(11):2478–2487, 2003.

[146] J. Mennis and J. W. Liu. Mining association rules in spatio-temporal data: An analysis of urban socioeconomic and land cover change. *Transactions in GIS*, pages 5–17, 2005.

[147] C. T. Mills. *Time Series Techniques for Economists*. Cambridge University Press, 1990.

[148] H. D. K. Moonesignhe and P. N. Tan. Outlier detection using random walks. In *Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 532–539, Washington, DC, USA, January 2006.

[149] Y. Morimoto. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–358, San Francisco, California, 2001. ACM.

[150] R. Munro, S. Chawla, and R. Sun. Complex spatial relationships. *Proceedings of the 3rd IEEE International Conference on Data Mining*, 3:227, 2003.

[151] V. S. Nalwa. Automatic on-line signature verification. In *Proceedings of the 3rd Asian Conference on Computer Vision*, volume I, pages 10–15, London, UK, 1997. Springer-Verlag.

[152] M. Nanni. Clustering methods for spatio-temporal data. Technical report, PhD thesis, CS Department, University of Pisa, 2002.

[153] M. A. Nascimento and Jefferson R. O. Silva. Towards historical r-trees. In *Proceedings of the ACM symposium on Applied Computing*, pages 235–240, Atlanta, Georgia, USA, 1998.

[154] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, 1998.

[155] R. T. Ng and J. W. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, Santiago de Chile, Chile, 1994.

[156] V. Niennattrakul and C. A.Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering*, pages 733–738, April 2007.

[157] K. R Nuller, A. J. Smola, G. Ratsch, B. Scholkopf, J. kohlmorgen, and V. Vapnik. Predicting time series with support vector machine. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, volume 1327, pages 999–1004, Lausanne, Switzerland, 1997. Springer LNCS.

[158] A. Ober-Sundermeier and H. Zackor. Prediction of congestion due to road works on freeways. In *Proceedings of the IEEE Intelligent Transportation Systems*, pages 240–244, Oakland, California, USA, 2001.

[159] S. Olga and D. Liu. Visual interactive clustering and querying of spatio-temporal data. In *Proceedings of the International conference on computational science and its applications*, page 1362, Perugia, Italy, 2005.

[160] S. Ozden and R. A. Silberschatz. Cyclic association rules. In *Proceedings of 14th International Conference on Data Engineering*, pages 412–421, Orlando, Florida, USA, 1998. IEEE Computer Society.

[161] S. L. Ozesmia and U. OZesmib. An artificial neural network approach to spatial habitat modelling with interspecific interaction. *Ecological Modeling*, 116(1):15–31, March 1999.

[162] R. K. Pace, R. Barry, J. M. Clapp, and M. Rodriquez. Spatiotemporal autoregressive models of neighborhood effects. *Journal of Real Estate Finance and Economics*, 17(1):15–33, July 1998.

[163] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[164] D. Papadias, Y. F. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *Proceedings of the 18th International Conference on Data Engineering*, page 166, Washington, DC, USA, 2002. IEEE Computer Society.

[165] M. Pawlak and M. F. Yat Fung Ng. On kernel and radial basis function techniques for classification and function recovering. In *Proceedings of the 12th International Conference on Pattern Recognition*, volume 2, pages 454–456, Jerusalem, Israel, 1994.

[166] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, 39(1):547–553, 2000.

[167] D. Pokrajac and Z. Obradovic. Combining regressive and auto-regressive models for spatio-temporal prediction. In *Proceedings of 17th International Machine Learning Workshop on Spatial Knowledge*, 2000.

[168] D. Pokrajac and Z. Obradovic. Improved spatial-temporal forecasting through mining. In *Proceedings of the 1st SIAM International Conference on Data Mining (SDM)*, Chicago, USA, 2001.

[169] T. D. Popescu. Multivariate time series forecasting using independent component analysis. In *Proceedings of 9th IEEE Conference on Emerging Technologies and Factory Automation*, pages 782–789, Lisbon, Portugal, 2003.

[170] C. Potter, P. N. Tan, M. Steinbach, S. Klooster, V. Kumar, R. Myneni, and V. Genovese. Major disturbance events in terrestrial ecosystems detected using global satellite data sets. *Global Change Biology*, 9(7):1005–1021, 2003.

[171] C. Potter, P. N. Tan, M. Steinbach, S. Klooster, V. Kumar, R. Myneni, and V. Genovese. Major disturbance events in terrestrial ecosystems detected using global satellite data sets. *Global Change Biology*, 9(7):1005–1021, 2003.

[172] C. S. Potter, P. N. Tan, V. Kumar, C. Kucharik, S. Klooster, V. Genovese, W. Cohen, and S. Healey. Recent history of large-scale ecosystem disturbances in north america derived from the avhrr satellite record. *Ecosystems*, 8:808–824, 2005.

