UNIVERSITY OF CALIFORNIA
RIVERSIDE

Learning from Time Series in the Presence of Noise: Unsupervised and Semi-Supervised Approaches

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Dragomir Dimitrov Yankov

March 2008

The Dissertation of Dragomir Dimitrov Yankov is approved:

Committee Chairperson

University of California, Riverside

Acknowledgements

Needless to say, I would not reach this stage of graduate school if it was not for my advisor Dr. Eamonn Keogh. I have never worked with another person with so much drive and passion for what they do, and I just hope that at least part of these qualities were acquired by me too. Eamonn taught me the basic practices and knowledge a data mining researcher needs to have, but for what its worth, it is his attitude that probably made the biggest impact on me. Any single time I would talk to him, he would be positive and encouraging. Thank you, Eamonn, for being there for me and for the rest of your students!

I would like to thank my dissertation committee - Dr. Vassilis Tsotras and Dr. Stefano Lonardi. Vassilis sent me my acceptance letter exactly five years ago promising that I will enjoy the atmosphere in UC Riverside. I really did! Stefano was there for my most important publication - the first one. From him I learned that every detail matters, that every word needs to be accurately placed.

I had three great internships with Yahoo!. I worked with incredible people to whom I am greatly thankful. The first summer my mentor was Dr. Dennis DeCoste. Dennis inspired many of my subsequent interests, such as ensemble learning and support vector machines. I would like to thank Dr. Nicolas Mayoraz and the whole Data Mining and Research (DMR) group with whom I spent the other two internships. DMR has been one of the friendliest and most inspiring environments I have worked in. I am sure that joining them next month will be the beginning of an exciting and rewarding working career for me.

I am especially thankful to Terri Phonharath, who would never let me worry about any funding or administrative issues, so I could focus just on my research.

I would like to thank all of my collaborators, but mostly Umaa Rebbapragada and Kin Kan. Umaa's visit to our lab initiated the work on the discord detection algorithm. The insightful discussions with Kin and his advisor in the machine learning group, Dr. Christian Shelton, lead to the locally constrained support vector clustering work.

During my graduate years in Riverside I made so many good friends, whom I am really thankful to for all the happy moments that we had together. Out of all I would like to mention Petko Bakalov. Petko was always there, whenever I needed a friend to rely on.

Finally, I would like to thank the people closest to me for what they have done and still do for me - my parents, my brother and Anelia. Anelia was the reason because of which I started graduate school, and it is her whom I would like to dedicate this work to.

ABSTRACT OF THE DISSERTATION

Learning from Time Series in the Presence of Noise: Unsupervised and Semi-Supervised Approaches

by

Dragomir Dimitrov Yankov

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2008
Dr. Eamonn Keogh, Chairperson

Characteristic for most real-world data mining tasks are the availability of a large number of unlabeled examples and the large rates of noise obliterating the similarity patterns. We study datasets with such characteristics collected across a wealth of domains - web search queries, celestial systems, anthropological artifacts, surveillance footages, etc. Diverse as they are, data from these domains turn out to have intuitive time series representation. As a major theme herein we advocate the belief that the time series representation is versatile enough to allow for accurate learning in the presence of noise, and when there are few or no labeled examples available. This is further achieved without compromising on the efficiency and the scalability of the problems at hand.

For certain applications the noisy patterns themselves can be of particular interest - anomalous trajectories, surprising web queries, erroneously recorded light-curves - identifying them, and drawing the domain expert's attention to them, often leads to unexpected discoveries. The early chapters of the thesis introduce the *time series discords* as a notion of "interesting" outliers in the high dimensional representation space. We develop an effective and highly efficient discord detection algorithm, which used in a parallel fashion can handle hundreds of millions of examples in only a few hours. We then turn our attention to the question of how to decouple the noise from the structured signal within the data. The problem can be especially hard when the time series are embedded within complex, non-convex topological spaces. Local manifold reconstruction techniques, such as Isomap and Locally Linear Embedding, easily fail in the realistic settings of noise. An extension of the Isomap algorithm is thus derived, that remedies the effect of noise and learns more accurately the data embedding subspaces. As a more principled approach, we further develop a dual treatment of the time series manifold reconstruction problem. It computes a global density estimate of the data, which is then regularized locally through the help of multiple factor analyzer models. This results into smoothly reconstructed time series manifolds, unbiased by the effect of the noisy examples. Cast within a semi-supervised framework, our method achieves similar accuracy and several orders of magnitude speed-up over the state of the art transductive learning approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Data mining in the presence of noise

A combination of pure machine learning theory and hands-on data driven practices, data mining has already shaped as a life changing science for humanity. In medicine, for example, lethal illnesses are being detected by exploring relations with early observable preconditions; in environmentology, patterns between human activities and their impact on the environment are studied to help preserve a greener planet; while in a much evolved information technology world, daily queries from 20% of the Earth's population are meaningfully answered through the help of search engines. These, and many more, form a broad spectrum of vital problems, the study and the resolution of which has been largely attributed to data mining.

The two arguably most distinguishing properties of all problems targeted by data mining are the vast amounts of data that they require to be handled, and the non-deterministic, yet ultimately sufficient, nature of their answers. In contrast, machine learning is concerned with the theoretical aspects that can help explain more accurately the data and the hidden dependencies within, while the question of how to make the derived tools practical is often assumed to be outside of its scope. On the other hand, traditional database systems deal with large volumes of data, but the queries they answer are marked by the determinism of the underlying relational algebra. This, while important for a large number of practical tasks, turns out to be inadequate when more subtle dependencies need to be explored. Consider questions of the like - "What are the most relevant web documents that contain the term 'curved manifolds'?", "Is a mail legitimate or a spam?", "How typical is the performance of a stock on the market?", etc. Approaching these questions requires studying distribution patterns, performing approximations, resolving ambiguities etc.

Confronted with queries that involve probabilistic treatment of the data, we need suitable tools that can derive accurate answers. The notion of suitable here goes beyond the ability to overcome the scale of the data, but also into the ability to deal with some intrinsic for most real-world datasets characteristics, such as sparsity within some regions or randomness within certain examples. These characteristics play vital role in the current work. It is the examples, sampled from the sparse regions of the dataspace or with significant fraction of randomness added to them, that are the focus of our discussion. Such examples are referred to in the literature, and also in the current text, as *pattern noise* or simply as *noise* [18, 83].

In many applications, the noisy examples can poise the learning process and obscure the fact that certain patterns of similarity exists in the data. We study instances of those in Chapter 3 and Chapter 4 of the text. For example, the collection of excavated projectiles demonstrated in these chapters, contains many exemplars which have deteriorated over time, deviating from their original shape. Yet, there are others within the collection that form important distinctive groups of arrowheads, indicative for the evolution of certain cultures. For other applications, the noisy patterns themselves can be of particular interest - anomalous trajectories, surprising web queries, erroneously recorded light curves - identifying them, and drawing the domain expert's attention to them, often leads to unexpected discoveries. Examples of such interesting deviating patterns will be the problem of study in Chapter 2 of the thesis.

Through the rest of the text we demonstrate how we can develop a set of data mining approaches, that can efficiently identify the spurious patterns, or can robustly decouple the pattern noise from the actual signal.

1.2 Detecting time series patterns: unsupervised and semi-supervised approaches

In defining tools that scale well and discriminate accurately the noisy examples from the structured data, we find the choice of data representation to be a central one. Compact representations attract with easier interpretability and computational efficiency. Often, however, a compact representation will crudely approximate the data at hand, and hence will be a bad compromise of effectiveness, especially in the presence of noise. A restraining drawback of using complex representations, on the other hand, is that they are memory demanding and computationally inefficient.

As a major theme herein, we advocate the belief that *time series* are the one ubiquitous representation to target all of the above concerns. Through a wealth of examples, in the chapters to come we demonstrate that time series can be expressive enough to allow accurate discrimination, even when high rates of noise obliterate the structured patterns. Surprisingly though, for the high dimensional representation that they comprise, time series still remain genuinely intuitive and lead to easily interpretable results. Extremely efficient methods are also to be derived proving the positive impact of the representation on the scale of the problems that can be tackled through it.

We start the treatment of the time series pattern recognition problem with the formal definition to be utilized through the rest of this work:

Definition 1.2.1. *A time series* $T = t_1, \ldots, t_m$, is defined as an ordered set of scalar or multivariate observations t_i, measured at equal intervals in time.

Time series are known to have virtually no alternatives as representation technique across datasets, such as stock market prices or medical (e.g. EEG or ECG) recordings. There is, however, a large number of other, not so intuitive but equally important domains, in which highly descriptive time series representations arise. Celestial systems, anthropological artifacts, surveillance footages, etc. - all of them turn out to have accurate time series description (see Figure 1.1[1]).

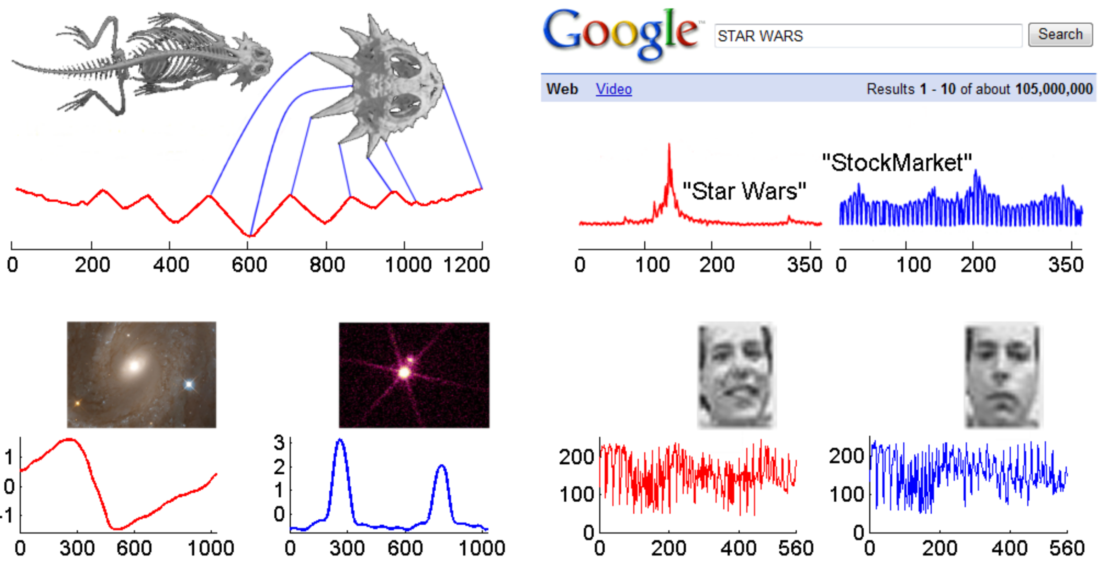

Figure 1.1: Time series representation for different domains. *Top left:* shapes converted to time series by mapping contour points to observations; *right:* web queries represented through their search frequencies during one year. *Bottom left:* Variable star systems (*Cepheids* and *Eclipsing Binaries*) expressed through the magnitude of their brightness during one cycle; *right:* Facial images represented as time series by replacing every pixel with its greyscale intensity.

[1]The first astronomical image shows spiral galaxy NGC 4603 with pulsating Cepheid variables. Image source: the European homepage for the NASA/ESA Hubble space telescope, www.spacetelescope.org
The second astronomical image is an X-ray image of Eclipsing binaries Sirius A and B from the Chandra X-ray Observatory. Image source: Marshall Space Flight Center NASA, www.msfc.nasa.gov

The suitable representation, however, is only one side of the coin when system effectiveness is concerned. The other side is selecting a similarity (i.e. distance) measure, that can accurately exploit the information encoded in the representation. For most of the algorithms presented here we will specifically focus on a set of measures known under the common name of L_p-norms, and defined as:

$$L_p(T, Q) = \Big(\sum_{i=1}^{m} |t_i - q_i|^p \Big)^{1/p} \tag{1.1}$$

As a special case, we will extensively consider one of these norms, the L_2-norm, more widely known as the *Euclidean* distance:

$$L_2(T, Q) = \Big(\sum_{i=1}^{m} |t_i - q_i|^2 \Big)^{1/2} \tag{1.2}$$

The Euclidean distance has been demonstrated to be one of the most accurate distance measures for time series pattern recognition [52]. Similarity searches under this measure also turn out to be extremely efficient, which is due to the fact that it satisfies a set of conditions defining it further as *metric*. This will be discussed, and subsequently utilized, in Chapter 3 of the thesis.

For many of the applications that are studied, the number of observations m within a time series is going to be significantly high, e.g. in orders of thousands, which partly explains the effectiveness of the selected representation. Definition 1.2.1, however, also points out another key difference between any arbitrary high dimensional vector and a time series. Namely, the ordering of the observations within the latter. While all of the presented here algorithms can be equally efficient when applied for any high dimensional representation, the ordering requirement turns out to be important when effectiveness is concerned. What this requirement states is that there are dependencies between neighboring observation. This fact, while not explored by the Euclidean distance, is the primarily reason for the often more accurate performance of distance measures that can account for certain amount of *warping* within the time series. A similarity measure that does this, for example, is the *Dynamic Time Warping* (DTW) [76]. If we denote the length of the time series with a subscript, i.e. $T_i = t_1, \dots, t_i$, then the DTW distance can be formalized recursively as:

$$DTW(T_m, Q_n) = L_2(t_m, q_n) + \min \begin{cases} DTW(T_{m-1}, Q_n) \\ DTW(T_m, Q_{n-1}) \\ DTW(T_{m-1}, Q_{n-1}) \end{cases} \tag{1.3}$$

Both the Euclidean and the DTW distance will be discussed in more detail further in the text, noting the cases when the derived methods can be applied with either of them. In Chapter 3 we will also introduce a derivative distance measure of the Euclidean distance, that proves to be more accurate for rotation invariant similarity comparisons of shapes when represented as time series. Similarity queries utilizing Gaussian kernels will also be studied while discussing the density estimation techniques in Chapter 4.

We now turn our attention to an important question that will dominate this work, i.e.: given the representation and the similarity measure, is there any structure in the subspace occupied by the data? Knowledge about the structural properties can help us find out, for example, whether a web query pattern deviates abnormally from the rest of the web queries, or how many distinctive groups of species populate certain habitat. If such structure is unveiled without additional human intervention, but solely based on the representation and the data placement in space, we then say that we have performed *unsupervised* learning from the time series examples [37, 51, 53, 54].

Often unsupervised time series methods are sufficient to infer the generating distributions. In the presence of high noise rates, however, additional supervision might still be required, despite of the expressiveness of the time series representation and the accuracy of the similarity measure. Partially labeled examples for instance have been shown to improve time series nearest neighbor learners [102], to allow the inference of better clusterings [98], or to help rebuild more accurately the nonlinear subspaces that are occupied by the high dimensional patterns [12]. Here we demonstrate that our unsupervised methods for time series learning in the presence of noise, can also benefit from even small amounts of labeled data, converting them essentially into *semi-supervised* approaches. These semi-supervised approaches will allow us to identify new, class dependent outliers, or to build more robust time series clustering techniques.

1.2.1 Detecting outliers in terabyte-sized datasets

In the early sections of the thesis we start an exploration of a method capable of detecting interesting examples, which appear to be severe outliers within the time series datasets. The definition of the "most significant time series *discord*" is introduced as a notion of the outlier that is furthest away from all other examples. In Figure 1.2, for instance, the time series that appear to be the two most deviating examples are the ones derived from shapes A and E. Here A corresponds to a reptile that is of a more distant genotype compared to the rest of the species (cf. Chapter 3). Its presence in the database might lead to a surprising discovery, such as the finding that certain species populate uncharacteristic for their genotype habitat. Note, that while there is not much randomness added to its representation, we still consider it as a noisy example as it was obviously produced by some randomly interfering process populating with examples sparser regions of the space. Time series E, on the other hand, is obviously extracted erroneously, with noise added to the actual representation. A domain expert should be made aware of it, so that they could correct it or remove it from the database not to bias any further analysis. In both cases, detecting A and E proves to be a data mining task of major significance.

While the dataset from Figure 1.2 contains only 5 data points, one can easily identify domains where discords need to be detected among much larger amounts of data - astronomers look for celestial anomalies among tens of millions of star-light curves [66], while daily web logs of search companies comprise multi-terabyte datasets. We therefore

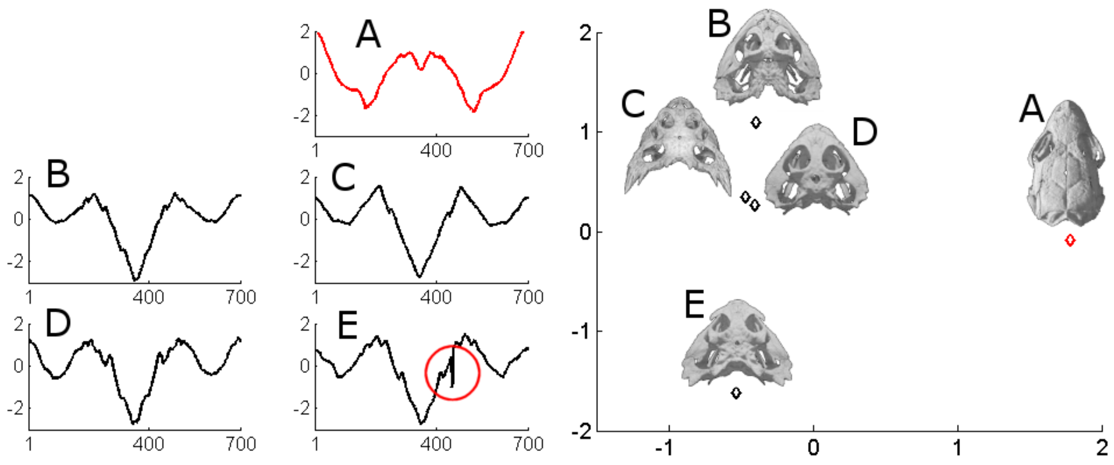

Figure 1.2: *Left:* Time series representation of five reptile skulls. *Right:* Illustrative PCA projection: outliers within the time series space can help us detect unexpected examples in the dataset, such as shape A representing a different type of species, or identify noisy shapes, such as shape E which has been extracted erroneously.

provide in Chapter 2 a discord detection algorithm that scales in reasonable time beyond any dataset attempted so far in the literature. The algorithm can be applied for an arbitrary similarity measure and consumes only a modest amount of memory resources. It is generally studied as an unsupervised method, which only utilizes the pairwise element similarities, as well as the underlying distance distributions. With element class information, however, it is demonstrated to be easily extendable into a semi-supervised approach that is able to identify discords specific for particular classes. Straightforward extensions of the method, such as parallelization will also be discussed further in the text.

The discord detection algorithm, presented in Chapter 2, is intended primarily as a technique for mining global outliers, i.e. examples which emerge as outliers when a global view of the entire data is assumed. As argued before, smaller amounts of randomness (noise) are also a common artifact in real-world data. Accumulated noise causes multiple examples to locally deviate from their original prototype. The end result is a much polluted dataset and subsequently impeded learning process. The question of how to deal with the many noisy examples has therefore motivated many learning and data mining approaches [18, 83]. It becomes even more important when data are embedded within non-convex subsets of the original space. Identifying clusters or class belongings for such noisy non-convex subspaces turns out to be a significant challenge. We thus focus our attention in Chapter 3 at methods that can reconstruct the embedding subspace, and subsequently utilize it in a number of data mining tasks.

1.2.2 Time series manifold reconstruction

Informally [60] specifies *topological manifolds*, or simply manifolds, as "curves and surfaces that might be of a higher dimension". A more formal definition in [60] states that the space X is an n dimensional manifold if for every point of it there is a neighborhood homeomorphic to Euclidean space in $\mathbb{R}^n$. This implies that there is some structure within the neighborhood of every point within the space. In general, we will be interested in structures that are of intrinsically lower dimensionality, than the original high dimensional space, in which they are embedded.

Manifold reconstruction has been of surging interest in machine learning. The reasons for that are twofold. Knowing that a dataset is of intrinsically lower dimensionality, allows for applying different indexing techniques and more tractable computationally algorithms. Furthermore, in analogy to the accurate priors in Bayesian learning, knowing the proper structure of the data always relates to improved learning accuracies. Now the main point that remains to be answered is probably best summarized in [12], where the authors write "It is far from clear why the space of documents should be a manifold. However there is no doubt that it has a complicated intrinsic structure and occupies only a tiny portion of the representation space". For many types of datasets, and time series in particular, a similar question stands open - do they occupy a lower dimensional embedding in the high dimensional space, and if so, what can this be attributed to? For a number of domains, such as time series produced by dynamical systems, the answer is known to be positive [104]. In Chapter 3, we surprisingly observe that manifold structures also appear in the space defined by a special type of *pseudo* time series - i.e. time series extracted from shapes of objects. Figure 1.3, demonstrates that a set of *diatoms*, a type of unicellular plants, can be represented with shape time series, which using a particular dimensionality reduction technique form a nearly smooth one dimensional manifold.

We attribute this to the fact, that the original image data follow the manifolds representing certain geometric transformations, with rotations in particular. Due to the specific pseudo time series representation, the effect of these transformations is preserved in the time series space too. Interestingly enough, a manifold structure is present even when a *rotation invariant* distance measure is defined over this space. In the text to come we show that this rotational measure exhibits metric properties, which helps us derive an extremely efficient algorithm for reconstructing the shape manifolds invariantly to any present rota-

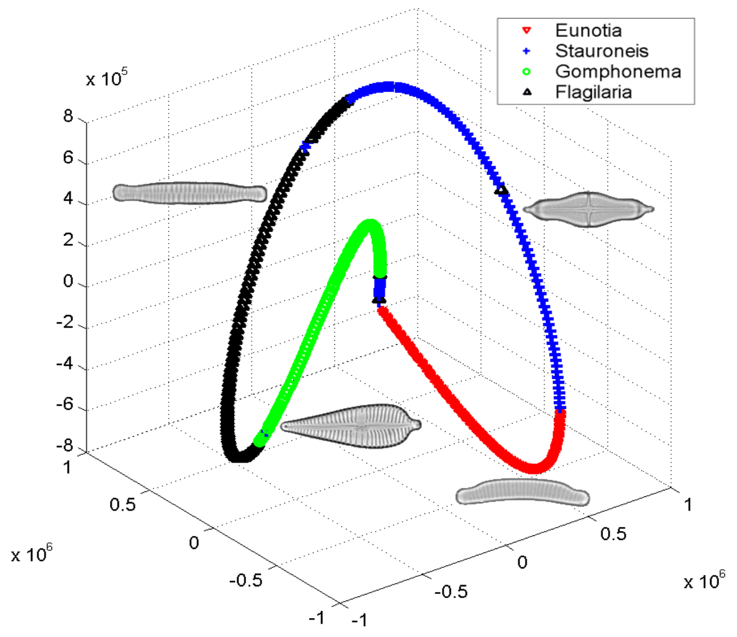

Figure 1.3: Time series extracted from diatom shapes form almost smooth one dimensional manifolds using *b-Isomap* embedding of degree one (cf. Chapter 3). Identifying the individual classes within the manifold, however, is hard without accounting for the possibility of rotations.

tions. The reconstructed manifolds are then used to infer clustering of the shape space, that is considerably more accurate than the one obtained with traditional clustering techniques, such as mixtures of Gaussians.

Again, while learning from shape manifolds, as a significant challenge emerges the effect of the noise. It originates from the lower quality that some of the images may have. For example, many of the microscope diatom images are blurred, which causes edge detection techniques to degrade and to outline some rough, even inaccurate, object contours. The obscured examples are then seen as widely spread Gaussian noise distributed along the main trajectory of the manifold. Manifold reconstruction techniques, such as Isomap [93] and Locally Linear Embedding (LLE) [79], are known to be highly unstable for such noisy manifolds [9]. We thus derive an improved Isomap method capable of isolating the noisy examples and of following closely the true trajectory of the manifolds. Chapter 3 also demonstrates how the unsupervised shape manifold reconstruction can be extended into a semi-supervised clustering approach. Using a sample of labeled data an ensemble of algorithms is presented that infers more accurate clustering for a wide range of noise rates.

1.2.3 Constraining the time series manifold structures

The problems targeted in Chapter 2 and Chapter 3 achieve two complementary results. The former identifies individual severe outliers, as they might capture important anomalies within the data. This is achieved without building a global model, but only by studying the neighborhoods of the individual examples. The latter chapter, on the other hand, focuses on the good examples and derives methods that can filter out the noise to improve the learning process. It also builds a model for the entire dataset - a manifold structure that embeds the observable data. Still this model simply connects multiple local views into a global structure, i.e. it builds a neighborhood graph or a minimum spanning tree. A drawback of this method is that it requires additional supervision in determining how the local views should be mutually embedded: within the *geodesic* structure of the neighborhood graph or within the branches of the minimum spanning tree. In Chapter 4 we overcome this problem building a model capable of inferring structure without additional supervision. A novel approach is presented that combines a global view of the data with multiple local views capable of modeling even large rates of Gaussian distributed noise around the main

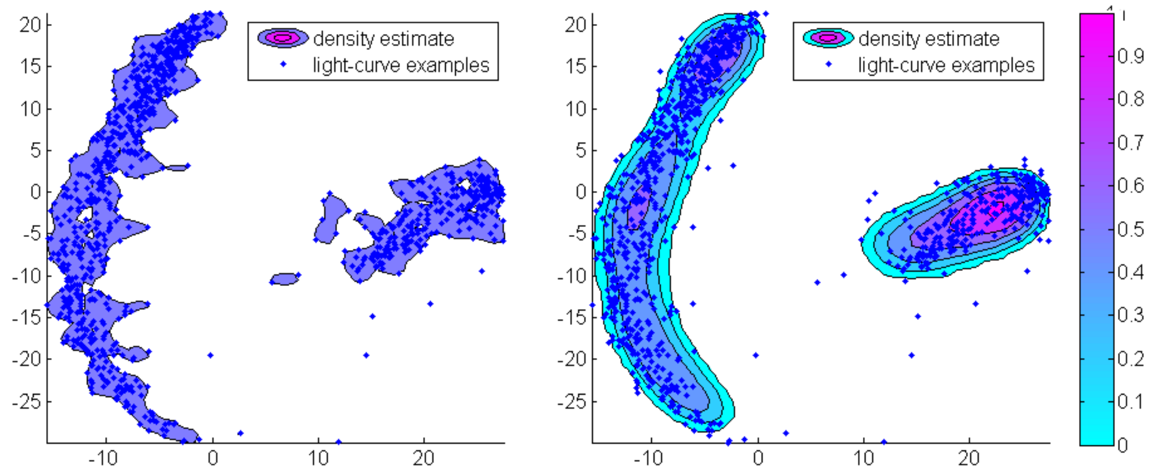

Figure 1.4: PCA projection of a set of *light-curve* time series (cf. Chapter 4). *Left:* density estimation methods alone (e.g. one-class SVM) are unable to smoothly reconstruct the occupied subspace. *Right:* Adding a local prospective (e.g. constraints from an inferred mixture of factor analyzers model) smooths the boundaries of the learned manifolds.

trajectory of the manifold. With the term "global view" here, we refer to one common density support that is met across the entire data space as illustrated in Figure 1.4.

The density support is estimated using the so called *one-class* support vector machines [82, 85]. It computes a global decision functions which outlines contours around the dense regions in the space. The density estimation method, however, cannot recognize whether the dense regions are part of an interesting manifold structure, or simply spurious noisy spots. To facilitate this, a local approach of the data comes into play. We build a mixture of *factor analyzers* - Gaussian like models that can capture noise of different variance along individual dimensions [43]. The analyzers make local decisions of which points constitute noisy examples and do not conform with the rest of the data within a neighborhood. These decisions are then used to regularize the decision function of one-class support vector machines. Through this approach spurious formations become easy to isolate and a cleaner global structure is revealed (see Figure 1.4 right).

Manifold reconstruction is not the only unsupervised task that can benefit of the dual view of the data. One-class classification extends naturally into a clustering scheme [14], called support vector clustering (SVC). One-class SVM possesses all nice theoretical properties attributed to kernel machines for supervised learning, such as provable bounds on the complexity of the induced decision function and conformance with one of the founding principle of the statistical learning theory, namely - the structural risk minimization which guarantees the good generalization properties of the method [94]. Regardless of this its extension, the SVC method, is not very attractive as a generic clustering approach, because it fails to provide for a good control over how many and what clusters are being identified. In Chapter 4 we demonstrate that our improved one-class SVM algorithm remedies this effect, and turns SVC into a robust clustering algorithm. Equally important is the effect of our method on the efficiency of one of the most popular semi-supervised learning techniques - the *transductive* support vector machines. Generally considered impractical when more than a few thousand unlabeled examples are available [50], we demonstrate how our constraint support vector clustering can scale them to efficiently handle several orders of magnitude lager unlabeled datasets.

1.3 Contributions

We can summarize the main contributions of the thesis as follows:

- We introduce the most efficient time series outlier detection algorithm known thus far in the field of data mining. The algorithm is demonstrated with unprecedented in the literature terabyte-sized datasets. It is shown to be also extremely accurate, detecting the essential outliers within tens of millions of examples in just a few hours on a single computer. Additional supervision in the form of class labels adapts the method for detecting important class dependent outliers.

- We demonstrate a highly efficient approach for accurately computing the similarities between two-dimensional shapes using a time series representation. A rotationally invariant distance measure is introduced, for which metric properties are proven. This allows us to perform rotationally invariant shape similarity searches many orders of magnitude faster, than the strawman brute force techniques.

- We present an improvement of the Isomap approach that overcomes its instability in the presence of noise, making the algorithm applicable beyond the instructive scholar datasets in the literature, but for real-world data mining applications instead. With the use of a limited number of labels the method is extended into an ensemble procedure for robustly reconstructing the manifold structure of the datasets regardless of the rates of noise that obliterate them.

- Finally, we derive a regularization technique that smooths the density estimate of one-class support vector machines. Combining local and global views of the data, our method correctly reconstruct the manifolds that embed the sample space even in the presence of noise. This also allows for deriving a practically useful clustering algorithm, generalizing the one-class SVM method. In a semi-supervised setting, we further show that our method can introduce several orders of magnitude speed-up in the performance of transductive support vector machines, making them applicable for datasets that contain hundreds of thousands of unlabeled examples without sacrificing in the achievable accuracy.

This thesis generalizes our own work presented in the following publications [109, 110, 113, 114].

Chapter 2

Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets

The chapter discusses the problem of detecting largely deviating time series. These are time series that are either corrupted with large rates of additive noise or are generated by a randomly interfering process that populates with examples otherwise sparse parts of the representation space. In both cases we treat the examples as excessively noisy.

The problem of finding such unusual time series has recently attracted much attention, and several promising methods are now in the literature. However, virtually all proposed methods assume that the data reside in main memory. For many real-world problems this is not be the case. For example, in astronomy, multi-terabyte time series datasets are the norm. Most current algorithms faced with data which cannot fit in main memory resort to multiple scans of the disk/tape and are thus intractable. Here we show how one particular definition of unusual time series, the time series discord, can be discovered with a disk aware algorithm. The proposed algorithm is exact and requires only two linear scans of the disk with a tiny buffer of main memory. Furthermore, it is very simple to implement. We use the algorithm to provide further evidence of the effectiveness of the discord definition in areas as diverse as astronomy, web query mining, video surveillance, etc., and show the efficiency of our method on datasets which are many orders of magnitude larger than anything else attempted in the literature.

2.1 Introduction

The problem of finding unusual (abnormal, novel, deviant, anomalous) time series has recently attracted much attention. Areas that commonly explore such unusual time series are, for example, fault diagnostics, intrusion detection, and data cleansing. There, however, are other more uncommon yet interesting applications too. For example, a recent paper suggests that finding unusual time series in financial datasets could be used to allow diversification of an investment portfolio, which in turn is essential for reducing portfolio volatility [100].

Despite its importance, the detection of unusual time series remains relatively unstudied when data reside on external storage. Most existing approaches demonstrate efficient detection of anomalous examples, assuming that the time series at hand can fit in main memory. However, for many applications this is not be the case. For example, multi-terabyte time series datasets are the norm in astronomy [66], while the daily volume of web queries logged by search engines is even larger. Confronted with data of such scale current algorithms resort to numerous scans of the external media and are thus intractable. In this chapter, we present an effective and efficient disk aware algorithm for mining unusual time series. The algorithm is exact and requires only two linear scans of the disk with a tiny buffer of main memory. Furthermore, it is simple to implement and does not require tuning of multiple unintuitive parameters. The introduced method is used to provide further evidence of the utility of one particular definition of unusual time series, namely, the time series *discords*. The effectiveness of the discord definition is demonstrated for areas as diverse as astronomy, web query mining, video surveillance, etc. Finally, we show the efficiency of the proposed algorithm on datasets which are many orders of magnitude larger than anything else attempted in the literature. In particular we show that our algorithm can tackle multi-gigabyte datasets containing tens of millions of time series in just a few hours.

2.2 Related work and background

The time series discord definition was introduced in [54]. Since then, it has attracted considerable interest and follow-up work. For example, [32] provide independent confirmation of the utility of discords for discovering abnormal heartbeats, in [5] the authors apply discord discovery to electricity consumption data, and in [103] the authors modify the definition slightly to discover unusual shapes.

However, all discord discovery algorithms, and indeed virtually all algorithms for discovering unusual time series under any definition, assume that the entire dataset can be loaded in main memory. While main memory size has been rapidly increasing, it has not kept pace with our ability to collect and store data.

There are only a handful of works in the literature that have addressed anomaly detection in datasets of anything like the scale considered in this work. In [33] the authors consider an astronomical dataset taken from the Sloan Digital Sky Survey, with 111,456 records and 68 variables. They find anomalies by building a Bayesian network and then looking for objects with a low log-likelihood. Because the dimensionality is relatively small and they only used 10,000 out of the 111,456 records to build the model, all items could be placed in main memory. They report 3 hours of CPU time (with a 400MHz machine). For the secondary storage case they would also require at least two scans, one to build the model, and one to create anomaly scores. In addition, this approach requires the setting of many parameters, including choices for discretization of real variables, a maximum number of iterations for EM (a sub-routine), the number of mixture components, etc.

In a sequence of papers Otey and colleagues [44] introduce a series of algorithms for mining distance based outliers. Their approach has many advantages, including the ability to handle both real-valued and discrete data. Furthermore, like our approach, their approach also requires only two passes over the data, one to build a model and one to find the outliers. However, it also requires significant CPU time, being linear in the size of the dataset but quadratic in the dimensionality of the examples. For instance, for two million objects with a dimensionality of 128 they report needing 12.5 hours of CPU time (on a 2.4GHz machine). In contrast, we can handle a dataset of size two million objects with dimensionality 512 in less than an hour, most of which is I/O time.

Distance based outliers are also the problem of study in Knorr et al. [58] and Tao et al. [91]. Both works discuss a quadratic (in the dataset size) nested loop algorithm for outlier detection and subsequently suggest ways for its improvement. Knorr et al. [58] propose an algorithm that performs three scans through the database and also requires significant amount of main memory. The algorithm further uses a partitioning scheme, whose performance deteriorates in higher dimensions. Tao and colleagues sample the data space to build a set of partitions which they later use to prune non-outlier examples. They explicitly require that the distance function used be a metric, but for time series data non-metric functions have been demonstrated to be often superior [106]. The method again is intended for a lower dimensional datasets and is demonstrated to require two linear scans of the data base.

Another method that scales to large disk resident datasets, requiring only two linear scans of the disk, was recently proposed by Angiulli and colleagues [8]. An extension of the method for online detection of distance based outliers from streaming data [7] has also been presented by the same authors. To perform faster range queries, the algorithm again builds an index in main memory, based on the concept of pivot points. Apart of the already pointed problems of degrading efficiency in higher dimensions and the requirement of metric distance function, using pivots for indexing poses additional challenges too. For example, selecting good pivot points might itself require to first detect a set of outliers. This is so, because as Shapiro [86] suggests good pivots tend to be points that are far from any dense region in the data.

The problem of outliers detection in higher dimensional spaces was target in an influential paper by Jagadish et al. [48]. They propose to find unusual time series (which they call *deviants*) with a dynamic programming approach. Again this method is quadratic in the length of the time series, and thus it is only demonstrated on kilobyte sized datasets.

The discord introducing work [54] suggests a fast heuristic technique (termed *HOT-SAX*) for pruning quickly the data space and focusing only on the potential discords. The authors obtain a lower dimensional representation for the time series at hand and then build a trie in main memory to index these lower dimensional sequences. A drawback of the approach is that choosing a very small dimensionality size results in a large number of discord candidates, which makes the algorithm essentially quadratic, while choosing a more accurate representation increases the index structure exponentially. The datasets used in that evaluation are also assumed to fit in main memory.

In order to discover discords in massive datasets we must design special purpose algorithms. The main memory algorithms achieve speed-up in a variety of ways, but all require random access to the data. Random access and linear search have essentially the same time requirements in main memory, but on disk resident datasets, random access is expensive and should be avoided where possible. As a general rule of thumb in the database community it is said that random access to just 10% of a disk resident dataset takes about the same time as a linear search over the entire data. In fact, recent studies suggest that this gap is widening. For example, [78] notes that the internal data rate of IBM's hard disks improved from about 4 MB/sec to more than 60 MB/sec. In the same time period, the positioning time only improved from about 18 msec to 9 msec. This implies that sequential disk access has become about 15 times faster, while random access has only improved by a factor of two.

Given the above, efficient algorithms for disk resident datasets should strive to do only a few sequential scans of the data.

2.3 Notation

We now utilize Definition 1.2.1, which formalizes a time series as a sequence of m evenly spaced ordered observations. When m is very large, looking at the time series as a whole does not reveal much useful information. Instead, one might be more interested in *subsequences* $C = t_p, \ldots, t_{p+n-1}$ of T with length $n << m$ (here p is an arbitrary position, such that $1 \leq p \leq m - n + 1$).

Working with *time series databases* there are usually two scenarios in which the examples in the database might have been generated. In one of them the time series are generated from short distinct events, e.g. a set of astronomical observations (see Section 2.6.1). In the second scenario, the database simply consists of all possible subsequences extracted from the time series of a long ongoing process, e.g. the yearly recordings of a meteorological sensor. Knowing whether the database is populated with subsequences of the same process is essential when performing pattern recognition tasks. The reason for this is that two subsequences C and M extracted from close positions p_1 and p_2 are very likely to be similar to one another. This might falsely lead to a conclusion that the subsequence C is not a rare example in the database. In these cases, when p_1 and p_2 are not "significantly" different, the subsequences C and M are called *trivial matches* [31]. The positions p_1 and p_2 are significantly different with respect to a distance function $Dist$, if there exists a subsequence Q starting at position p_3, such that $p_1 < p_3 < p_2$ and $Dist(C, M) < Dist(C, Q)$.

With the above notation in hand, we can now present the formal definition of time series discords:

Definition 2.3.1. *Time Series Discord:* Given a database S, the time series $C \in S$ is called the most significant discord in S if the distance to its nearest neighbor (or its nearest non-trivial match in case of subsequence databases) is largest. I.e. for an arbitrary time series $M \in S$ the following holds: $\min(Dist(C, Q)) \geq \min(Dist(M, P))$, where $Q, P \in S$ (and Q, P are non-trivial matches of C and M in case of subsequence databases).

Similarly, one could define the second-most significant or higher order discords in the database. To capture the case of a small group of examples in the space that are close to each other but far from all other examples, we might want to generalize Definition 2.3.1 so that the distance to the k-th instead of the first nearest neighbor is considered:

Definition 2.3.2. K^{th} *Time Series Discord:* Given a database S, the time series $C \in S$ is called the most significant k-th discord in S if the distance to its k-th nearest neighbor (or its k-th nearest non-trivial match in case of subsequence databases) is largest.

The generalized view of discords (Definition 2.3.2) is equivalent to another notion of unusual time series that is frequently encountered in the literature, i.e. the *distance based outliers* [58]. The definition can be generalized further to compute the average distance to all k nearest neighbors, which is in fact the non-parametric density estimation approach [87]. The algorithm proposed in the next sections can easily be adapted with any of these outlier definitions. We use Definition 2.3.1 because of its intuitive interpretation. Our choice is further justified by the effectiveness of the discord definition demonstrated in Section 2.6.1.

Unless otherwise specified we will use as a distance measure the Euclidean distance (cf. equation (1.2)), still the derived algorithm can be utilized with any distance function which may not necessarily be a metric, such as the dynamic time warping (equation (1.3)) or the *uniform scaling* distance [112]. In computing $Dist(C, M)$ we expect that the arguments have been normalized to have mean zero and a standard deviation of one. Throughout the empirical evaluation of the chapter we assume that all subsequences are stored in the database in the above normalized form. This requirement is imposed so that the nearest neighbor search is invariant to transformations, such as shifting or scaling [52].

2.4 Finding discords in secondary storage

So far we have introduced the notion of time series discords, which is the focus of the current chapter. Here, we are going to present an efficient algorithm for detecting the top discords in a dataset. Firstly, the simpler problem of detecting what we call *range discords* is addressed, i.e. given a range r the presented method efficiently finds all discords at distance at least r from their nearest neighbor. As providing r may require some domain knowledge, the next section will demonstrate a sampling procedure that will solve the more general problem of detecting the top dataset discords without knowing the range parameter.

The discussion is limited to the case where the database S contains $|S|$ separate time series of length n. If instead the database is populated with subsequences from a long time series the fundamental algorithm remains unchanged, with some additional minor bookkeeping to discount trivial matches.

2.4.1 Discord refinement phase

The range discord detection algorithm has two phases: a candidate selection phase (phase1), and a discord refinement phase (phase2). For clarity of exposition we first outline the second phase of the algorithm.

The discord refinement phase accepts as an input a subset $C \subset S$ (built in phase1), which is assumed to contain all discords C_j at distance $C.dist_j \geq r$ from their nearest neighbor in S, and possibly some other time series from S. If this is case, then the following simple algorithm can be used to prune the set C to retain only the true discords with respect to the range r:

Although all discords are assumed to be in C, prior to starting Algorithm1 it is unknown which items in C are true discords, and what their actual discord distances are. Initially, all these distance are set to infinity (line 2). The above algorithm simply scans the disk resident database, comparing the list of candidates to each item on disk. The actual distance is computed with an optimized procedure which uses an upper bound for early termination [54] (line 9). For example, in the case of Euclidean distance, the *EarlyAbandon* procedure will stop the summation $Dist(S_i, C_j) = \sum_{k=1}^{n} \sqrt{(s_{ik} - c_{jk})^2}$ if it reaches $k = p$, such that

Algorithm 1 Discord Refinement Phase

```
procedure $[C, C.dist]$=DC_Refinement(S, C, r)
in:  $S$: disk resident dataset of time series
     $C$: discord candidates set
     $r$: discord defining range
out: $C$: list of discords
     $C.dist$: list of NN distances to the discords
1:  for $j = 1$ to $|C|$ do
2:      $C.dist_j = \infty$
3:  end for
4:  for $\forall S_i \in S$ do
5:      for $\forall C_j \in C$ do
6:          if $S_i == C_j$ then
7:              $continue$
8:          end if
9:          $d = EarlyAbandon(S_i, C_j, C.dist_j)$
10:         if $(d < r)$ then
11:             $C = C \setminus C_j$
12:             $C.dist = C.dist \setminus C.dist_j$
13:         else
14:             $C.dist_j = min(C.dist_j, d)$
15:         end if
16:     end for
17: end for
```

$1 \leq p \leq n$ for which $\sum_{k=1}^{p}(s_{ik} - c_{ik})^2 \geq C.dist_j^2$. If this happens then the new item S_i obviously cannot improve on the current nearest neighbor distance $C.dist_j$, and thus the summation may be *abandoned*.