[173] C. S. Potter, P. N. Tan, M. Steinbach, V. Kumar, S. Klooster, R. Myneni, and V. Genovese. Major disturbance events in terrestrial ecosystems detected using global satellite data sets. *Global Change Biology*, 9(7):1005–1021, 2003.

[174] L. Rabiner and B.H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1993.

[175] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th Annual International Conference on Machine Learning*, volume 227, pages 759–766. ACM, 2007.

[176] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Record*, 29(2):427–438, 2000.

[177] C. E. Rasmussen. K nearest neighbors for regression knn-cv-1, 1996.

[178] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[179] I. Rish. An empirical study of the naive bayes classifier. In *Proceedings of the 17th International Joint Conferences on Artificial Intelligence Workshop on Empirical Methods in Artificial Intelligence*, pages 41–48, Seattle, Washington, USA, 2001.

[180] P. P. Rodrigues, J. Gama, and J. P. Pedroso. Hierarchical clustering of time-series data streams. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):615–627, May 2008.

[181] R. A. Rohde. Instrumental temperature record. `http://www.globalwarmingart.com/wiki/Image:Instrumental_Temperature_Record_png`.

[182] R. A. Rohde. Global warming predictions. http://www.globalwarmingart.com/wiki/Image:Global_Warming_Predictions_png, 2001.

[183] J. Sander, M. Ester, H. P. Kriegel, and X. W. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[184] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and sons, 1998.

[185] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis. Discovering dynamics using bayesian clustering. In *Proceedings of the 3rd International Symposium on Intelligent Data Analysis*, page pages. Springer, 1999.

[186] S. Shekhar and Y. Huang. Discovering spatial co-location patterns: A summary of results. *Advances in Spatial and Temporal Databases*, pages 236–256, 2001.

[187] S. Shekhar, C. T. Lu, and P. Zhang. A unified approach to spatial outliers detection. Technical report, Department of Computer Science and Engineering, University of Minnesota, 2003.

[188] S. Shekhar, C. T. Lu, and P. S. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, pages 371–376. Press, 2001.

[189] S. Shekhar, P. R. Schrater, R. R. Vatsavai, W. Wu, and S. Chawla. Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia*, 4:174–188, 2002.

[190] S. Shekhar, P.S. Zhang, Y. Huang, and R. Vatsavai. Trends in spatial data mining. *Data Mining: Next Generation Challenges and Future Directions*, 2003.

[191] A.J. Smola and B. Schoelkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 1998.

[192] T. Starner and A. Pentl. Visual recognition of american sign language using hidden markov models. In *Proceedings of the IEEE International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, 1995.

[193] M. Steinbach, P. N. Tan, V. Kumar, S. Klooster, and C. Potter. Discovery of climate indices using clustering. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–55, Washington, DC, USA, 2003.

[194] B.J. Stocks, J. A. Mason, J. B. Todd, E. M. Bosch, B. M. Wotton, B. D. Amiro, M. D. Flannigan, K. G. Hirsch, K. A. Logan, D. L. Martell, and W. R. Skinner. Large forest fires in canada. *Journal of Geophysical Research*, 108(D1, 8149):1959–1997, 2002.

[195] A. Subasi, A. Alkan, E. Koklukaya, and M. K. Kiymik. Wavelet neural network classification of eeg signals by using ar model with mle preprocessing. *Neural Network*, 18(7):985–997, 2005.

[196] Y. X. Sun, K. Ma, X. Jin, X. Pu, and W. Gao. Detecting spatio-temporal outliers in climate dataset: A method study. *International Geoscience And Remote Sensing Symposium*, 2:760, 2005.

[197] U.S. Geological Survey. Antelope fire, sioux falls, south dakota usa, august 2002. ftp://edcftp.cr.usgs.gov/pub/data/landcover/files/2002/wupa/ante02a_meta.pdf.

[198] Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, and Dimitris Papadias. Spatio-temporal aggregation using sketches. In *Proceedings of the International Conference on Data Engineering*, pages 214–226, Boston, USA, 2004.

[199] Q. Tian, J. Yu, Q. Xue, and N. Sebe. A new analysis of the value of unlabeled data in semi-supervised learning for image retrieval. In *Proceedings of IEEE International Conf. on Multimedia and Expo*, pages 1019– 1022, Taipei, Taiwan, 2004.

[200] Y. Tian, Y. Zhang, Y. Knyazikhin, R. B. Myneni, and S. W. Running. Prototyping of modis lai/fpar algorithm with lasur and landsat data. *IEEE Transaction on Geoscience and Remote Sensing*, 38(5):2387–2401, 2000.