Based on the distance calculations, for each S_i there are three situations:

1. The distance between the discord candidate in C and the item on disk is greater than the current value of $C.dist_j$. If this is true we do nothing.

2. The distance between the discord candidate in C and the item on disk is less that r. If this happens it means that the discord candidate can not be a discord, it is a false positive. We can permanently remove it from the set C (line 11 and line 12).

3. The distance between the discord candidate in C and the item on disk is less than the current value of $C.dist_j$ (but still greater than r, otherwise we would have removed it). If this is true we simply update the current distance to the nearest neighbor (line 14).

It is straightforward to see that upon completion of Algorithm1 the subset C contains only the true discords at range at least r, and that no such discord has been deleted from C, provided that it has already been in it. The time complexity for the algorithm depends critically on the size of the subset size $|C|$. In the pathological case where $|C| = |S|$, it becomes a brute force search, quadratic in the size $|S|$. Obviously, such candidate set could be produced if the range parameter r is equal to 0. If, however, the candidate set C contains just one item, the algorithm becomes essentially a linear scan over the disk for the nearest neighbor to that one item. A very interesting observation is that if the candidate set C contains two or three items instead of one, this will most likely not change the time for the algorithm to run. This is so, because for a very small $|C|$ the CPU required calculations will execute faster than the disk reading operations, and thus the running time for the algorithm is just the time taken for a linear scan of the disk data. To summarize, the efficiency of Algorithm1 depends on the two critical assumptions that:

1. For a given value of r, we can efficiently build a set C which contains all the discords with a discord distance greater than or equal to r. This set may also contain some non-discords, but the number of these "false positives" must be relatively small.

2. We can provide a "good" value for r which allows us to do '1' above. If we choose too low of a value, then the size of set C will be very large, and our algorithm will become slow, and even worse, the set C might no longer fit in main memory. In contrast, if we choose too large a value for r, we may discover that after running the algorithm above the set C is empty. This will be the correct result; there are simply no discords with a distance of that value. However, we probably wanted to find a handful of discords.

2.4.2 Candidates selection phase

In this section we address the first of the above assumptions, i.e. given a threshold r we present an efficient algorithm for building a compact set C with a small number of false positives. A formal description of this candidate selection phase is given as Algorithm2.

The algorithm performs one linear scan through the database and for each time series S_i it validates the possibility for the candidates already in C to be discords (line 5). If a candidate fails the validation, then it is removed from this set. In the end, the new S_i is either added to the candidates list (line 11), if it is likely to be a discord, or it is omitted. To show the correctness of this procedure, and hence of the overall discord detection algorithm, we

Algorithm 2 Candidates Selection Phase

```
procedure [C]=DC_Selection(S, r)
in:  $S$: disk resident dataset of time series
     $r$: discord defining range
out: $C$: list of discord candidates
1:  $C = \{S_1\}$
2:  for $i = 2$ to $|S|$ do
3:      $isCandidate = true$
4:      for $\forall C_j \in C$ do
5:          if $(Dist(S_i, C_j) < r)$ then
6:              $C = C \setminus C_j$
7:              $isCandidate = false$
8:          end if
9:      end for
10:     if $(isCandidate)$ then
11:         $C = C \cup S_i$
12:     end if
13: end for
```

first point out an observation that holds for an arbitrary distance function:

Proposition 2.4.1. *Global Invariant.* Let S_i be a time series in the dataset S and d_{s_i} be the distance from S_i to its nearest neighbor in S. For any subset $C \subset S$ the distance d_{c_i} from S_i to its nearest neighbor in C is larger or equal to d_{s_i}, i.e. $d_{c_i} \geq d_{s_i}$.

Indeed, if the nearest neighbor of S_i is part of C then $d_{s_i} = d_{c_i}$. Otherwise, as C does not contain elements outside of S, the distance d_{c_i} should be larger than d_{s_i}.

Using the above global invariant, we can now easily justify the following proposition:

Proposition 2.4.2. Upon completion of Algorithm2, the candidates list C contains all discords S_i at distance $d_{s_i} \geq r$ from their nearest neighbors in S.

Proof. Let S_i be a discord at distance $d_{s_i} \geq r$ from its nearest neighbor in S. From the global invariant it follows that the distance d_{c_i} from S_i to its nearest neighbor in C is larger or equal to d_{s_i}. Therefore, the condition on line 5 of the algorithm will never be satisfied for S_i and hence it will be added to the candidates list (line 11). $\square$

Proposition 2.4.2 together with the analysis presented for the refinement phase demonstrate the overall correctness of the algorithm. More formally, the following proposition holds:

Proposition 2.4.3. *Correctness.* The candidates selection and the refinement steps detect the discords and only the discords at distance $d_{s_i} \geq r$ from their nearest neighbor in S.

The time complexity of the presented discord detection algorithm is upper-bounded by the time necessary to scan the database twice plus the time necessary to perform all distance computations, which has complexity $O(f{=}\max(|C|)|S|)$. In the experimental evaluation we will demonstrate that, for a good choice of the range parameter, the function f is essentially linear in the database size $|S|$.

2.5 Finding a good range parameter

The range discord detection algorithm presented in the previous section is deterministic in the sense suggested by Proposition 2.4.3, i.e. it finishes by either identifying all discords at range r, or by returning an empty set which indicates that no elements have the required property. Providing a good value for the threshold parameter, however, may not be very intuitive. Furthermore, it may also be the case that the users would like to detect the top k discords regardless of the distance to their neighbors. In those cases, specifying a large threshold will result in an empty set, while a very small range parameter may have high time and space complexity. With this in mind, a reasonable strategy to detect the top k discords would be to start with a “relatively large” r and if in the end $|C| < k$, to restart the algorithm with a smaller parameter. Such iterative restarts will increase the number of database scans, yet we argue that with a sampling procedure we can obtain a good estimate for r that decreases the probability of having multiple scans of the database. We further provide a way to reevaluate the range parameter, so that if a second run of the algorithm is required, the new value of r with high probability will lead to a solution.

A good estimate for the range parameter can easily be obtained by studying the nearest neighbor distance distribution (*nndd*) of the dataset, and more precisely the number of elements that fall in its tail. Computing the *nndd*, however, is hard, especially in high dimensional spaces as is the case with time series [16, 90]. The available methods require that random portions of the space are sampled and the nearest neighbor distances in those portions to be computed. Unfortunately, for a robust estimate, this requires scanning the entire database once, regardless of whether an index is available, and also involves some extensive computations [90]. Another drawback of this approach is that the *nndd* is also dependent on the number of elements in the data, which means that if new sequences are added to the dataset the whole evaluation procedure should be performed again. Consider for example the graphs in Figure 2.1.

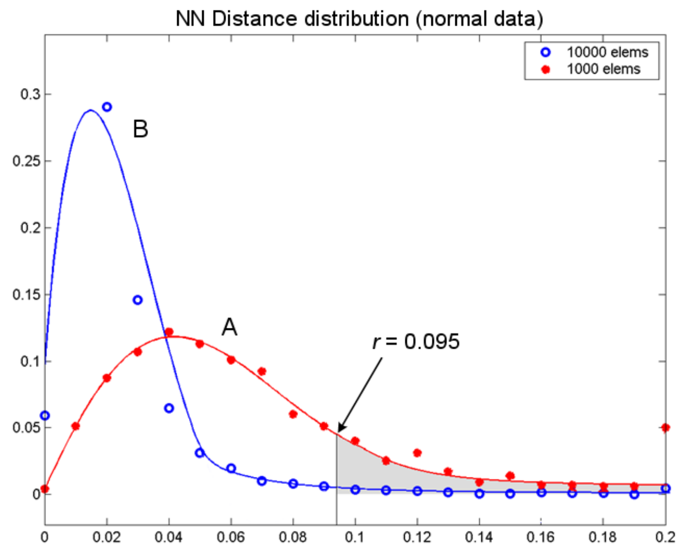

Figure 2.1: Points sampled from the same normal distribution produce different nearest neighbor distance distributions. The mean and the volume of the tail cut by r decrease with adding more data.

Both graphs show the *nndd* for a normally distributed two dimensional dataset $S \in \mathcal{N}(0,1)$. Graph A represents the probability density function when $|S| = 10^3$, while graph B shows the function when $|S| = 10^4$. Intuitively, the mean of the distribution shifts to zero as new points are added, because for larger percentage of the points their nearest neighbors are likely to be found in close proximity to them. For infinite tail data distributions though (as the normal), increasing the sample size also increases the chance of having elements sampled from its tail. These elements will be outliers and are likely to be far from the other examples. Therefore, their nearest neighbor distances will fall in the tail of the corresponding distance distribution too.

Using the above intuition, rather than sampling from the distance distribution, we perform the less expensive sampling from the data distribution and compute the *nndd* of this sample. The exact steps of the sampling procedure are:

1. Select a uniformly random sample S' from S. In the evaluation, for datasets of size $|S| \geq 10^6$ we choose $|S'| = 10^4$. For the smaller datasets we use $|S'| = 10^3$.

2. If the user requires that k discords are detected in their data, then using a fast memory based discord detection method (e.g. [103]) detect the top k discords in S'. Order the nearest neighbor distances $d_i, i = 1..k$ for these discords in S'. I.e. we have $d_1 \geq d_2 \geq \ldots \geq d_k$.

3. Set $r = d_k$.

Note that S' is an unbiased sample from the data and it can be used if new examples generated by the same underlying process are added to the database. This means that we do not need to run the sampling procedure every time that the dataset is updated.

It is relatively easy to see that the above procedure is unlikely to overflow the available memory, regardless of the data distribution. To demonstrate this, consider for example the case when $|S| = 10^6$ and $|S'| = 10^4$. The probability that none of the top 10^3 discords fall in S' is $\hat{p} = \binom{10^6-10^3}{10^4}/\binom{10^6}{10^4}$, which using Stirling's approximation gives $\hat{p} \sim e^{-10}$. This implies that S' almost certainly contains one of the top 10^3 discords. If that discord is S_i, from the global invariant in Section 2.4.2 it follows that its nearest neighbor distance $d_{s'_i}$ in S' is larger or equal to its nearest neighbor distance d_{s_i} in S. But we also have that $d_1 \geq d_{s'_i}$, which leads to $d_1 \geq d_{s_i}$. This means that if we set r to d_1 (or equivalently to d_k, for small k), it is very likely that r will be larger than the nearest neighbor distance of the 10^3-th discord in S. As will be demonstrated in the experimental evaluation, the majority of the time series that are not discords and enter C during the candidate selection phase get removed from the list very quickly which restricts its maximum size to at most several orders of magnitude the size of the final discord set. Therefore, for the above example the maximum amount of memory required will be linear in the amount of memory necessary to store 10^3 time series. Slightly relaxed, but still reasonable, upper bounds can be demonstrated even when S contains an order of 10^8 examples.

The more challenging case is the one when at the end of the discord detection algorithm we have $|C| < k$. In this situation we will need to restart the whole algorithm, yet this time a better estimate for the threshold r can be computed, so that no other restarts are necessary. For the purpose, prior to running the algorithm, a second sample S'' of size 100 is drawn uniformly at random from S'. During the candidates selection phase, for every element S_i in the database, apart of updating the candidates list C, we also update the nearest neighbor distances $S''.dist_q, q = 1..100$. As the size of S'' is relatively small, this will not increase significantly the computational time of the overall algorithm. At the same time, the list $S''.dist$ will now contain an unbiased estimate of the true nearest neighbor distance distribution. Selecting a threshold $r' = \max_{q=1..100}(S''.dist_q)$ will lead to C having on average 1% of the examples. Finally, if k is much smaller than 1% the size of S, but still larger than the size $|C|$ obtained for the initial parameter r, we might further consider an intermediate value r'', such that $r' < r'' < r$ and one that will increase sufficiently the initial size $|C|$.

2.6 Experimental evaluation

In this section we conduct two kinds of experiments. Although the utility of discords has been noted before, e.g. in [5, 32, 40, 54, 103], we first provide additional examples of its usefulness for areas where large time series databases are traditionally encountered. Then we empirically demonstrate the scalability of our algorithm.

2.6.1 The utility of time series discords

Star light-curve dataset

Globally there are myriads of telescopes covering the entire sky and constantly recording massive amounts of valuable astronomical data. Having humans to supervise all observations is practically impossible [66].

The goal for this evaluation is to see to what extent the notion of discords, as specified in Definition 2.3.1, agrees with the notion of astronomical anomalies as suggested by methods used in the field. The data used in the evaluation are light-curve time series from the Optical Gravitational Lensing Experiment [1]. A light-curve is a real-valued time series of light magnitude measurements. The series are derived from telescopic images of the night sky taken over time. Astronomers identify each star in the image and convert the star's manifestation of light into a light magnitude measurement. The set of measurements from all images for a given star results in a light-curve. The light-curves that we obtained for this study are pre-processed (containing a uniform number of points) by domain experts.

The entire dataset contains 9236 light-curves of length 1024 points. The curves are produced by three classes of star objects: *Eclipsed Binaries* - EB (2580 examples); *Cepheids* - Ceph (1329), and *RR Lyrae variables* - RRL (5326) (see Figure 2.2). Both Ceph and RRL stars have very similar pulsing pattern which explains the similarity in their light-curve shape.

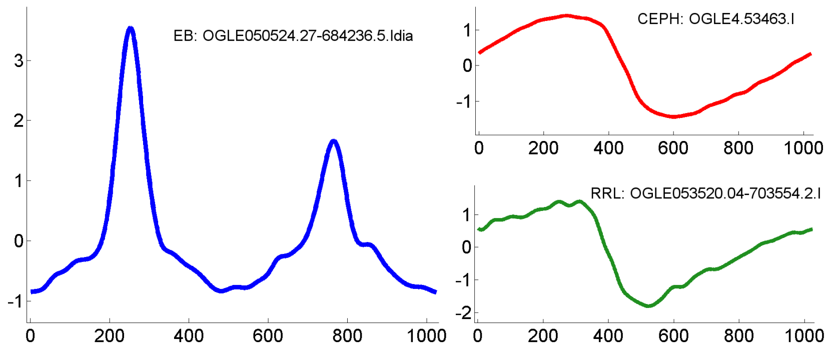

Figure 2.2: Typical examples from the three classes of lightcurves: Left) Eclipsed Binary, Right Top) Cepheid, Bottom) RR Lyrae.

Having information about all or part of the labels, can convert the presented in this chapter unsupervised method into an effective and efficient supervised or semi-supervised outlier detection approach. To evaluate its application as such, in the following experiment we assume all labels are available. We now compare our method with an outlier detection technique developed by domain experts. More precisely, for each of the three classes we also compute the ranking of their examples for being anomalous based on the results of the first method presented in [75]. This gives us, for instance, for the topmost anomaly in every class the ranking 0, the second anomaly has ranking 1, and so on. The method is an $O(n^2)$ algorithm that exhaustively computes the similarity (via cross correlation) between each pair of light-curves. The anomaly score for each light-curve is simply the weighted average of its $n-1$ similarity scores.

We then compute the top ten discords in each of the three classes and compare them with the top ten anomalies inferred with the above ranking. The sampling procedure described in Section 2.5 is performed with a set S' of size 10^3 elements and the threshold r is selected so that at least ten elements from each class fall in the tail of the distance distribution computed on S' (we obtained $r = 6.22$ using Euclidean distance). Running the discord finding algorithm produces a discord set C of size 1161. Figure 2.3 shows several examples of the most significant discords in each class.

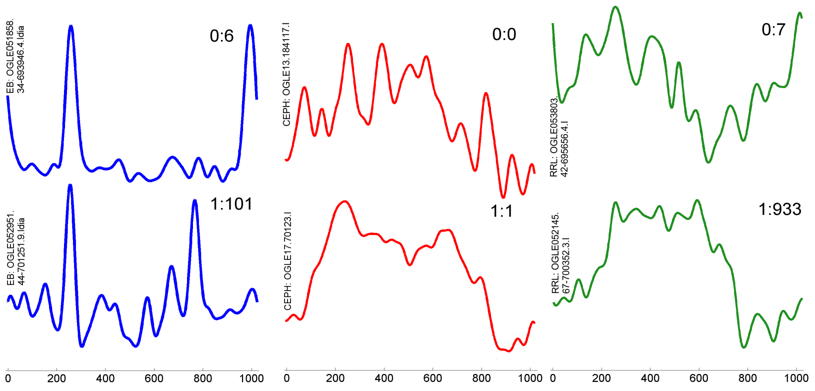

Figure 2.3: Top light-curve discords in each class. For each time series on the top right corner are indicated its *discord rank : anomaly rank*.

One of the top ten EB discords is also among the top ten EB anomalies, three of the top ten RRL discords are among the top ten RRL anomalies and six of the CEPH discords are among the corresponding anomalies. The poor consensus between the one nearest neighbor discords and the anomalies for the EB class results from the fact that the Euclidean distance does not account well for the small amount of warping that is present between the two magnitude spikes. Substituting the Euclidean distance with a phase invariant or a dynamic time warping distance function may improve on this problem. For the other two classes the discord definition is more consistent with the expert opinion on the outliers. Even for elements where they disagree significantly, the discord algorithm still returns some intuitive results. For example, the second most significant RRL discord (see Figure 2.3, bottom right) deviates greatly from the expected RRL shape.

Web queries dataset

Another domain where large scale time series datasets are observed daily are the search engines query logs. For example, we studied a dataset consisting of MSN web queries made in 2002. A casual inspection reveals that most web query logs seem to fall into a handful of patterns. Most have a "background" periodicity of seven days, which reflects the fact that many people only have access to the web during the workweek. This background weekly pattern is sometimes augmented by seasonal effects or bursts due to news stories. The two curves labeled "Stock Market" and "Germany" in Figure 2.4 are such examples. Another common type of pattern we call the *anticipated burst*; it consists of a gradual build up, a climax and a fall off. This is commonly seen for seasonally related items ("Easter", "Tour de France", "Hanukkah") and for movie releases as in "Spiderman" and "Star Wars".

Also common is the *unanticipated burst*, which is seen after an unexpected event, such as the death of a celebrity. This pattern is characterized by a near instantaneous burst,

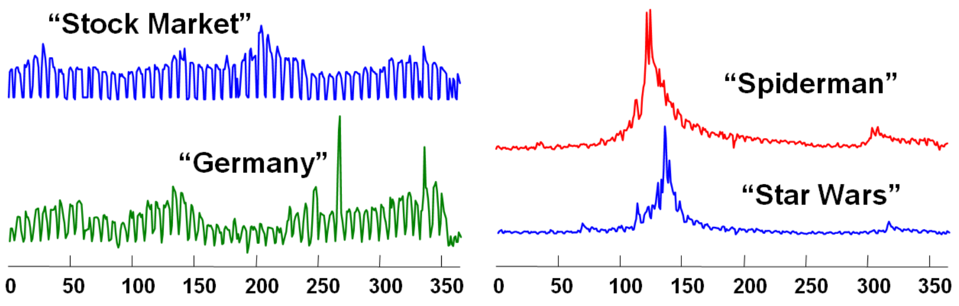

Figure 2.4: Some examples of typical patterns in web query logs in 2002. Most patterns are dominated by a weekly cycle, as in "stock market" or "Germany", with seasonal deviations and bursts in response to news stories. The *anticipated burst* is seen for movie releases such as "Spiderman/Star Wars", or for seasonal events.

followed by a tapering off. Given that both anticipated and unanticipated bursts can happen at any point in the year, we use phase invariant Euclidian distance as discord distance measure. The number one discord is shown in Figure 2.5.

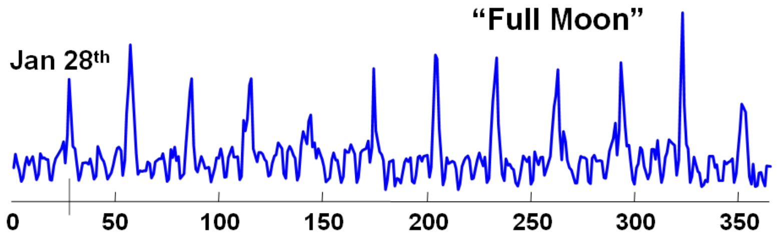

Figure 2.5: The number one discord in the web query log dataset is "Full Moon". The first full moon of 2002 occurred on January 28th at 22:50 GMT. The periodicity of the subsequent spikes is about 29.5 days, which is the length of the synodic month.

This discord makes perfect sense with a little hindsight. Unlike weather or cultural events which are intrinsically local, the phases of the moon are perhaps the only changing phenomena that all human beings can observe. While some other queries have a weak periodicity corresponding to calendar months, this query has a strong periodicity of 29.5 days, corresponds to the synodic month.

Population growth dataset

We can further demonstrate the utility of discords by examining datasets for which we need external sources of knowledge to evaluate findings. We examined a dataset of population growth rates of 206 countries covering 1965 to 2005. We wanted to know the most dramatic 5-year events in this dataset, so we queried the data for the top 5-year discords. Figure 2.6 shows the top two such discords, together with some other representative counties (Argentina, Belgium, Cameroon, Canada, Honduras, Hong Kong, Iceland, India, Indonesia, Ireland) for contrast.

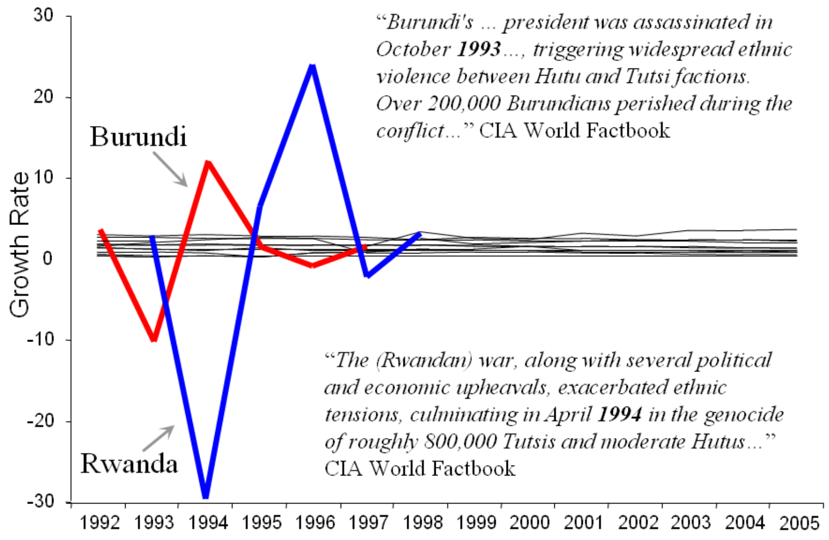

Figure 2.6: The top two 5-year discords discovered in a growth rate database covering 206 counties over 40 years. Ten other counties are shown for contrast (thin lines).

The dramatic differences shown by the discords have intuitive and poignant explanations [2]. The extreme drop in population is clearly understood, but why is it followed in both cases by a spike in growth rate? We conjecture that this corresponds to refugees that fled during the worst of the crises later returning to their homelands. Note that in this case the brute force algorithm would be fast enough to produce these results in reasonable time.

Trajectory dataset

We obtained two trajectory datasets used in [69] and [74] respectively, which have been purposefully created to test anomaly detection in video sequences. The time series are two dimensional (comprised of the x and y coordinates for each data point), and are further normalized to have the same length. In both datasets several deliberately anomalous sequences are created to have a ground truth. The datasets contain 156 [69] and 239 [74] trajectories, with 4 and 2 annotated anomalous sequences respectively. Figure 2.7 shows the number one discord (2D version of the Euclidean distance has been used) found in the dataset of [69]. It is one of the labeled anomalies too.

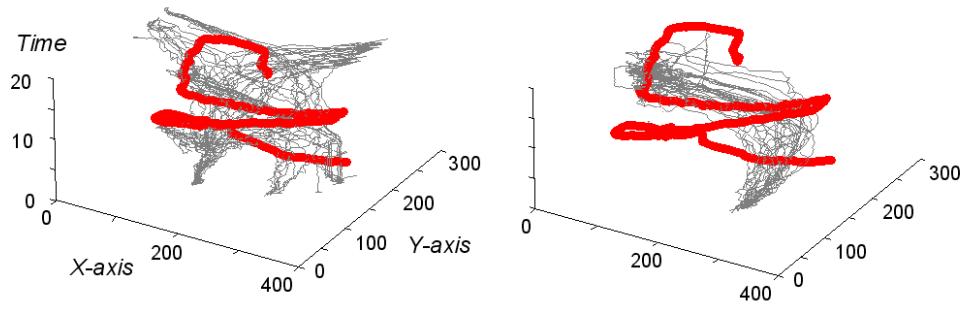

Figure 2.7: Left) The number one discord found in a trajectory data (bold line) with 50 trajectories. It is difficult to see why the discord is singled out unless we cluster all the non-discord trajectories and compare the discord to the clustered sets. Right) When the discord is shown with the clustered trajectories, its unusual behavior becomes apparent (just one cluster is shown here).

On both datasets the discord definition achieves perfect accuracy, as do the original authors. Since all the data can easily fit in main memory our algorithm takes much less than one second. We do not compare efficiency directly with the original works, but note that [69] requires building a SOM, which are generally noted for being lethargic, while [74] is faster, requiring $O(m \log(m) n)$ time, with m being the number of time series and n their dimensionality. Neither algorithm considers the secondary storage case.

Throughout this text we have omitted discussion of determining when a discord is truly anomalous/unusual. We plan to address this issue in a separate work. However, in Figure 2.8 we hint at one possible line of research. The Pokrajac video surveillance dataset [69] is created with two anomalous trajectories (sequences 225 and 237). If we run our algorithm to discover the top sixteen discords, we find that the top two have significantly greater discord distances than all the rest.

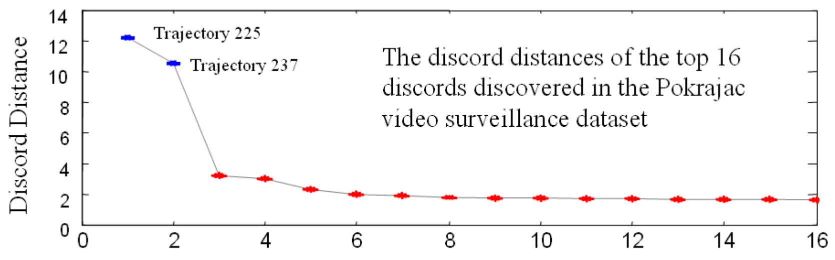

Figure 2.8: The discord distances for the planted anomalies differ notably from the discord distances of the rest 14 top discords. The fact can be used to evaluate the significance of the detected discords.

2.6.2 Scalability of the discord algorithm

We test the scalability of the method on a large heterogeneous dataset of real-world time series and on three synthetically generated datasets of size up to a third of a terabyte. Two aspects of the algorithm were the focus of this evaluation. Firstly, whether the threshold selection criterion from Section 2.5 can be justified empirically (at least for certain underlying distributions) for data of such scale. Secondly, we were interested on how efficient our algorithm is, provided that a good threshold is selected. For both, the synthetic and the real time series datasets, the data are organized in pages of size 10^4 examples each. All pages are stored in text format on an external Seagate FreeAgent hard drive of size 0.5Tb with 7200 RPM and a USB2.0 connection to a computer using Pentium D 3.0 GHz processor. Our implementation of the algorithm loads one page for 5.6 secs.: 0.4 secs. for reading the data and 5.2 secs. for parsing the text matrices. The algorithm was coded in Java.

Random walk dataset

We generated three datasets with random walk time series. The datasets contain 10^6, 10^7 and 10^8 examples respectively. The length of the time series is set to 512 points. Additionally, six non-random walk time series are planted in each of the datasets (see Figure 2.9).

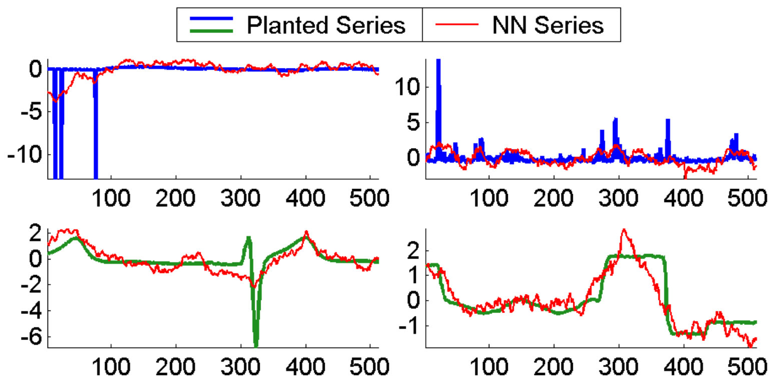

Figure 2.9: Planted non-random walk time series with their nearest neighbors. The top two time series are among the top discords, the bottom two time series fail the range threshold. $|S| = 10^6$.

To compute the threshold a sample of size $|S'| = 10^4$ is used. We set the threshold to the nearest neighbor distance of the tenth discord, hoping to detect some of the planted anomalies among the top ten discords in the entire datasets. Thus it was obtained $r = 21.45$. The time series in the three datasets come from the same distribution and therefore, as mentioned in Section 2.5, the same sample S' (and hence the same threshold r) can be used for all of them. Note that this threshold selection procedure requires less than a minute. After the discord detection algorithm finishes, the set C contains 24 discords for the dataset of size 10^6, 40 discords for the dataset of size 10^7 and 41 discords for the dataset of size 10^8. The running time for the three cases is summarized in Table 2.1.

Table 2.1: *Randomwalk dataset.* Time efficiency of the algorithm.

Examples	Disk Size	I/O time	Total time
1 million	3.57Gb	19min	28min
10 million	35.7Gb	3h 12min	5h 43min
100 million	0.35Tb	31h 18min	66h 17min

In all cases the list C contains the required number of 10 discords, so no restart is necessary. From the planted time series three are among the top 10 discords and for the other three a random walk nearest neighbor is found that is relatively close (see Figure 2.9 for examples). This does not decrease the utility of the discord definition, and is expected as the random walk time series exhibit some extreme properties with respect to the discord detection task, i.e. they cover almost the entire data space that can be occupied by all possible time series of the specified length.

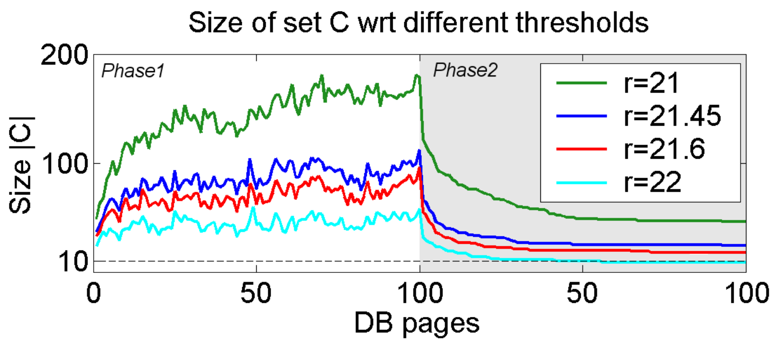

Figure 2.10: *Randomwalk dataset* ($|S| = 10^6$). Number of examples in C after processing each of the 100 pages during the two phases of the algorithm. The method remains stable even if we select a slightly different threshold r during the sampling procedure.

We further note, that the time necessary to find the nearest neighbor for an arbitrary example is 8.5 minutes for the dataset of size one million and approximately 18 hours for the dataset of size 100 million. This means that our algorithm detects the most significant discords in less than four times the time necessary to find the nearest neighbor of a single example only.

Figure 2.10 demonstrates the size $|C|$ after processing each database page. The graphs also show how the size varies when changing the threshold. The plots demonstrate that with a $2\% - 5\%$ change in its values we still detect the required 10 discords with just two scans, while the maximum memory and the running time do not increase drastically. It is interesting to note how quickly the memory drops after the refinement step is initiated. This implies that most of the non-discord elements in the candidates list get eliminated after scanning just a few pages of the database. From this point on the algorithm performs a very limited number of computation to update the nearest neighbor distances for the remaining candidates in C. Similar behavior was observed throughout all datasets studied.

Heterogeneous dataset

Finally we check the efficiency of the discord detection algorithm on a large dataset of real-world time series coming from a mixture of distributions. To generate such dataset we combined three datasets each of size $4\text{x}10^5$ (1.2 million elements in total). The time series have length of 140 points. The three datasets are: motion capture data, EEG recordings of a rat, and meteorological data from the Tropical Atmosphere Ocean project (TAO) [3].

Table 2.2: *Heterogeneous dataset*. Time efficiency of the algorithm.

Examples	Disk Size	Time(Phase1)	Time(Phase2)
1.2 mill.	1.17Gb	8min. 45secs.	9min. 15secs.

Table 2.2 lists the running time of the algorithm on the heterogeneous dataset. Again we are looking for the top 10 discords in the dataset. On the sample the threshold is estimated as $r = 12.86$. After the candidate selection phase the set C contains 690 elements, and at the end of the refinement phase there are 59 elements that meet the threshold r. No restarts of the algorithm were necessary for this dataset either. The discords detected are mostly from the TAO class as its time series exhibit much larger variability compared to the time series for the other two classes.

Parallelization of the discord detection algorithm

Recently there has been an emerging interest in scaling some of the best off-the-shelf machine learning algorithms, through the means of parallelization across grids of multiple computers. A limited number of works target the parallel outlier detection problem too.

For example, Hung et al. [46] introduce a parallel version of the quadratic nested loop algorithm for distance based outliers, discussed in [58]. Lozano et al. [65] propose two algorithms for parallel mining of distance and density based outliers. The distance based algorithm is a parallel modification of Bay's randomized nested loop algorithm [10], and the density based version is a modification of the popular local outlier factor (LOF) algorithm [22]. Both parallel variants proceed in a similar fashion: First the data space is partitioned across different computers and outliers, local for each computer, are identified. Subsequently, the results from all computers are merged within a 'master process' and a global anomaly score is assigned to each outlier.

A common problem with many parallel data mining approaches is that they require implementing specific distributed architectures, as well as distributed indexing structures. The technical overhead of this prohibits the adoption of these methods across the global data mining community. Recently, however, an intuitive yet extremely scalable framework for parallel data mining has emerged, namely the *MapReduce* framework [34]. *MapReduce* is quickly turning into a parallel data mining standard and is already adopted by large companies, such as Google, Yahoo! and Microsoft. The framework operates in two steps. All examples in the data that can be part of the final solution are first *mapped* to some corresponding keys. Then a user specified function is called to *reduce* the key-value pairs to a set that contains only the final answer.

We conclude the efficiency and scalability evaluation of our algorithm by demonstrating that both of its phases can easily fit into the *MapReduce* framework, where the overhead of the parallelization remains relatively small and a nearly linear (in the number of machines used) speed-up is achieved.

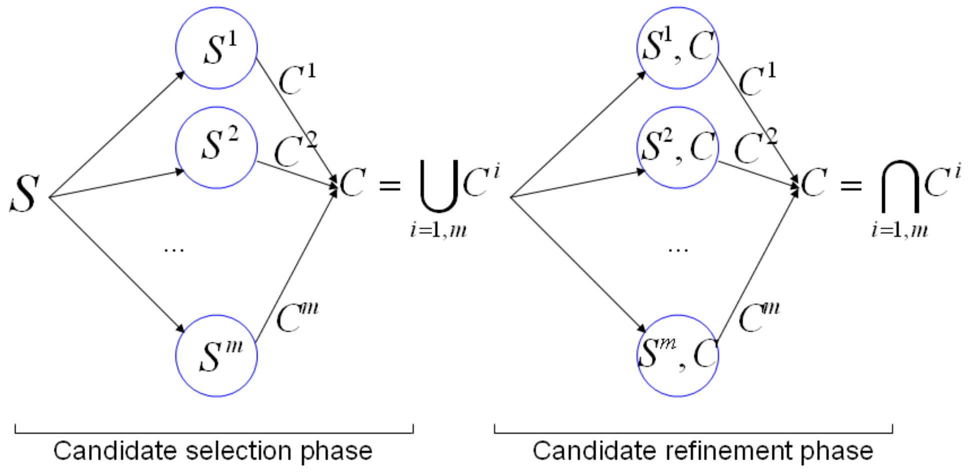
S
S^1
S^2
S^m
C^1
C^2
C^m
$C = \bigcup_{i=1,m} C^i$
S^1, C
S^2, C
S^m, C
$C = \bigcap_{i=1,m} C^i$
Candidate selection phase
Candidate refinement phase

Figure 2.11: Parallelization of the disk aware discord detection algorithm with m computers.

For the experiments, we assume that the input dataset S is split evenly across m computers as shown in Figure 2.11. The candidate selection phase of the discord detection algorithm is then run simultaneously on all computers with input parameters $S = S^i$ and the same threshold parameter r, producing m distinct candidate sets C^i. This concludes the mapping function of the first phase, and the reducing function then simply combines all candidate sets into one $C = \bigcup_{i=1..m} C^i$. Note that when constructing C^i we use only the data from S^i, which means that it might introduce to the union C examples that would be pruned, had we used a single computer that scans the entire dataset S. Therefore, the combined candidate set C for m computers is actually larger than the one that would be obtained with one computer and below we show the overhead introduced by this increased candidate set size. What is essential at this point, though, is the fact that Proposition 2.4.2 remains valid, i.e. no false dismissals are introduced by the parallel modification of the candidate selection phase.

Once the union operation is performed, the combined candidate set C is distributed to all computers and the candidate refinement phase is run simultaneously on all of them with input parameters $S = S^i$, C, and r. This can again be represented by a mapping function that produces refined sets C^i, $i = 1..m$. We now make the observation that the true discords, and only the true discords, with respect to the range parameter r and the entire dataset S, will be present in every refined set C^i. Hence, the final result requires that we *reduce* the refined candidates C^i by performing an intersect among all of them (see Figure 2.11). The final discords are thus given by the set $C = \bigcap_{i=1..m} C^i$.

We simulate the above parallel implementation with $m = 1, 2, 4, 8$ computers each having the same specification as indicated in the beginning of Section 2.6.2. Figure 2.12 demonstrates the running time of the entire algorithm and the candidate set size $|C|$ after the first phase, for the random walk dataset with $|S| = 10^6$ elements. The running time (Figure 2.12, left) is the combined running time for the slowest computer during the first phase, plus the running time for the slowest computer during the second phase. The communication time (broadcasting C^i at the end of each phase, and distributing C at the beginning of the second phase) is not included, yet it is negligible as $|C|$ is much smaller

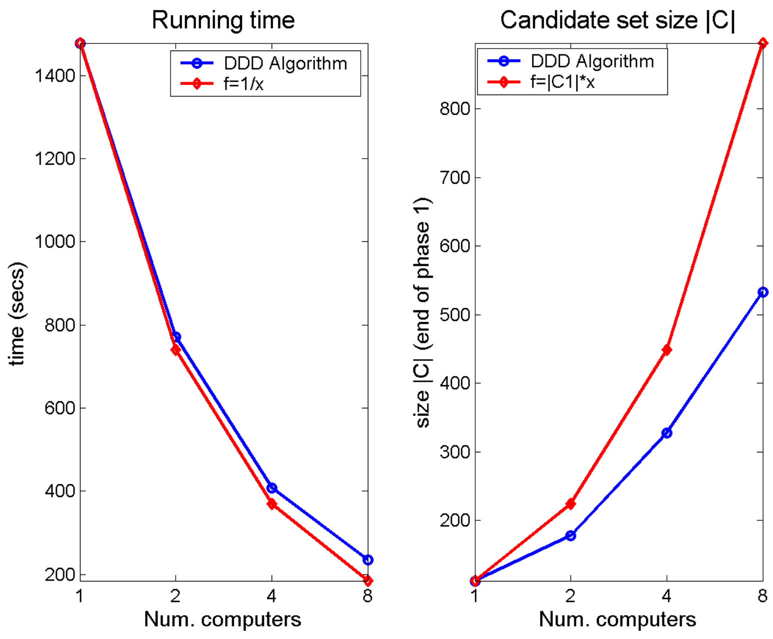

Figure 2.12: *Randomwalk dataset* ($|S| = 10^6$)*. Left:* Running time of the disk aware discord detection algorithm (DDD) with 1, 2, 4 and 8 computers. *Right:* Size of the discord candidates set after the first phase of the algorithm. Refer to the text for details.

than $|S|$ and also we do not need to broadcast the entire time series but only their indices. The union and intersect operations can also be ignored. In our implementation for both of them we used hashing which requires time linear in the size $|C|$.

Figure 2.12 left, shows that with 8 computers the algorithm finishes in 240 seconds (the same threshold $r = 21.45$ was used as in Section 2.6.2). The red curve with diamonds on the plot shows simply a division of the time for the algorithm on one computer by x (i.e. if we had x computers and no parallelization overhead). From the plot it is visible that, for example, with 8 computers this 'perfect' running time would be 180 seconds. This means that the increased candidate set size contributes for approximately 30% running time loss when the discord detection algorithm is run with 8 computers. This is due to the approximately five times larger candidate size, obtained at the end of the first phase (see Figure 2.12 right, the blue curve for 8 computers). Just for comparison we have also plotted the function $f = |\mathrm{C}_1| * x$, where $|\mathrm{C}_1|$ is the candidate set size when using only one computer, to indicate that adding x computers does not necessarily increase the candidate set size x times. Overall the main memory requirement remains admissible for a fairly large number of computers, while the gain in speed-up is enormous. With a single computer, as indicated in Section 2.6.2, the total running time is approximately 28 minutes while now with 8 computers it takes only 4 minutes.

2.7 Discussion

In a sense, the approach taken here may appear surprising. Most data mining algorithms for time series use some approximation of the data, such as DFT, DWT, SVD etc. Previous (main memory) algorithms for finding discords have used SAX [54, 103], or Haar wavelets [40]. However, we are working with just the raw data. It is worth explaining why. Most time series data mining algorithms achieve speed-up with the Gemini framework (or some variation thereof) [37]. The basic idea is to approximate the full dataset in main memory, approximately solve the problem at hand, and then make (hopefully few) accesses to the disk to confirm or adjust the solution. Note that this framework requires one linear scan just to create the main memory approximation, and our algorithm requires a total of two linear scans. So there is at most a factor of two possibility of improvement. However, it is clear that even this cannot be achieved. Even if we assume that some algorithm can be created to approximately solve the problem in main memory. The algorithm must make some access to disk to check the raw data. Because such random accesses are ten times more expensive than sequential accesses [78], if the algorithm must access more that 10% of the data it can no longer be competitive. In fact, it is difficult to see how any algorithm could avoid retrieving 100% of the data in the second phase. For all time series approximations, it is possible that two objects appear arbitrarily close in approximation space, but be arbitrarily far apart in the raw data space. Most data mining algorithms exploit lower bound pruning to find the nearest neighbor, but here upper bounds are required to prune objects that cannot be the furthest nearest neighbor. While there has been some work on providing upper bounds for time series, these bounds tend to be exceptionally weak [99]. Intuitively this makes sense, there are only so many ways two time series can be similar to each other, hence the ability to tightly lower bound. However, there is a much larger space of possible ways that two time series could be different, and an upper bound must somehow capture all of them. In the same vein, it is worth discussing why we do not attempt to index the candidate set C in main memory, to speed up both the phase one and phase two of our algorithm. The answer is simply that it does not improve performance. The many time series indexing algorithms that exist [37, 99] are designed to reduce the number of disk accesses, they have little utility when all the data resides in main memory (as with the candidate set C). For high dimensional time series in main memory it is impossible to beat a linear scan; especially when the linear scan is highly optimized with early abandoning. Furthermore, in phase one of our algorithm every object seen in the disk resident dataset is either added to the candidate set C or causes an object to be ejected from C, this overhead in maintaining the index more than nullifies any possible gain.