[201] C. Tilke. Variowin - software for spatial data analysis in 2d. *Computational Statistics and Data Analysis*, 25(2):243–244, July 1997.

[202] J. Tilton. Image segmentation by region growing and spectral clustering with a natural convergence criterion. In *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, pages 1766–1768, Seattle, Washington,U SA, 1998.

[203] P. Tino, C. Schittenkopf, and G. Dorffner. Volatility trading via temporal pattern recognition in quantized financial time series. *Pattern Analysis and Applications*, 4(4):283–299, 2001.

[204] F. Verhein. k-stars: Sequences of spatio-temporal association rules. In *Proceedings of 6th IEEE International Conference on Data Mining*, pages 387–394, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[205] F. Verhein and S. Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In *Proceedings of the 11th International Conference on Database Systems for Advanced Applications,Springer Lecture Notes in Computer Science*, pages 187–201, 2006.

[206] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pages 985–992, 2002.

[207] N. Vivoy. Automatic classification of time series (acts): A new clustering method for remote sensing time series. *International Journal of Remote Sensing*, 45(3):191–200, 2000.

[208] J. Wang and J. Han. Bide: efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, pages 79–90, Boston, USA, 2004.

[209] J. Wang, W. Hsu, and M. L. Lee. Flowminer: finding flow patterns in spatio-temporal databases. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 14–21, Nov. 2004.

[210] M. Wang, X. S. Hua, Y Song, L. R. Dai, and H. J. Zhang. Semi-supervised kernel regression. In *Proceedings of the 6th International Conference on Data Mining*, pages 1130–1135, Washington, DC, USA, 2006.

[211] L. Wei and E. Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753, Philadelphia, PA, USA, 2006. ACM.

[212] L. Wei, N. Kumar, V. N. Lolla, E. Keogh, and S. Lonardi. Assumption-free anomaly detection in time series. In *Proceedings of the 17th International Scientific and Statistical Database Management Conference*, pages 237–240, Santa Barbara, CA, 2005.

[213] L. Wei, N. Kumar, V. N. Lolla, E. Keogh, and S. Lonardi. Outliers detection in multivariate time series by independent component analysis. *Neural Computation*, pages 1962–1984, 2007.

[214] R. L. Wilby, S. P. Charles, E. Zorita, B. Timbal, P. Whetton, and L. O. Mearns. Guidelines for use of climate scenarios developed from statistical downscaling methods. Available from the DDC of IPCC TGCIA, 2004.

[215] C. C. Wong, M. C. Chan, and C. C. Lam. Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization. Technical Report 61, Society for Computational Economics, 2000.

[216] C. Wu, M. Berry, S. Shivakumar, and J. McLarty. Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning*, 21(1-2):177–193, 1995.

[217] P. C. Wu and T. G. Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the 21th Annual International Conference on Machine Learning*, pages 871–878, Banff, Alberta, Canada, 2004.

[218] H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, and J. Yoo. A framework for discovering co-location patterns in data sets with extended spatial objects. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 80–93, Lake Buena Vista, Florida, USA, 2004.

[219] Y. M. Xiong and D. Y. Yeung. Mixtures of arma models for model-based time series clustering. In *Proceedings of the IEEE International Conference on Data Mining*, pages 717–720, Maebashi City, Japan, 2002.

[220] K. Yamanishi and J. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 320–324, Edmonton, Alberta, Canada, 2002.

[221] Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69:125–139, 1995.

[222] K. Zeitouni and N. Chelghoum. Spatial decision tree-application to traffic risk analysis. In *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, volume 0, page 0203, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[223] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: discriminative nearest neighbor for visual object recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136, New York, NY,USA, 2006.

[224] K. Zhang and W. Fan. Forecasting skewed biased stochastic ozone days: analysis, solutions and beyond. *Knowledge and Information Systems*, 14(3):753–764, 2008.

[225] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record*, 25(2):103–114, 1996.

[226] X. Zhang, N. Mamoulis, D. W. Cheung, and Y. Shou. Fast mining of spatial collocations. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 384–393, Seattle, Washington, USA, 2004. ACM.

[227] D. Y. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*, volume 16, pages 321–328, British Columbia, Canada, 2003. MIT Press.

[228] Z. H. Zhou and M. Li. Semi-supervised regression with co-training. In *Proceedings of 19th International Joint Conference on Artificial Intelligence*, pages 908–913, Edinburgh, Scotland, UK, 2005.

[229] X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

[230] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, volume 20, pages 912–919, Melbourne, Florida, USA, 2003.

[231] X. J. Zhu and A. Goldberg. Kernel regression with order preferences. In *Proceedings of the 22nd Conference on Artificial Intelligence*, page 681, Vancouver, British Columbia, Canada, 2007.