2.8 Concluding remarks

The chapter introduced a highly efficient algorithm for mining range discords in massive time series databases. The algorithm performs two linear scans through the database and a limited amount of memory based computations. It is intuitive and very simple to implement. We further demonstrated, that with a suitable sampling technique the method can be adapted to robustly detect the top k discords in the data. The utility of the discord definition combined with the efficiency of the method suggest it as a valuable tool across multiple domains, such as astronomy, surveillance, web mining, etc. Experimental results from all these areas have been demonstrated.

While here we focused on the problem of how to find interesting and excessively noisy examples, we now turn our attention to the opposite task. Namely, in the next chapter we demonstrate a method that tries to identify good, non-noisy examples, which are then used to reconstruct an embedding subspace for the data, if such exists. A major issue in this manifold reconstruction process is how to decouple the the noisy examples from the actual structured signal in the time series datasets.

Note: We would like to thank Dr. M. Vlachos for providing us the web query data, Dr. P. Protopapas for the light-curves, Dr. A. Naftel and Dr. L. Latecki for the trajectory datasets.

Chapter 3

Manifold Clustering of Shapes

The chapter demonstrates that shapes of object have a surprising, yet very effective representation as a special type of *pseudo* time series. The time series representation can be utilized in a number of unsupervised task, with clustering in particular. Shape clustering can significantly facilitate the automatic labeling of objects present in image collections. For example, it could outline the existing groups of pathological cells in a bank of cyto-images; the groups of species on photographs collected from certain aerials; or the groups of objects observed on surveillance scenes from an office building.

Here we demonstrate that a nonlinear projection algorithm such as Isomap can attract together the time series representation of shapes describing similar objects, suggesting the existence of isometry between the time series space and a low dimensional nonlinear embedding. Whenever there is a relatively small amount of noise in the data, the projection forms compact, convex clusters that can easily be learned by a subsequent partitioning scheme. We further propose a modification of the Isomap projection based on the concept of degree-bounded minimum spanning trees. The proposed approach is demonstrated to move apart bridged clusters and to alleviate the effect of noise in the data.

3.1 Introduction

The effectiveness of object recognition and content-based image retrieval systems is highly dependent on the accurate identification of shapes. Features such as color, texture, positioning etc., though important, are insufficient to convey the information that could be obtained through shape analysis [13, 64, 81, 96]. In this chapter we propose an algorithm for clustering of 2D shapes. The method is invariant to basic geometric transformations, e.g. scale, shift, and most importantly, rotation. It is robust to noise, sparsity in the data and outliers that may bridge clusters representing more similar classes.

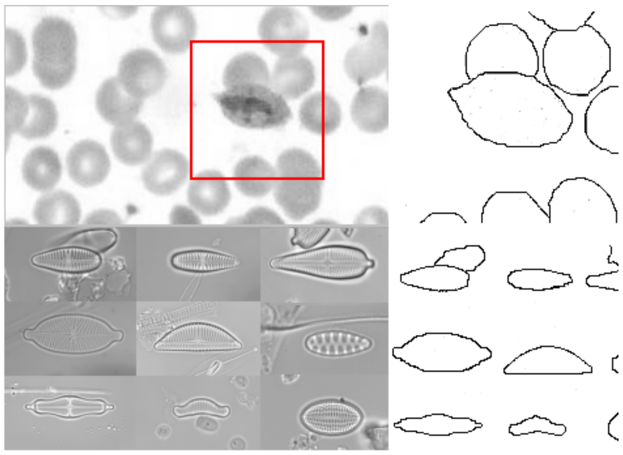

Figure 3.1: Cytology images. *Top*: *Plasmodium ovale* is one of the four malaria agents that can affect humans. The infected blood cells become larger with oval shape. *Bottom*: Diatoms are aquatic eukaryote plants, that appear in multiple shapes. Several types of diatoms can inhabit the same habitat.

The shape clustering problem is of practical importance in many areas where image or video data collections are used. Labeling objects in such collections usually requires manually examining huge volumes of data. Consider for example the field of cytology or the task of video data analysis. For many medical projects large banks of microscope cell images need to be processed (Figure 3.1 top)[1]. The ability to cluster different types of cells (normal cells or cells corresponding to pathologies and diseases) without human supervision could considerably facilitate the medical analysis. Botanists, on the other hand, are interested in detecting and documenting the genotypes populating certain aerials (Figure 3.1 bottom). In tasks such as automatic surveillance or content exploration, the detection of different groups of objects that appear in scene sequences is usually required. Again, for these applications, an unsupervised shape clustering approach would be extremely beneficial.

Constructing a robust clustering algorithm is not trivial, as it should consider certain specifics of the shape data and the intuitively expected outcome. One natural requirement in shape recognition is to detect similarities invariant to basic geometric transformations. For example, in Figure 3.1 top, we would like to distinguish just two classes of cells - a normal and a pathological one, regardless of the many sizes and orientations that elements of each class could have. And while the scale and shift invariance are easily achievable with a suitable representation, the rotational invariance is much harder to deal with [63]. Important factors, that should be noted when dealing with rotational invariance, are how effective and

[1]The malaria images are part of the Hoslink medical databank: http://www.hoslink.com/, and the diatoms images are part of the collection used in the ADIAC project [23]

efficient an algorithm is, as well as what level of control over the admissible rotations it provides. For example, in cytology analysis, we would like to consider all possible rotations when identifying the shape clusters, but in the case of handwritten character recognition we might need to confine the admissible rotations within the interval $[-15\,^\circ; 15\,^\circ]$. Otherwise we would detect as similar shapes that correspond to the digits "6" and "9" or the letters "b" and "q".

Another challenge in the shape clustering task is introduced by the high dimensionality of the input space. Accurate shape representations generally require selecting a large number of features [55]. Additionally, there is significant amount of noise for many of the features, which is either related to the complexity of the studied shapes or is accumulated during certain preprocessing steps as image filtering and edge detection. Therefore, the resulting space is very high dimensional, with a lot of noise and possibly outliers. Clustering in such a space is practically meaningless, so a suitable dimensionality reduction should be applied.

A promising direction towards the outlined problems, relies in the fact that object data usually resides in some nonlinear embedding of the original space, that has a relatively low dimensionality [79, 93]. Nonlinear reduction techniques such as Isomap [93] or Locally Linear Embedding [79] are particulary suitable for projecting such data. Here we focus on the Isomap algorithm and demonstrate that it groups well shapes from equivalent classes, using a very low (two or three) dimensional representation. This suggests that shapes data are also isometric to some nonlinear embedding of the original space. Furthermore, as the classes tend to form compact, convex clusters, they are easy to learn with a subsequent partitioning algorithm, e.g the classical Expectation-Maximization (EM).

However, if different regions have different densities, or if there is considerable amount of noise, Isomap fails to reconstruct correctly the exact structure of the embedding. In sparse regions the embedding becomes disconnected, while in dense regions ”short circuits” are formed between otherwise geodetically distant parts of the embedded surface [9]. As a result, some clusters representing elements from the same class are separated, while clusters representing different classes are often merged.

To project the shape data in cases of noise, bridged or partially overlapping clusters, we introduce the idea of *degree-bounded Isomap*. The algorithm constructs a degree-bounded minimum spanning tree to approximate the underlying embedding. It is demonstrated to move further apart clusters corresponding to more similar classes and to decrease the effect of noise in the data.

The contributions in this chapter can be summarized as:

1. The problem of clustering rotationally invariant shapes is studied and a robust approach for its solution is proposed.

2. An isometry between the shape space and a nonlinear low dimensional embedding is demonstrated, suggesting that nonlinear reduction algorithms should be preferred in learning from shape data for different tasks.

3. The question of Isomap's topological stability is revised and a method is proposed that avoids the problem of having multiple disconnected components in the projection or forming short circuits between geodetically distant regions of the embedding.

The rest of the chapter is organized as follows. In Section 3.2 we briefly review the shape recognition and manifold learning literature. Section 3.3 describes the selected representation and introduces the rotationally invariant metric, used for evaluating the shapes similarity. The proposed manifold clustering approach is described in Section 3.4. Section 4.6 provides an evaluation of the approach on several publicly available datasets. Section 3.6 concludes the discussion and outlines some directions for future research.

3.2 Related work and background

A key factor in the efficiency of shape recognition systems is the selected shape representation. If the representation is not robust to noise, is ambiguous, or does not adapt to geometric transformations, then the clustering quality will be naturally poor. Here we briefly outline the possible shape representation techniques and point out some of their strengths and drawbacks. For more detailed information on the topic, we refer the reader to extensive surveys such as [25, 95, 116].

As outlined by Zhang et al. [116], the representation methods could roughly be divided into contour and region based. Region based methods extract features from the two dimensional image information, e.g. geometric moments, enclosed area or shape covering convex

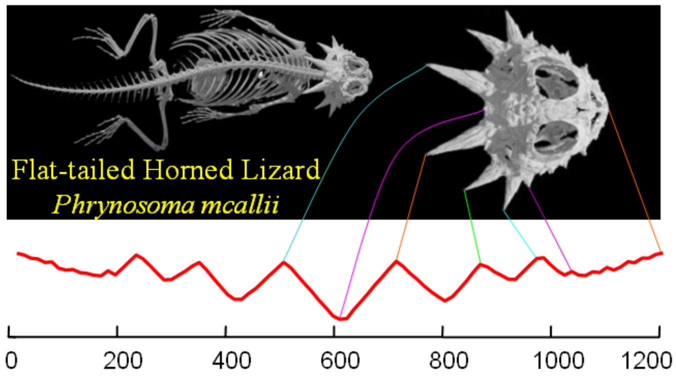

Figure 3.2: Global contour representation. Shapes can be converted to "time series". The distance from every point on the profile to the center is measured and treated as the Y-axis of the time series.

hulls. Some of the region based methods are computationally intensive and often require tracing the contour as well, so that better accuracy could be achieved. Others, such as the moment invariants, are not so robust to distortions and might be ambiguous if the shapes have more complex boundaries.

Here we adopt a *global* contour representation, in which the entire contour is converted to a 1D time series (see Figure 3.2, also Section 3.3). The representation is shift invariant, and by resampling all time series to the same length, or by using a warping metric to compare them, one could achieve invariance with respect to scale too.

Contour based representations in general construct a feature vector using only the points from the shape boundary. To obtain better efficiency, certain contour methods extract a very limited number of features that are either rotation invariant [72], or allow a corresponding alignment [17]. Both of the approaches, while suitable for particular settings, do not have good discriminative ability in the presence of noise and distortions [55, 116, 117]. For example, the alignment approach (also called *landmarking*) often uses the two principle axes of a shape to determine the features. This however is prone to ambiguities, as shapes from different classes may turn out to have similar axes [55] (see Figure 3.3).

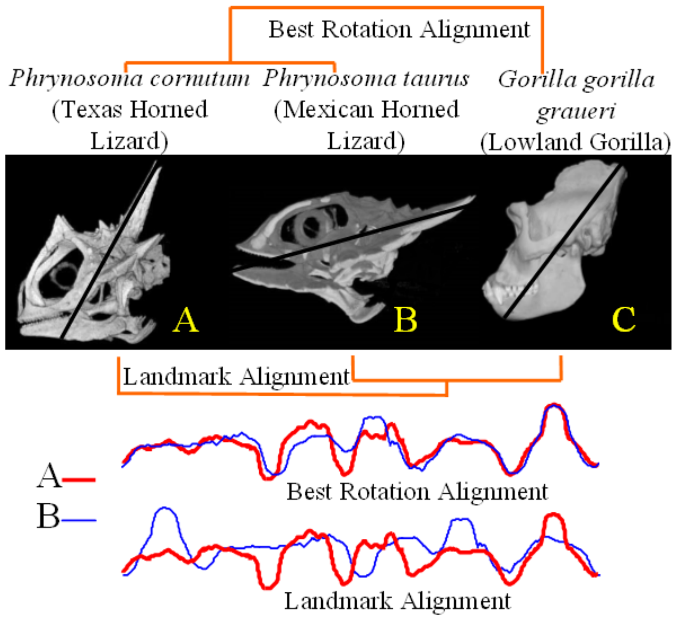
Best Rotation Alignment
Phrynosoma cornutum (Texas Horned Lizard)
Phrynosoma taurus (Mexican Horned Lizard)
Gorilla gorilla graueri (Lowland Gorilla)
A
B
C
Landmark Alignment
A
B
Best Rotation Alignment
Landmark Alignment

Figure 3.3: *Top*: Two skulls of lizards from the same genus, and a primate skull are hierarchically clustered using both the landmark rotation beginning at the major axis, and the best rotation. *Bottom*: The landmark-based alignment of **A** and **B** demonstrates why the landmark-based clustering is unsuitable: a small amount of rotation error results in a large difference in the distance measure.

To resolve the problem, one should rather consider all possible rotations of the compared representations [4, 101], which renders a computationally intensive method. A potential way to remedy the problem is to consider the spectral information of the extracted time series by applying a Fourier transformation [30, 57, 97]. Charalampidis [30] and Klassen et al. [57] further utilize the transformation in partitioning and hierarchical shape clustering schemes. They demonstrate accuracy in performance, for cases when all rotations need to be considered. As we pointed out, however, we would like the approach to give us control over which rotations are admissible and which should be excluded.

Another drawback of a more complex representation as the global contour one, apart of its computational complexity, is that many of the features might be irrelevant or noisy. To decrease the detrimental effect of such features, a suitable dimensionality reduction should be applied. Manifold approaches have been demonstrated to be particularly suitable for projecting image extracted data [21, 79, 88, 93]. In their clustering approach Srivastava et al. [89], also observe the manifold structure of the shape data. The authors implicitly assume a 2D structure for the embedding and build a Markov model to partition the reconstructed 2D surface. Instead, we allow for a nonlinear reduction algorithm to automatically detect the best dimensionality for projecting the space. In particular, we focus on the Isomap [93] algorithm and demonstrate that clustering on the Isomap projection significantly outperforms clustering on the linearly projected data.

3.3 Measuring shape similarity

Formally, a shape representation technique transforms the shape space S into the vector space V through a particular mapping function ϕ:

$$\phi(\mathbf{s}_i) = \mathbf{v}_i \in V, \forall \mathbf{s}_i \in S$$

where ϕ constructs an ordered set of n descriptive features: $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n})$ [116]. With this ordering requirements in hand, $\mathbf{v}_i$ can be treated as a *pseudo* times eries. Indeed, all features $v_{i,j}$ can be assumed to be observation evenly recorded in time, which makes our representation comply with Definition 1.2.1. The size n of the vectors depends on how many distinct features are necessary for the technique to describe the shapes in S. As we pointed out, many existing techniques target lower dimensionality in V in order to obtain better computational efficiency. The downside, however, is poor discriminative ability observed in multiple domains.

Here we adopt a global contour based representation, where every dimension $v_{i,j}$ corresponds to a point on the shape contour as illustrated in Figure 3.2. More precisely, here ϕ is the function that maps every contour point $s_{i,j}$ to the distance between this point and the shape's mass center. This representation is known as *centroid-based* approach and has been introduced by Chang et al. [26]. The space V now consists of all time series extracted from shapes with the above mapping. The time series are further standardized to have mean zero and standard deviation one. The dimensionality of V is usually rather high, but we will demonstrate that a suitable nonlinear reduction in that space can preserve accurately the pairwise element similarities. It is interesting to note that, if several highly descriptive features do exist for a particular dataset, which is what landmarking relies upon, they will most likely be identified during the nonlinear reduction process automatically. If however such features are not present or are ambiguous, because of the shapes' complexity or the presence of noise, the nonlinear reduction can still determine a suitable set of representative features on which to project V.

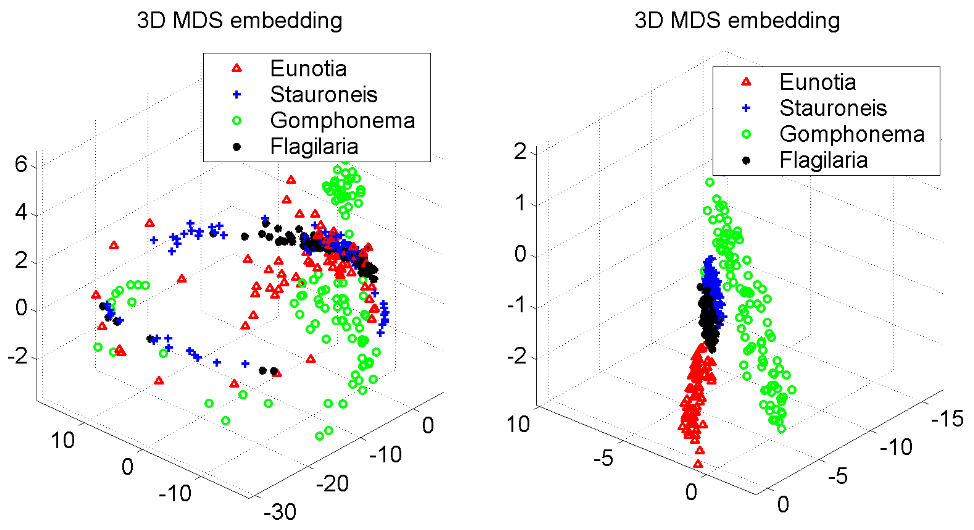

Figure 3.4: MDS projection of the diatoms dataset (Section 3.5.1). *Left*: the Euclidean distance fails to capture the similarities in the presence of rotations. *Right*: Using the rotationally invariant measure rd, elements of the same class are grouped together.

3.3.1 Rotationally invariant distance measure

It is easy to see that due to the centroid-based representation, the selected mapping ϕ is shift invariant. Standardization alone, however, is insufficient to achieve scale invariance. Different object sizes or different image resolutions are likely to result in shape contours of variable lengths too. Scale invariance could be achieved on the representation level by augmenting ϕ with a simple resampling step, during which all extracted time series are resized to the same length n. Another approach is to use a warping distance measure (equation (1.3)) that will find the best alignment between the series and thus compensate for the different lengths. Using a warping distance, however, has been demonstrated to introduce little improvement over the basic resampling step followed by an Euclidean distance estimate [55].

A more difficult challenge is presented by rotation invariance. Even with the resapmling step, ϕ is still unable to capture any possible similarities if rotations are present. As an example, consider Figure 3.4 left, which demonstrates the Multidimensional Scaling (MDS) projection for the diatoms dataset (see Section 3.5.1). Given a matrix of pair-wise distances, MDS tries to compute the coordinates for the data that approximate best the information in the matrix. Once the coordinates are identified, the algorithm performs an eigen decomposition of the data, and projects it along the dimensions determined by the top few eigenvectors.

Here, the distances provided to MDS are the Euclidean distances between the resampled time series. The three dimensional projection was selected as the most accurate for the dataset. All elements have been uniformly randomly rotated, which leads to the spherical form of the projection (or circular if 2D projection was to be selected). The larger the angle of rotation, the further the examples from a single class are projected. At the same time, elements from different classes that should appear distant, are placed close to each other. As expected, the resulting clustering is essentially meaningless.

To approach the problem, it is important to notice that all rotations of a shape $\mathbf{s}_i$ can be approximated by a suitably selected *circular shift* $\mathbf{v}_i^r$ (also called *time series rotation*) of the original vector $\mathbf{v}_i$, where a circular shift is defined as:

$$\mathbf{v}_i^r = (v_{i,r+1}, v_{i,r+2}, \ldots, v_n, v_{i,1} \ldots, v_{i,r}), r \in [0..n-1]$$

Detecting the clusters invariantly to shape rotations requires measuring the pairwise shape distances with respect to all possible rotations. In the vector space V this is equivalent to computing the minimum distance between all possible circular shifts of the two representative vectors:

$$rd(\mathbf{v}_i, \mathbf{v}_j) \quad = \quad \min_{0 \le r \le n-1} d(\mathbf{v}_i^r, \mathbf{v}_j) \tag{3.1}$$

In the following discussion the distance $d(\mathbf{v}_i, \mathbf{v}_j^r)$ is set to be the Euclidean distance between the corresponding vectors. For clarity of the exposition, to discriminate between the rotational distance rd and the distance d, we will refer to the latter as *inner distance*. By computing the minimum only over subintervals $[p, q] \subset [0, n-1]$, we can further restrict the admissible rotations. In this way, for example, we can avoid grouping together the shape representations of the handwritten digits “6” and “9”.

Using the newly introduced rotationally invariant distance, we apply again MDS to project the four class diatom dataset. The result is depicted in Figure 3.4 right. Unlike the non-rotated distance case, now the elements from the same class are grouped together. This demonstrates that a meaningful approach towards rotationally invariant shape clustering should consider the rd distance, rather than a simple application of the distance d. Yet, there is large overlap between some of the projected classes, which will deteriorate the accuracy of an arbitrary clustering scheme. In Section 3.4 we show that a nonlinear dimensionality reduction can mitigate this effect by separating better the projected clusters.

To justify the effectiveness of the adopted shape representation and the rotational invariant distance measure we also used supervision, measuring the accuracy on a number of labeled datasets. Table 3.1 shows the error rate of one-nearest neighbor classification as measured using leaving-one-out evaluation.

Table 3.1: The classification error of the rotationally invariant distance with time series extracted from all contour points. The utilized inner measure is the Euclidean distance.

Name	Classes	Instances	Class. Error
Face	16	2240	3.839%
SwedishLeaf	15	1125	13.33%
Chicken	5	446	19.96%
MixedBag	9	160	4.375%
OSU Leaves	6	442	33.71%
Diatoms	37	781	27.53%
Aircraft	7	210	0.95%
Fish	7	350	11.43%
Yoga	2	3300	4.70%

Recall that Euclidean distance has no parameters, once the time series are extracted. For the *Face* and leaf datasets the (approximate) correct rotation was known. We removed this information by randomly rotating the images. The *MixedBag* dataset is small enough to run the more computationally expensive Chamfer [19] and Hausdorff [71] distance measures. They achieved an error rate of 6.0% and 7.0% respectively (see also [97]), slightly worse than Euclidean distance. Likewise the *Chicken* dataset allows us to compare directly to [68], which used identical experiments to test six different algorithms based on discrete sequences extracted from the shapes. The best of these algorithms had an error rate of 20.5%. For the *Diatom* dataset, the results are competitive with human experts, whose

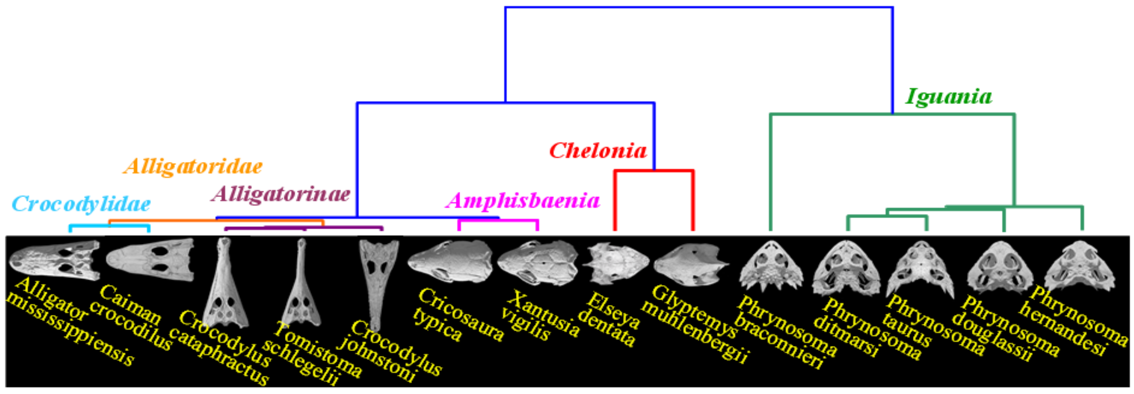

Figure 3.5: Hierarchical clustering of fourteen reptile skulls using the adopted representation.

error rates ranged from 57% to 13.5% [49], and only slightly worse than the Morphological Curvature Scale Spaces (MCSS) approach of [49], which got 26.0%. Note, however, that the MCSS has several parameters to set.

Finally, we also performed extensive "sanity check" experiments using a large database of reptile skulls. We performed a hierarchal clustering, where no dimensionality reduction was used but rather the agglomerative building of the hierarchy was performed in the original time series space. This was done to illustrate that the clusters do not emerge when shrinking the data in the lower dimensional space, but rather exists as reasonable formations in the high dimensional space too. The result was compared with the current consensus on reptilian evolution as suggested by DNA evidence [45, 47]. Figure 3.5 shows a typical example. Two things should be noted: the clustering is subjectively sensible and clearly is rotation invariant. Furthermore, while the global clustering does not perfectly agree with the evolutionary consensus, all the major groups are clustered together as we have annotated in Figure 3.5. In other words, the shape measurements do produce high quality morphological phenograms, although convergent evolution prevents us from obtaining the true global phylogenetic tree from just an examination of skulls.

We now look into a property of the distance measure rd that will be used to allow us for the efficient computation of this measure, and hence for the feasible reconstruction of the underlying embedding, on which the non-linear projection operates.

3.3.2 Metric properties of rd

As pointed out, searching for the most similar shape to a given query in the dataset V can easily become intractable as its size increases. We demonstrate a simple property of the rotation invariant distance that allows one to perform highly efficient best-match searches, regardless of the size of the dataset. Namely, that the rotation invariant distance $rd(v_i, v_j)$ defines a *pseudo-metric* over the space V.

The distance function $d(v_i, v_j)$ is said to be a *metric* over the space V, if for arbitrary elements $v_i, v_j, v_k \in V$ it satisfies the following three properties:

- *Positivity*: $d(v_i, v_j) \geq 0$, with equality iff $v_i = v_j$
- *Symmetry*: $d(v_i, v_j) = d(v_j, v_i)$
- *Triangle inequality*: $d(v_i, v_j) + d(v_k, v_j) \geq d(v_i, v_k)$

When only the second and the third of the above properties are satisfied, $d(v_i, v_j)$ is said to define a pseudo-metric. Showing that a distance function satisfies the triangle inequality is of particular importance when working with large datasets, as it can significantly decrease the searching time by excluding from consideration many of the dataset elements. A number of techniques that utilize the triangle inequality have been proposed over the years, e.g. [24, 41], as well as some popular indexing structures as the Vantage Point trees [115]. Here we show that, provided the inner distance $d(\cdot, \cdot)$ satisfies the triangle inequality, the rotation distance satisfies it too.

Proposition 3.3.1. *If the inner distance $d(v_i, v_j)$ is a pseudo-metric over the space of the shape time series V, then the rotation invariant distance $rd(v_i, v_j)$ also defines a pseudo-metric over V.*

Proof. Without loss of generality, assume that $v_i^{r_0}$ is the rotation of v_i that has a minimal inner distance to v_j, i.e. $rd(v_i, v_j) = d(v_i^{r_0}, v_j)$. Similarly, let $rd(v_k, v_j) = d(v_k^{r_1}, v_j)$ and $rd(v_i, v_k) = d(v_i^{r_2}, v_k)$.

Symmetry: We first note that the alignment $(v_i^{r_0}, v_j)$, where the first time series is rotated, corresponds to an alignment (v_i, v_j^x), where the second time series is rotated. And as d is symmetric we have $rd(v_i, v_j) = d(v_i^{r_0}, v_j) = d(v_i, v_j^x) = d(v_j^x, v_i)$. This, together with formulation (3.1), implies $rd(v_j, v_i) \leq d(v_j^x, v_i) = rd(v_i, v_j)$. Analogously, we can show that $rd(v_i, v_j) \leq rd(v_j, v_i)$. Hence, $rd(v_i, v_j) = rd(v_j, v_i)$.

Triangle inequality: The following holds:

$$\begin{aligned} rd(v_i, v_j) + rd(v_k, v_j) &= d(v_i^{r_0}, v_j) + d(v_k^{r_1}, v_j) \\ &\geq d(v_i^{r_0}, v_k^{r_1}) \geq d(v_i^{r_2}, v_k) = rd(v_i, v_k) \end{aligned}$$

The first inequality above is true as d satisfies the triangle inequality, and the second one follows from the fact that $d(v_i^{r_2}, v_k)$ is the distance between the optimal alignment of v_i and v_k, while $(v_i^{r_0}, v_k^{r_1})$ also corresponds to an alignment between the same time series. $\square$

The symmetry property is important as it allows us to perform only unidirectional computation of the distances between the series, which is essential for the overall efficiency. Requiring d to satisfy the triangle inequality may seem restrictive for the rotation distance. However, some of the distance functions that have been demonstrated to perform best for time series analysis are metrics, e.g. the L_p-norms (equation (1.1)) with the Euclidean distance in particular. Even if the distance function d does not satisfy the triangle inequality, it can still be used as a pruning criterion provided that there exists a lower bounding function LB_D (i.e. $LB\_D(v_i, v_j) \leq d(v_i, v_j), \forall v_i, v_j \in V$) which is a metric. For example, for the dynamic time warping (equation (1.3)), such a metric bounding function has been demonstrated to be the LB_Kim [56] lower bound. In general, the tighter the lower bounding metric that we find, the better the pruning capability of any algorithm utilizing the triangle inequality.

Next we demonstrate how the obtained result can be used for building an efficient best-match searching algorithm for rotation invariant shapes.

3.3.3 Efficient best-match shape searching

This section introduces a scheme for fast rotation invariant best-match searching in the subspace of all shapes S. The speed-up in the scheme results from several levels of pruning different distance computations:

1. *Pruning of rotation distance computations.* The previously derived property allows us to avoid computing a large percentage of the rd distances between the query v_q and the elements of the dataset S.

2. *Pruning of inner distance computations.* As the inner distance d also satisfies the triangle inequality, for every time series v_i that was not pruned on the previous level, only part of the inner distances between v_q and the rotated versions of v_i need to be computed.

3. *Pruning of primitive distance operations.* Using a simple technique, called *early abandon* (to be described later), one can further speed up the inner distance computations that were not pruned in the previous step, by skipping some of the primitive pairwise computations between the scalar elements of the compared time series.

All three levels contribute to the speed-up of the nearest neighbor searches in the rotation invariant space, but it is the pruning of rotation distance computations that becomes of particular importance especially as the dataset size grows very large. While the pruning of inner distance computations and primitive operations still requires that all m dataset time series are retrieved, the pruning of rotation distances makes it feasible for comparing only part of them. This makes the simple result from the previous section extremely important for cases when large amounts of streaming data should be processed or disk retrievals and indexing are required.

Lastly, it is important to note that, as a pruning criterion is applied only when the distance to an element is guaranteed to be larger than some already found distance, the algorithm is guaranteed to make no false dismissals.

Best-match searching algorithm

The proposed scheme is an adaptation of Burkhard-Keller's fast file searching algorithm described in [24]. Here we assume that all rotation distances from the elements of V to a preselected *center point* v_r (see Section 3.3.3) are computed and stored in a sorted list RL. We also precompute the *self-distances* between v_i and its rotations and store them in a sorted n-dimensional vector DL_i, i.e. $DL_i = \{d(v_i^1, v_i^j)\}$, $\forall j \in [1..n]$. Maintaining all m self-distance vectors is necessary for the second level inner distances pruning and increases twice the memory requirement for the proposed scheme compared to the simple brute force search. This linear increase in space complexity is a reasonable and acceptable overhead, as it refers to the compact 1D time series representation rather than the 2D original images. The pseudo-code with a detailed explanation of the rotation invariant searching is presented as Algorithm 3. For clarity of presentation the described algorithm returns only the best-match to a given query and the optimal distance. An extension finding the k most similar time series to the query is straightforward. We have also introduced the notation $\mathbf{V}_i$ to indicate the set of all circular shifts of the vector v_i.

Algorithm 3 Rotation invariant best-match search

```
Preprocessing:
 1: $v_r \in V$ - preselected center
 2: $RL = \{rd(v_r, v_i)\}$ - sorted list, $i \in [1..m]$
 3: $DL_i = \{d(v_i^1, v_i^j)\}$ - sorted lists, $i \in [1..m],\ j \in [1..n]$
Search:
 4: $\forall v_q$: [$bm\_Q$, $Min\_Dist$] = $RI\_Search(V, v_r, RL, rd)$

procedure $[bm, \xi]$=RI_Search(Cnd, C, L, df)
in:  $Cnd$: candidates; $C$: center; $L$: list of distances;
     $df$: distance function
out: $bm$: best-match; $\xi$: minimal distance
 5: $\xi$ = $df(C, v_q)$
 6: $bm = C$
 7: $Cnd = Cnd \setminus C$
 8: while $Cnd \neq \varnothing$ do
 9:     using the sorted $L$, select $v_i \in Cnd$ such that:
          $|df(v_i, C) - df(v_q, C)| \leq |df(v_j, C) - df(v_q, C)|$,
          $\forall v_j \in Cnd, i \neq j$
10:     if $df == rd$ then
11:         $[bm_{fake}, \xi_{tmp}] = RI\_Search(\mathbf{V}_i, v_i^1, DL_i, d)$
12:     else
13:         /* $df$ is a reference to the inner distance $d$ */
14:         $\xi_{tmp} = EarlyAbandon(v_i, v_q, \xi)$
15:     end if
16:     if $\xi_{tmp} < \xi$ then
17:         $\xi = \xi_{tmp}$
            $bm = v_i$
18:     end if
19:     $\forall v_l \in Cnd \ \wedge \ |df(v_l, C) - df(v_q, C)| > \xi$:
          $Cnd = Cnd \setminus v_l$
20: end while
```

For every incoming query the search routine RI_Search is invoked with: a best-match candidates list Cnd initialized as the whole dataset V; the list RL of the precomputed rotation distances from all dataset elements to the center point; and the type of distance function set to rd. The distance function is also used to differentiate between the first and second levels of pruning. RI_Search sets the initial best-match element bm to the center point and the current minimal distance ξ to the distance between the query and the center. The iterative search of the candidates list then proceeds in three steps.

While the list is not empty, a new candidate is selected (line 9), using the heuristic suggested by Burkhard and Keller (described below). If the distance to the new candidate is smaller than the current minimal distance, then the best-match so far is updated to the new candidate (line 17). Finally, the triangle inequality is applied (line 19) to prune all candidates that are guaranteed to be further from the query than ξ. More precisely, as $df = rd$ or d which both satisfy the triangle inequality, the following two inequalities hold: $df(v_q, v_l) + df(v_q, C) \geq df(v_l, C)$ and $df(v_q, v_l) + df(v_l, C) \geq df(v_q, C)$, or in a more compact form $df(v_q, v_l) \geq |df(v_l, C) - df(v_q, C)|$. Therefore, if the difference of the already computed $df(v_l, C)$ and $df(v_q, C)$ is larger than the currently minimal distance ξ, then the distance from the query to the candidate v_l is guaranteed to be also larger than ξ, and there is no need to explicitly compute it.

For the first step, the candidate selection, Burkhard and Keller suggest choosing an element v_i still in the candidates list, for which the difference $|df(v_i, C) - df(v_q, C)|$ is minimal (line 9). Note that this difference is a lower bound for the distance $df(v_q, v_i)$, thus it is likely that by choosing the element with the minimal difference we are also choosing an element that is closer to the final solution. In the experimental evaluation we found out that the heuristic is essential for the pruning capability of the algorithm and its faster convergence to the solution. As the distance list L is sorted, the first candidate can be selected in logarithmic time. Suppose the binary search for a candidate shows that $df(v_i, C) < df(v_q, C) < df(v_{i+1}, C)$, where i corresponds to the position of $df(v_i, C)$ in the sorted list L. This means that the heuristic will return as candidate either v_i or v_{i+1}. On subsequent iterations, one does not need to perform the binary search again but rather select the candidate that is still in the candidates list and whose distance to the center is closest to $df(v_q, C)$ in either direction left (i.e. v_i, or v_{i-1} if v_i was previously selected) or right (i.e. v_{i+1}, or v_{i+2} if v_{i+1} was previously selected).

When the algorithm is invoked with the rotation distance rd as distance function, i.e. we are on the first pruning level, the selected candidate v_i needs to be rotated n times and the inner distances $d(v_i^j, v_q), j \in [1..n]$ need to be computed. We can do this again by applying the RI_Search procedure (line 11, second pruning level), this time with a center v_i^1, sorted list of distances to the center DL_i, and the inner distance d as a distance function. The list of candidates is now composed of every rotation of v_i, which is simply the rotation matrix $\mathbf{V}_i$. When a candidate v_i^j for an inner distance computation is identified, the actual inner distance $d(v_q, v_i^j)$ can be optimized further by computing it with an early abandon technique (line 14, third pruning level).

Early abandon

The early abandoning is a simple, yet extremely efficient technique for speeding up the computations of a distance function. Later on we demonstrate that the running time of the brute force search can be improved with more than a factor of two, by simply modifying it with an early abandon criterion. The method uses a threshold ξ_t and computes the inner distance d by accumulating the primitive pairwise distances as long as the sum is smaller than the threshold. If the threshold is reached, the computation of the inner distance is abandoned.

Algorithm 4 Early abandon for L_p-norms

```
procedure $\xi$=$EarlyAbandon(C, Q, \xi_t)$
in: $C = (c_1, \ldots, c_n)$; $Q = (q_1, \ldots, q_n)$; $\xi_t$: threshold
out: $\xi$: $L_p(C, Q)$ terminated by threshold $\xi_t$
  1: $\xi = 0$; $\xi_t = (\xi_t)^p$; $i = 1$
  2: while $(\xi < \xi_t) \wedge (i \leq size(C))$ do
  3:     $\xi \mathrel{+}= |c_i - q_i|^p$
  4:     $i \mathrel{+}= 1$
  5: end while
  6: $\xi = \sqrt[p]{\xi}$
```

For completeness of the presented searching scheme we list the early abandon method, for the case when the inner distance is an L_p-norm (see Algorithm 4). The algorithm is specific for the distance function used, as different functions might pair different scalar elements. For example in the case of Dynamic Time Warping, c_i might be paired with q_j $(i \neq j)$. Still, an equivalent early abandoning cut-off criterion can be applied to prune some of the paths in the dynamic programming matrix used by the DTW algorithm.

Center selection

The percentage of distance computations that are excluded from consideration, and thus the performance of the algorithm, is highly dependent on the pruning capability of the selected center point v_r. In the original Burkhard-Keller algorithm, the selection is made at random. There are two factors that determine how good v_r is, namely, its position in the space V with respect to the other dataset points, and its position with respect to the queries. A suitable center point will have a small difference $|rd(v_j, v_r) - rd(v_q, v_r)|$ for just a few dataset points v_j. Shapiro [86] argues that good centers can be points which are further from the center of any cluster that might be present in the dataset. This is so, because points close to the cluster centers will be in close proximity to many other points, and for most of those neighbors the above difference will be small. Shapiro suggests an extension of Burkhard-Keller's searching algorithm in which k random centers, rather than one, are used. While this has the potential to mitigate the effect of choosing one inappropriate center, it also comes at the cost of increasing the memory requirements k times, if we would like to apply it for the second level of inner distance pruning too.

In our implementation we still use a single center point, but rather than randomly selecting it, we use subsampling. For the purpose, the preprocessing step (line 1, Algorithm 3) is modified as follows. A small training and validation subsets, are randomly selected from V. The center v_r is set to the point from the training subset that has the best pruning capability for the queries from the validation set. Using subsampling in the center selection process implicitly takes into consideration the specific data distribution, which leads to better pruning ability and smaller variance as compared to random center selection.

The above preprocessing is performed only for the first pruning level. For the second pruning level we always use as centers the original series, i.e. v_i^1. Still, as seen from the efficiency evaluation below, the variance in the performance is very small, which suggests that any rotation v_i^j is an equally suitable center. An intuition of the phenomenon is provided by the observation that for L_p-norms the following equality is true: $d(v_i^{j_1}, v_i^{(j_1+k)\mathbf{mod}(n)}) = d(v_i^{j_2}, v_i^{(j_2+k)\mathbf{mod}(n)})$, $j_1, j_2, k \in [1..n]$. The fact implies that every rotation v_i^j is distributed in the same way among the rest of the rotations of v_i. The small variation in the performance results from the difference in the mutual positions of the query and the different rotations, but on average every rotation will have similar pruning power.

We conclude this section with a brief empirical demonstration of the efficiency of the presented rotation invariant shape search algorithm.

Efficiency of the shape search algorithm

The dataset that we explore represents a collection of arrowheads with various shapes and sizes. Figure 3.17 depicts some representative classes from the data.

We have further augmented the dataset with new images by scaling, deforming and rotating some of the original shapes. The overall size of the resulting dataset is 15000 samples. After extracting the time series from the shapes, we resample them to $n = 250$ time points, which for this dataset seems to preserve the accurate representation. Prior to

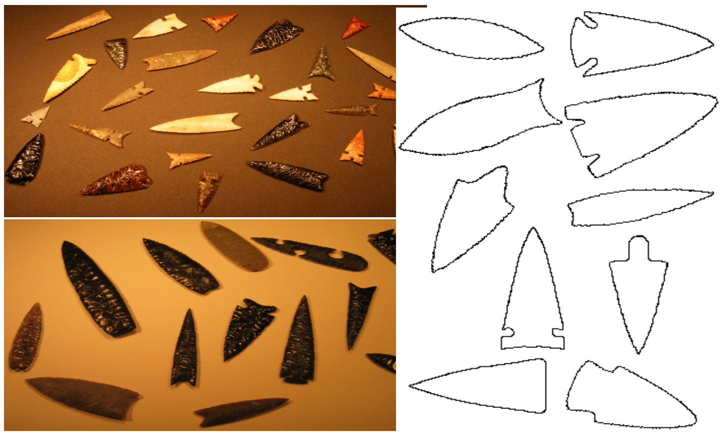

Figure 3.6: *Arrowheads* dataset. *Left*: Representative examples. *Right*: Some of the extracted shapes.

storing, all resampled time series have further been normalized to have a mean zero and standard deviation one. This is done so that the nearest neighbor search could be invariant to transformations, such as shifting or scaling [52]. To measure the performance with respect to the dataset size, we also extract uniformly random samples of sizes from 32 up to 8000 elements from the original dataset.

The percentage improvement of our approach over the *BruteForce* search in terms of performed primitive distance computations is illustrated on Figure 3.7. Results for both applying the method with (RI_Search) and without ($RI\_SearchNoEA$) the *EarlyAbandon* optimization are presented. The performance of simply applying the *EarlyAbandon* technique is also included for comparison.

There are several important aspects to observe in the result. Increasing the space density, i.e. introducing more samples in the dataset, increases the pruning power of the algorithm. The effect is expected as with more elements the chance of finding a sample that is

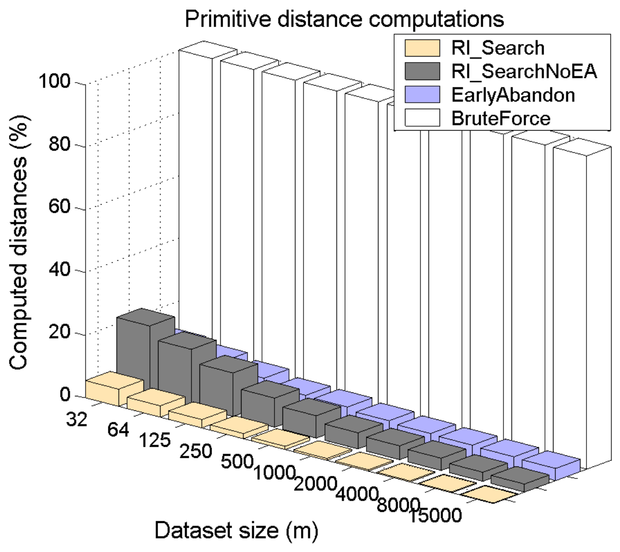

Figure 3.7: *Arrowheads* dataset. Expected percentage of primitive operations to be performed.

very close to the query is higher. Such samples minimize significantly the cut-off threshold ξ, and a lot of the remaining elements start failing the triangle inequality test. The same is true for the *EarlyAbandon* cut-off criterion. Still, the RI_Search algorithm performs far less operations than *EarlyAbandon* - less than half of the operations for the smallest dataset size (5.48$\%$ - RI_Search vs 12.04$\%$ - *EarlyAbandon*), and twenty times less operations for the largest dataset size (0.19$\%$ - RI_Search vs 3.88$\%$ - *EarlyAbandon*). For all dataset sizes of 500 elements and above the RI_Search algorithm performs less than 1$\%$ of the operations performed by the exhaustive *BruteForce* search algorithm.

As previously noted, the time improvement does not correlate exactly to the operations improvement because of language and implementation specifics (see Figure 3.8). Even though *EarlyAbandon* executes less than 10$\%$ of all primitive operations, in our implementation it hardly speeds up the search algorithm more than twice for any value of m. We

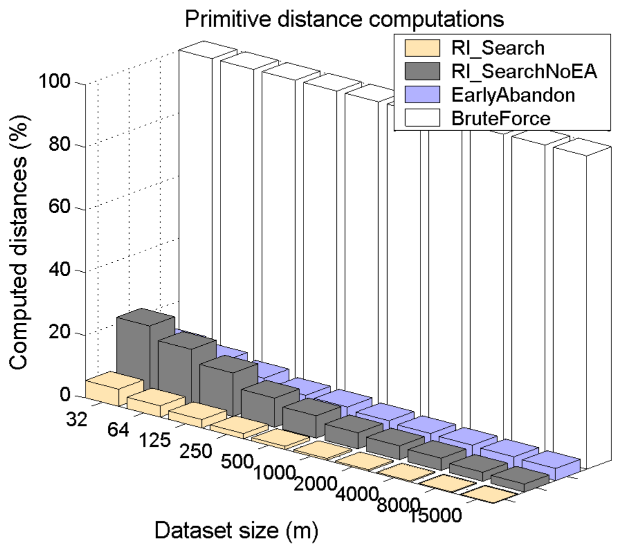

Figure 3.8: *Arrowheads* dataset. Expected running time improvement over *BruteForce* search.

believe this results from the fact that the time for accessing all training samples and their rotations dominates the time for loop computations over array structures as executed by the language. The time improvement for the RI_Search algorithm is also smaller than the operations improvement, which is due mainly to overheads from supporting the sorted candidates and distances lists. Additional, very small slow-down is also caused by the binary search of the first candidate and the traversal of the lists for excluding candidates that fail the triangle inequality. Overall, the proposed algorithm is from four ($m = 32$) to more than fifty ($m = 15000$) times faster than the *BruteForce* search. The graph once again demonstrates that simple bounding techniques, such as *EarlyAbandon*, which need to go through the entire dataset, thought useful, cannot accomplish the enormous improvement that can be achieved by pruning vast parts of the data space (especially for large datasets). We have included the *EarlyAbandon* optimization as part of our algorithm (RI_Search)

mostly for reasons of completeness. Yet, what it introduces as time improvement over the method without it ($RI\_SearchNoEA$) is not that significant, which is partially due to the fact that the distribution for pruned with the triangle inequality time series is different from the original data distribution. For clarity of exposition in the remaining experiments we demonstrate results with the RI_Search algorithm and omit $RI\_SearchNoEA$.

It is important to understand how much each pruning component contributes for the final operations improvement introduced by the algorithm. Fewer computations of the rotation distance imply accessing fewer shapes, which is essential especially when indexing is applied. And fewer inner distances to be considered suggest less memory accesses to different elements, which is also of primary importance. Table 3.2 gives a break-up for RI_Search into levels of pruning.

Table 3.2: *Arrowheads* dataset. Percentage of performed operations. *Row1*: Percentage of computed rotation distances. *Row2*: Percentage of computed distances out of all possible inner distances after level one was performed. *Row3*: Percentage of primitive operations out of all possible remaining operations after level two was performed.

Pruning	Mean(Deviation) of Performed Operations(%)		
Level	$m = 250$	$m = 1000$	$m = 15000$
1-st	52.1(12.3)	34.1(10.0)	22.7(5.81)
2-nd	19.9(0.85)	15.5(0.79)	9.81(0.31)
3-rd	16.0(0.91)	11.4(0.38)	8.61(0.30)

The table illustrates how powerful the triangle inequality is, especially for larger datasets. For example, when $m = 15000$ the algorithm avoids examining almost 80% of the shapes. The second and the third pruning levels are presented with respect to the possible operations after the previous pruning level has been performed. For example, after eliminating some rotation and inner distances, the early abandoning subroutine executes 8.61% of the remaining primitive operations.

Finally, the standard deviation in the performed operations for each pruning level is also presented in the table. The small variance in the second pruning level suggests that all rotated versions of a time series v_i are equivalent in a certain sense, as the rest of the rotated series are similarly distributed around them. Therefore, as discussed in Section 3.3.3, any rotation v_i^j can be considered an equally suitable choice for an inner distance center.

3.4 Manifold clustering of shapes

The previous section gave an efficient algorithm for computing the rotation invariant distances rd. We now return to our original problem of how to use it to infer more accurate clusters in the shape spaces. We have already demonstrated that, when applied to the matrix of rotationally invariant distances, MDS provides a natural choice for a simple reduction of the time series space. Often, however, the data lies on a low dimensional nonlinear embedding (also called manifold), which linear projections cannot identify. The distances measured on the surface of the embedding are called *geodesic* distances. It may turn out that points that have large geodesic distance, and therefore should be treated as dissimilar, are very close in Euclidean sense. Linear projections operate in the Euclidean space and are inadequate to reconstruct the structure, implied by the geodesic distances. As a result, MDS might move apart otherwise similar (with respect to the manifold) elements or bring closer elements that come from different classes (again with respect to the manifold). This effect is the reason for the poor separability between the clusters demonstrated in Figure 3.4 right.

Vision data are often shown to reside on such nonlinear embedding [79, 93]. We demonstrate that shapes data also lie on an embedded space that could be reconstructed with a suitable nonlinear dimensionality reduction technique. In particular, we study the performance of Isomap. After discussing briefly the specifics of the algorithm, we propose a modification for the cases when data are noisy, or when multiple bridging elements between different clusters deteriorate the stability of Isomap's projection.

3.4.1 Dimensionality reduction with Isomap

To improve the chances for a subsequent clustering algorithm to detect any existing clusters, we need to preserve the compactness achieved by the MDS algorithm. For the elements of distinct clusters, however, the distances should be augmented and the clusters should be moved further apart. We obtain the effect by applying the Isomap projection algorithm.

For clarity of discussion a summary of the Isomap's steps, utilizing the rotationally invariant distance, is provided below:

1. Build the distance matrix $M_{m,m}$ for the data as follows: For all elements $v_i, i \in [1..m]$, if v_j is among the k-nearest neighbors to v_i, set $M(i,j) = rd(v_i, v_j)$. Otherwise set $M(i,j) = \inf$.

2. In the graph defined by M, solve the all pairs shortest path problem (e.g. by applying Floyd-Warshall's algorithm). For all elements $v_i, i \in [1..m]$ set $M(i, j) = shortest\_path(v_i, v_j)$

3. Run MDS on M obtained from the previous step.

The first step constructs a k-neighborhood graph as an approximation of the manifold surface and assigns small distances to pairs of elements that are very close on that surface. This is later preserved by the MDS reduction (step 3) in the projected space too. On the contrary, elements from different classes are less likely to be part of each others neighborhoods, and thus will be moved apart in the projection. The second step approximates the actual geodesic distances on the surface of the manifold with the shortest paths in the k-neighborhood graph. The 3D projection of the diatom dataset using Isomap is presented in Figure 3.9. A neighborhood of size $k = 16$, optimal for the projection (see Section 3.5.1), has been used. As seen from the figure, the clusters now are moved further apart, which supports the previous conjecture of an existing isometry between the shape space and a lower dimensional nonlinear embedding.

An important aspect to note is that the goodness of the geodesic distance approximation depends on the right choice of the neighborhood size k. Selecting k larger may result in "short circuits" between distant elements, with respect to the manifold, similarly to the case when only Multidimensional Scaling is applied. In fact, in the asymptotic case when $k \to m$, Isomap is reduced simply to the MDS algorithm. On the other hand, selecting

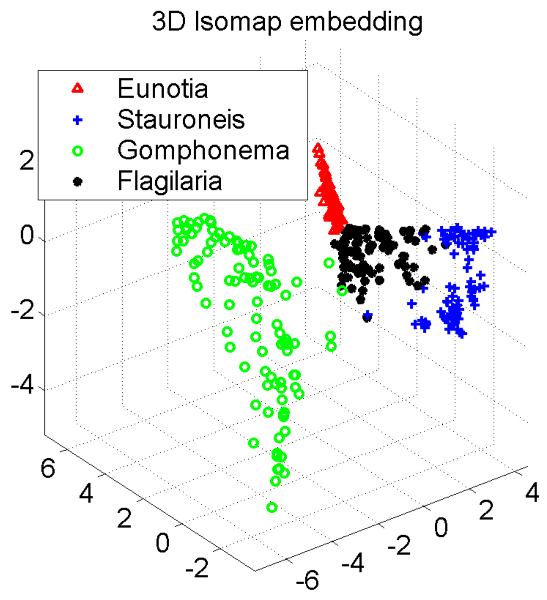

Figure 3.9: Isomap projection of the diatoms dataset. Clusters are better separated suggesting isometry between the shape space and a nonlinear embedding.

k too small may infer multiple disconnected components when building the neighborhood graph. In those cases MDS cannot reconstruct correctly the coordinates of the points. This results in a poor projection and thus in low clustering quality. And finally, depending on the sampling process, it may turn out that there is no one single k that is uniformly best across the whole dataset. For some samples a neighborhood of two elements may be most suitable, while for others, ten neighbors should be preferred. This dependence of the projection quality upon parameter k is referred to as *topological instability* of the Isomap algorithm. The impact, in the case of the shape clustering problem, can be observed in Figure 3.9, where the clusters of *Stauroneis* and *Flagilaria* diatoms are still not separated well, so that a clustering algorithm could discriminate them properly.

3.4.2 Stability of the Isomap projection

Balasubramanian et al. [9] argue that increasing the amount of noise in the data or having a comparatively sparse sample can cause multiple short circuits when Isomap tries to evaluate the correct geodesic distances. Softening the effect by selecting smaller neighborhood size k proves to be a poor solution, as in this case the constructed graph is split into multiple disconnected components. All distances between examples of two disconnected components are set by the algorithm to infinity and thus MDS cannot approximate correctly the coordinates for the elements. As a result, the MDS projection deteriorates significantly.

The solution Tenenbaum suggests [9] is to optimize a tradeoff function between the percentage of elements omitted from the largest connected component and the variance in the distances, as computed on the manifold surface and in the Euclidean projection. Using large number of neighbors will decrease the percentage of omitted elements, but will also lead to improper evaluation of the right dimensionality. Decreasing k will lead to smaller variance, but will increase the percentage of not accounted elements. The globally optimal value of k, with respect to those two criteria, should be selected for the projection.

If, however, regions with different densities exist in the sample, the problem still persists. In denser regions the compromise globally optimal k might again lead to short circuits, while sparse regions will result in disconnected components. Wu et al. [105] suggest a different approach, in which the smallest distance edge between the disconnected components is identified and is added to the k-neighborhood graph. The authors demonstrate that the method is suitable for identifying multiple classes in data, where different classes reside in relatively distant regions on the manifold surface and even on different embeddings. The scheme is generalized by Yang [107], who argues that single edges between disconnected components do not reconstruct smoothly the surface of the manifold. He proposes building an l-connected graph in which for any possible split of the vertices into two groups there exist at least l edges connecting those groups.

Note that all of the above cases still lack flexibility in choosing the right neighborhood size k for individual graph nodes. Ideally we would like a method that defines stronger connectivity in dense regions of the data, but will loosen the requirement for the number of neighbors in sparser regions. Next we suggest one such approach based on degree-k-bounded minimum spanning trees.

3.4.3 Degree-bounded Isomap

The degree-k-bounded minimum spanning tree (k-MST) is an approximation of the MST of a connected graph, in which every vertex is allowed to have degree at most k [73]. The problem has emerged in the context of network modeling, where a network with minimum flow is needed but there is a limit imposed on the capacity of flow that can go through each node.

In the case of Isomap dimensionality reduction, we would like to approximate the k-neighborhood graph with a structure that will ensure connectivity between all vertices. For that purpose, a MST could be constructed. In a MST, however, the local information is not guaranteed to be preserved correctly. Many nodes can be of degree one, while few nodes (especially if residing in dense regions of the data) may end up with some very high degree (e.g. forming stars). The k-MST avoids such undesired effects by restricting the degree of every vertex to be at most k. This also allows for the spanning tree to preserve better the locality around each node approximating the behavior of the k-neighborhood graph. In summary, the k-MST implicitly targets both of the problems outlined in the previous section, i.e. no disconnected components could be produced and there is no globally fixed neighborhood size k for all vertices.

Unfortunately, building the MST structure is a hard problem. In the case of $k = 2$, finding the k-MST is equivalent to the traveling salesman problem, which means that it is NP-complete. It has been demonstrated that constructing 3- and 4-MST is also NP-complete [73]. This may render the manifold representation with a k-MST impractical, yet we are going to approach the problem by making use of the metric properties derived earlier for the rotational distance rd.

Ravi et al. [77] prove that when the distance between the edges of a graph satisfies the triangle inequality, there exists a polynomial time algorithm for building an arbitrary k-MST with total cost at most twice the cost of the MST. We provide an outline of the algorithm below.

1. Build the MST for the data described by the distance matrix $M_{m,m}$ (e.g. we use Prim's algorithm). Select a root node **r** for the tree.

2. Starting from $\mathbf{r}$ do recursively for all non-leaf nodes $\mathbf{v}$: Assume that $(\mathbf{v}, \mathbf{v}_1)$, $(\mathbf{v}, \mathbf{v}_2)$, $\ldots$, $(\mathbf{v}, \mathbf{v}_d)$, are the edges in increasing weight from $\mathbf{v}$ to its children. If $degree(\mathbf{v}) > k$, replace the edges $(\mathbf{v}, \mathbf{v}_2)$, $(\mathbf{v}, \mathbf{v}_3)$, $\ldots$, $(\mathbf{v}, \mathbf{v}_{d-k+2})$ with the edges $(\mathbf{v}_1, \mathbf{v}_2)$, $(\mathbf{v}_1, \mathbf{v}_2)$, $\ldots$, $(\mathbf{v}_{d-k+1}, \mathbf{v}_{d-k+2})$

Step 2 above removes from $\mathbf{v}$ as many edges to child nodes as necessary to keep its degree exactly k. The procedure is repeated recursively for all child nodes too, producing a degree-k-bounded tree. The fact that the cost of the edges is at most two times that of the MST follows from the ordering of the edges and the validity of the triangle inequality. For example, we have $rd(\mathbf{v}_1, \mathbf{v}_2) \leq rd(\mathbf{v}, \mathbf{v}_1) + rd(\mathbf{v}, \mathbf{v}_2) \leq 2rd(\mathbf{v}, \mathbf{v}_2)$, which implies that the cost of every added edge is at most twice the cost of the deleted one.

We will term the Isomap algorithm in which the k-neighborhood graph is replaced with a degree-bounded MST as *b-Isomap* (from bounded Isomap). The b-Isomap projection of the diatoms dataset is presented in Figure 3.10. In this example k has been set to 4.

The figure shows that the *Stauroneis* and *Flagilaria* classes are moved further apart as desired; the classes have less overlapping and just a few bridging elements between the clusters. The clusters are elongated, revealing that most of the elements from a certain class are represented by degree 2 nodes in the k-MST. One negative effect of the projection is that the clusters are not convex as in the case of the Isomap projection. Instead, there might be several elongated branches rooted as a subtree, representing elements from the same class. When multiple such branches exist, there is a higher chance that some of them

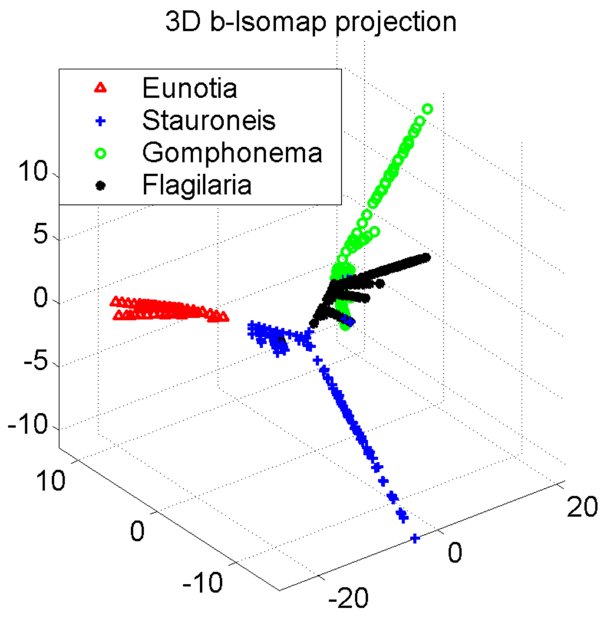

Figure 3.10: b-Isomap projection of the diatoms dataset. Sparser regions are loosely connected, which leads to better separability of bridged clusters such as the *Stauroneis* and *Flagilaria* ones.

will be assigned to different clusters degrading the quality of the clustering.

3.4.4 Shape clustering algorithm

We summarize the proposed clustering of rotationally invariant shapes in an end-to-end algorithm (see Algorithm 5). The algorithm builds on top of the introduced rotationally invariant distance metric rd, and uses a nonlinear projection to discover the inherent dimensionality of the shape data at hand.

The clustering scheme can be used as both unsupervised or semi-supervised algorithm (step 2). In the evaluation we use a semi-supervised approach in which the cluster quality is checked upon the apriory known true labels of the elements. In an unsupervised procedure the mean square error with respect to the cluster centers could be tested instead.

Algorithm 5 Manifold Shape Clustering

```
procedure $[D\_Labels]$ = ShapeClustering($D$, $C$)
in:  $D$: dataset of converted to time series shapes;
     $C$: number of clusters
out: $D\_Labels$: cluster labels

Projection Step:
1: $k$ = Refine_k($D$, $C$);                /*num.neighbors or degree*/
2: Checking the performance on a representative labeled subset choose between:
   Alternative1:  $D'$ = Isomap($D$, $k$);      /*projected data*/
   Alternative2:  $D'$ = b-Isomap($D$, $k$);    /*projected data*/

Clustering Step:
3: $IC$ = Refine_Seeds($D'$, $C$);                  /*initial seeds*/
4: $D\_Labels$ = Cluster_EM($D'$, $IC$, $C$);
```

As seen later from the experimental evaluation, which alternative performs better (Isomap or b-Isomap) depends on a number of things. As a general rule of thumb if no labeled information or other prior knowledge is available, the Isomap projection should be preferred as the more consistent of the two (see Section 4.6). It should also be preferred when the existing classes of shapes are known to be relatively distinct and with small amount of noise. If the existing classes are believed to be comparatively similar (i.e. with large amount of overlap or bridging elements), or there is large amount of noise in the data, then the b-Isomap projection should be applied. The projection parameter k, neighborhood size in the case of Isomap and maximum degree in the case of b-Isomap, can be selected again using additional supervision (if labels are available) and cross-validation (the approach used in our experiments) with subsamples of the data, or by applying the tradeoff optimization criterion discussed by Tenenbaum [9]. We decided to select a partitioning clustering algorithm, and EM in particular, as the clusters defined by Isomap and often by b-Isomap are convex and comparatively compact. The k-means algorithm in this setting is likely to fail due to the elongated structure of the clusters, while a k-medoid approach will have lower efficiency. The initial centers for the EM algorithm are selected as the best random seeds out of 10 runs again on subsamples of the data. An alternative approach is discussed by Fayyad et al. [38], which draws a set of very small subsamples and evaluates the centers that maximize the likelihood of the data based on those subsamples.

3.5 Experimental evaluation

We test the performance of the two manifold approaches and the MDS projection on three publicly available datasets - diatoms, marine creatures and arrowheads. The datasets are selected to have different characteristics in terms of noise, sparsity in the data and similarity between the available classes. The actual labels of the samples are known and are used in measuring the accuracy of clustering.

The following evaluation procedure has been applied for all methods. A 10 times random sampling is used with 80% random subsamples from the original dataset. For each subsample, after the data is projected with the corresponding method, an EM clustering is performed. As EM relies on the correct initial center selection, it is repeated 10 times, each time with randomly selected centers. The accuracy from the best of the 10 clusterings is reported as accuracy of the method for this subsample.

3.5.1 Diatoms dataset

Diatoms are eukaryote plants that live in aquatic environment. The dataset we use is collected as part of the ADIAC project [23]. It contains approximately 360 images of diatoms from four classes - *Eunotia*, *Stauroneis*, *Gomphonema*, *Flagilaria* (see Figure 3.11). All time series for the dataset are resampled to a length of 345 points.

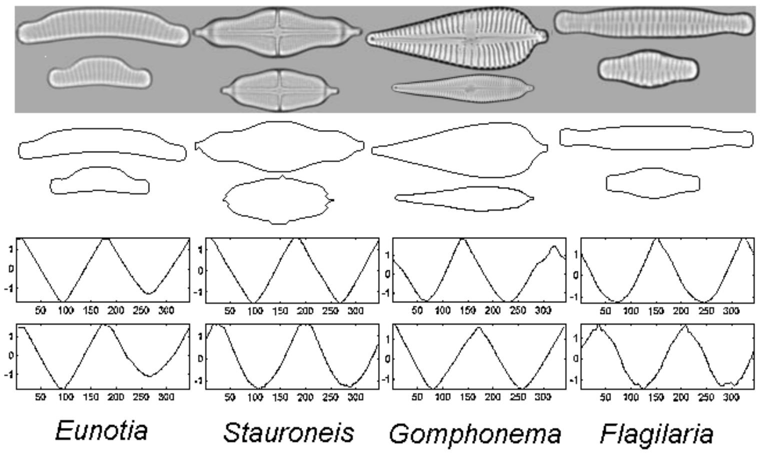

Figure 3.11: Diatoms dataset: original images - top, extracted shapes - middle, and time series representation - bottom. The four classes are relatively distinct with small similarities between some *Stauroneis* and *Flagilaria* diatoms.

To determine the number of dimensions that should be used in the projection, we measure the residual variance for any of the reduction methods as suggested by Tenenbaum et al. [93] (see Figure 3.12).

The ”elbow” of the curve indicates the dimension beyond which adding new dimensions does not increase significantly the variance in the data, and thus no improvement in the projection can be expected. In the case of Isomap and b-Isomap, the variance also de-

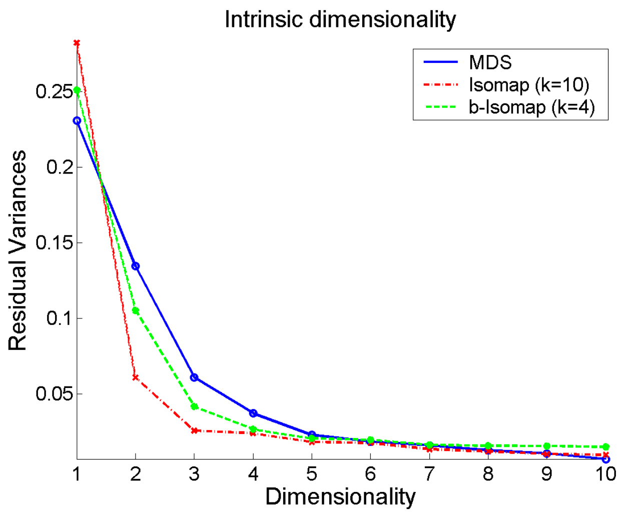

Figure 3.12: Detecting the intrinsic dimensionality of the data according to the three projections. The "elbow" of the curve points to the optimal number of dimensions to be used.

pends on the number of neighbors or the bounding degree parameter, still the structure of the curves remains similar for other values of the parameter too. The other datasets tested in the evaluation produced residual variance curves that differ in the speed with which they decay, but overall the best dimensions remain the same. Therefore, for all the datasets we tested the clustering accuracy, considering the 2D and 3D projections obtained by the three methods. The fact that two or three dimensions are descriptive for the data is not surprising, given the chosen representation. The time series usually have several extreme points, corresponding to those contour points that are closest/furthest from the shape centroid. It is the extreme points (global or in some case local extrema) that are usually detected as the most discriminative dimensions for the data.

Table 3.3 summarizes the details for the best accuracy obtained on the four class diatom dataset. Both nonlinear projections outperformed MDS with more than 20%. The best per-

Table 3.3: Clustering accuracy for the four class diatoms data.

Proj. Method	Dime-nsions	Parameter k	Average Accuracy (%)	Std (%)
MDS	3	N/A	62.3	5.2
Isomap	3	16	**86.2**	3.0
b-Isomap	3	4	83.0	3.6

formance was obtained with the Isomap algorithm using three dimensional projection. The best number of neighbors for Isomap is relatively high (16), which implies that there is little noise and overlapping between the clusters (except for the *Stauroneis* and *Flagilaria* classes). The b-Isomap reduction performed slightly worse on average, and was also less consistent across the subsamples, which is represented by the larger variance in the accuracy (3.6%, column 5). An illustration of why b-Isomap's projection was outperformed is presented for the two dimensional projection in Figure 3.13.

The figure compares the true labels (left graphs) with the labels as identified by the EM algorithm (right graphs). The elipses drawn around each cluster have radii equal to twice the standard deviation along the corresponding dimension. Some of the *Stauroneis* and *Gomphonema* "branches" in the b-Isomap projection are incorrectly identified by EM to be part of the distribution for the *Flagilaria* class. The effect is not that strong for the Isomap projection because of the convex shape of the clusters.

We also compared the clustering accuracy between the two most overlapping classes, in which we additionally added to the time series Gaussian noise with mean zero and standard deviation 0.1.

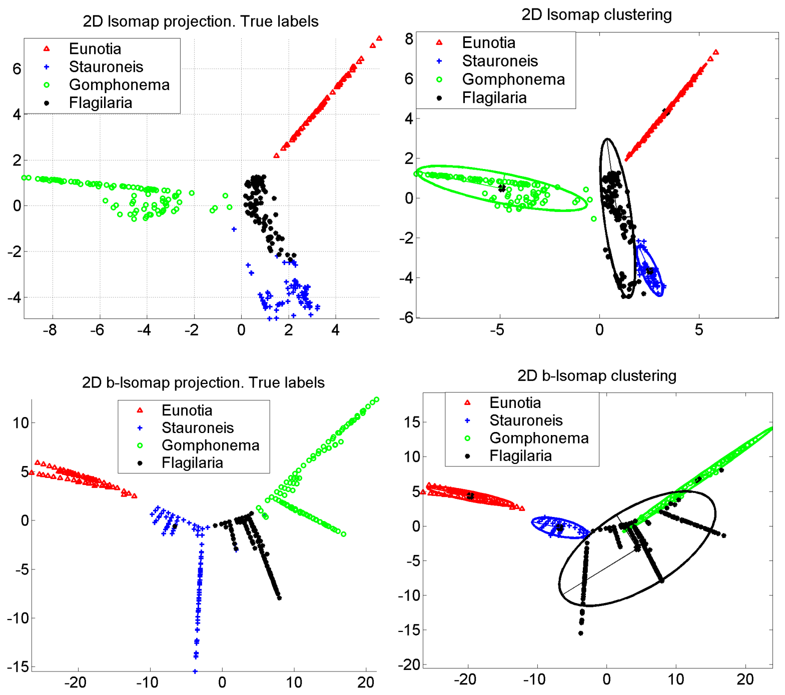

Figure 3.13: Clustering obtained from the 2D projections of Isomap (top) and b-Isomap (bottom). On the left are the true labels for the data, and on the right - the labels as computed by the EM algorithm. Note that the best projection is three dimensional, here two dimensions are shown for illustration only.

The two dimensional projection in this case and the EM clustering are shown in Figure 3.14. The clusters produced by b-Isomap now have higher density, compared to the Isomap clusters, and are easier to detect with the EM algorithm. The sparsity in the Isomap clusters results from the multiple short circuits between the two similar classes. The clustering obtained with b-Isomap is almost perfect when three dimensional projection is used Table 3.4. Isomap performs better in three dimensions too (the best 2D accuracy for the algorithm is 89%), but still it is dominated by b-Isomap's performance.

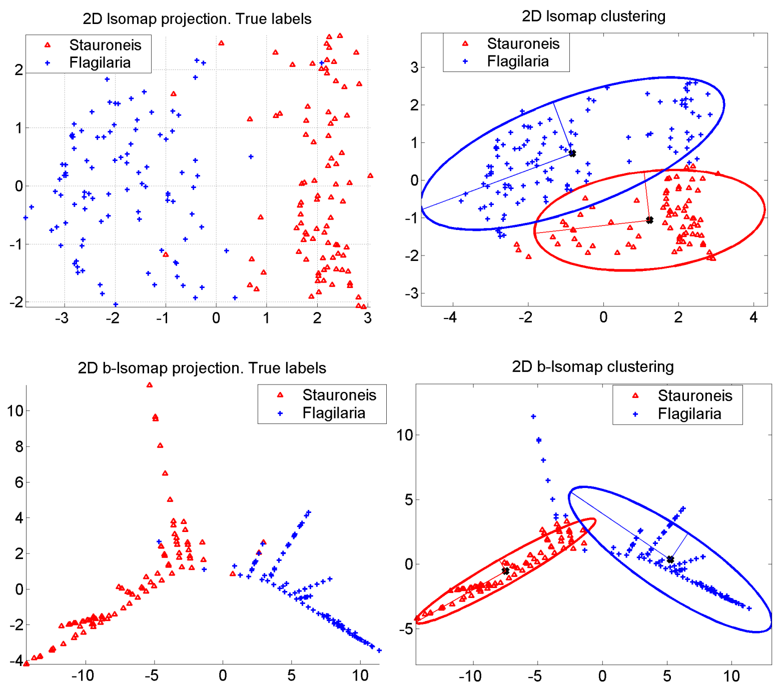

Figure 3.14: Clustering obtained from the 2D projections of Isomap (top) and b-Isomap (bottom) of the *Stauroneis* and *Flagilaria* classes only. On the left are the true labels for the data, and on the right - the labels as computed by the EM algorithm.

Table 3.4: Clustering accuracy for the two class diatom data.

Proj. Method	Dime- nsions	Parameter k	Average Accuracy (%)	Std (%)
MDS	3	N/A	90.2	1.3
Isomap	3	5	92.7	1.3
b-Isomap	3	3	**98.3**	0.9

The example demonstrates that significant improvement over Isomap can be achieved with the b-Isomap approach in the case of noise and when there is no strong distinction between the existing classes.

3.5.2 Marine creatures dataset

We used the prototype database of marine creatures discussed by Mokhtarian et al. [67]. The images for four classes of different types of fish were selected, with each class containing 50 examples (Figure 3.15).

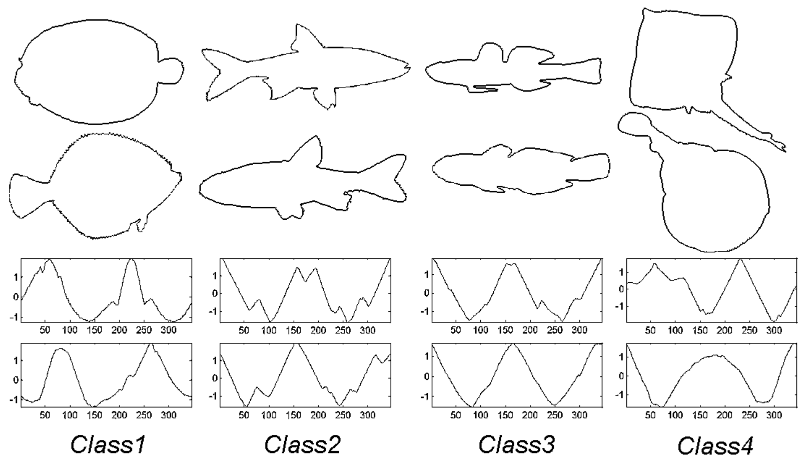

Figure 3.15: Marine creatures dataset: fish shapes - top, their time series representation - bottom.

The time series extracted from the shape contours are again resampled to 345 time points (see Figure 3.15, bottom). For this dataset there is significant amount of within-class variability too. The contour of the shapes has more complex structure than that of the diatoms, which is reflected in the representation too. The time series contain more noise and there is no strong visual distinction between some elements from different classes. For example, *Class1* appears visually similar to *Class4*, while some elements of *Class2* are similar to elements of *Class3*. This similarity is a prerequisite for the formation of bridging elements between the projected clusters. In this sense, the dataset is similar to the two class diatom case. As expected, in this setting the b-Isomap projection is better than the one obtained with Isomap (Figure 3.16).

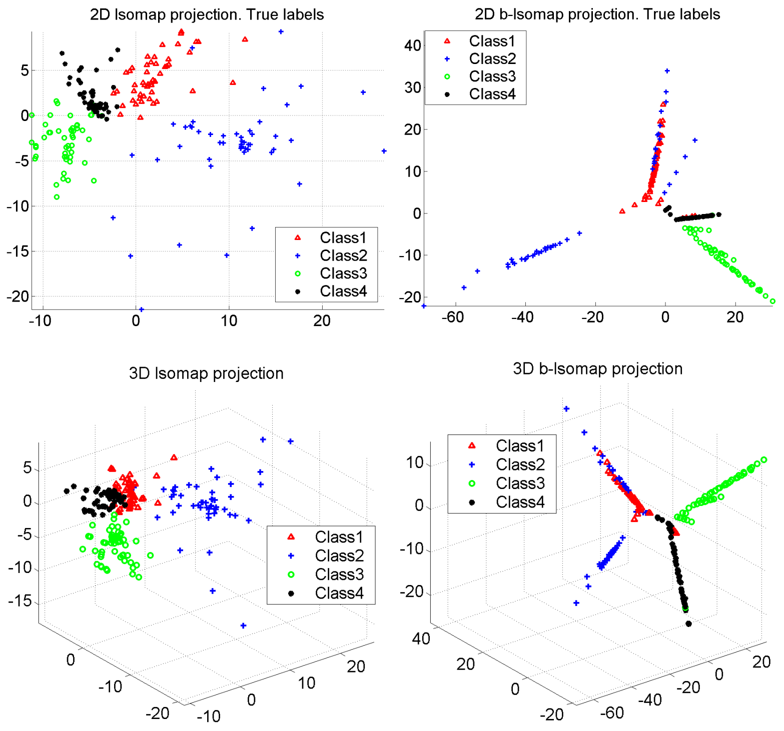

Figure 3.16: Marine creatures dataset: Isomap projection (left) compared to b-Isomap projection (right).

On average, clustering with b-Isomap is 2%-3% more accurate than clustering on the Isomap projection (Table 3.5). Again, the EM algorithm, applied on any of the nonlinear projections, significantly outperforms the clustering on the MDS projection. Yet, it is worth noting the larger variance of the nonlinear projections and especially of Isomap across the ten subsamples. This is partially due to the smaller number of examples (approximately 40 examples from each class are present in the subsamples), and partially to the larger amount of noise in the data.

Table 3.5: Clustering accuracy for the Marine creatures dataset.

Proj. Method	Dime-nsions	Parameter k	Average Accuracy (%)	Std (%)
MDS	2	N/A	61.0	3.0
Isomap	3	4	77.6	11.8
b-Isomap	3	4	**80.0**	7.8

3.5.3 Arrowheads dataset

The arrowheads dataset was described earlier in this chapter. For the current clustering evaluation we use a uniformly random subset of it containing 600 images with randomly rotated arrowheads. The arrowheads are representative of 6 distinct classes (Figure 3.17), with each class of 100 elements. All time series have been resampled to 250 time points.

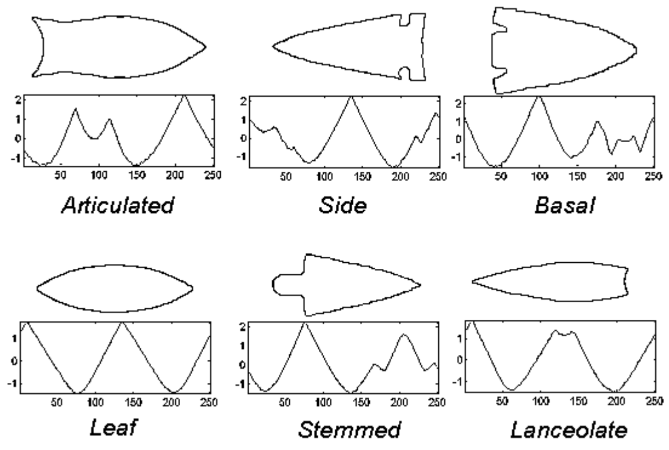

Figure 3.17: Arrowheads dataset: representative examples of the six classes and the corresponding time series representation.

The purpose of this evaluation was to test the behavior of the projections and the clustering algorithm when there is larger number of classes. Figure 3.18 demonstrates the 2D and 3D projections of Isomap and b-Isomap for the data.

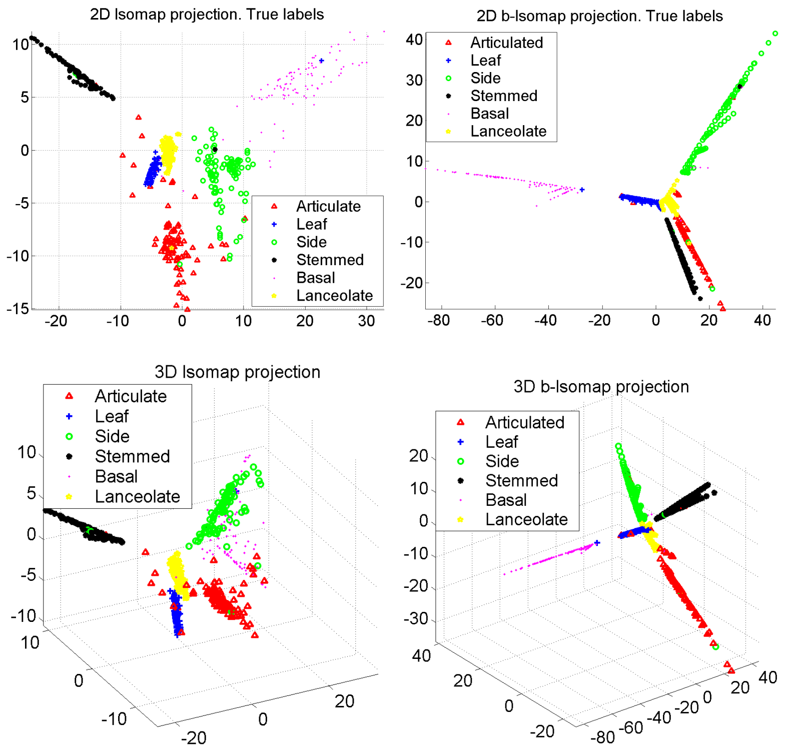

Figure 3.18: Arrowheads dataset: Isomap projection (left) compared to b-Isomap (right).

The performance of the three projections is summarized in Table 3.6. As the classes are distinct, and there is enough data from each class in the subsamples, Isomap reconstructs well the embedded structure and projects the classes in well defined sufficiently distant clusters. The two dimensional b-Isomap projection with bounding parameter $k = 6$ performed similarly well (85.1$\%$ accuracy). This is a result of the convexity of the clusters for this dataset. For most classes the degree-bounded spanning tree forms single long branches, which allows for all examples to be subsequently identified as coming from the same cluster. The b-Isomap clustering was also more consistent for the dataset, with twice smaller deviation as compared to Isomap. Both approaches again outperformed linear MDS.

Table 3.6: Clustering accuracy for the arrowheads dataset.

Proj. Method	Dime-nsions	Parameter k	Average Accuracy (%)	Std (%)
MDS	3	N/A	75.6	5.7
Isomap	3	14	**85.2**	6.2
b-Isomap	2	6	**85.1**	3.1

3.6 Concluding remarks

We presented a method for clustering shape data invariantly of basic geometric transformations as shifting, scaling and most importantly rotation. The results demonstrate that an Isomap projection built on top of a rotationally invariant distance metric can detect correctly the intrinsic nonlinear embedding in which the shape examples reside. We have further introduced a modification of the Isomap algorithm, based on the concept of degree-bounded minimum spanning trees, that decreases the effect of bridging elements and noise in the data.

A possible improvement of the algorithm would include an adaptive solution, which based on the data space density alternates between the more suitable reconstruction Isomap or b-Isomap. A weighted ensemble of the two algorithms might also be preferred for certain manifolds, rather than a simple selection between them. In the next chapter we introduce such an adaptive reconstruction technique, which however, combines not individual algorithms but instead two conceptually different views of the data - a global and a local one.

Chapter 4

Locally Constrained Support Vector Clustering

The chapter develops a dual treatment of the time series manifold reconstruction problem, combining a global density view of the data with multiple local views from a mixture model. The principles here are built upon the theoretical foundations of one-class classification and its derivative - support vector clustering. Support vector clustering transforms the data into a high dimensional feature space, where a decision function is computed. In the original space, the function outlines the boundaries of higher density regions, naturally splitting the the data into individual clusters. The method, however, though theoretically sound, has certain drawbacks which make it not so appealing to the practitioner. Namely, it is unstable in the presence of outliers and it is hard to control the number of clusters that it identifies. Parametrizing the algorithm incorrectly in noisy settings, can either disguise some objectively present clusters in the data, or can identify a large number of small and nonintuitive clusters.

Here, we explore the properties of the data in small regions building a mixture of factor analyzers. The obtained information is used to regularize the complexity of the outlined cluster boundaries, by assigning suitable weighting to each example. The approach is demonstrated to be less susceptible to noise and to outline better interpretable clusters than support vector clustering alone.

4.1 Introduction

One-class support vector machine (SVM) is an efficient approach for estimating the density of a population [82, 85]. It works by applying a transformation $\Phi : X \rightarrow \Phi(X)$ from the input space to a high dimensional feature space, such that points with denser neighborhoods are projected further from the origin of the coordinate system. The support vectors in the feature space are then used to outline closed contours around the dense regions in the input space, defining a binary decision function which is positive inside the contours and negative elsewhere (see Figure 4.1). The method has been demonstrated to be applicable for tasks, such as novelty and fault detection, context change detection, learning in image retrieval, etc.

One can easily extend one-class classification to a clustering scheme, by labeling the elements within each closed contour as different clusters. All elements, that are not enclosed by any contour, correspond to regions that are estimated to have lower density support in the high dimensional feature space. Such elements can be assigned the label of their closest contour in the original space (see Figure 4.1). This extension, called support vector clustering (SVC), was proposed by Ben-Hur et al. [14].

Despite its theoretical soundness the SVC method has remained relatively unpopular among the practitioners. There are several specific characteristics of SVC that diminish its appeal. For instance, the map Φ requires a parametrized kernel to be provided as an input from the user. The radial basis function $k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$ has been recognized as a preferred kernel function because of its ability to form closed contours [14, 92]. This means that the user needs to provide a suitable kernel width γ. Small values of γ (i.e. large kernel width) may disguise or merge some of the clusters, while very large γ may create a large number of closed contours which could outline some rather nonintuitive clusters (see Figure 4.1). This lack of control over the number of identified clusters is exacerbated by the fact that there is no clear objective criterion for comparing the clustering results when varying γ. For example, partitioning schemes, such as K-means optimize an objective function of the distance between all elements and the center of their corresponding clusters. However, in the case of non-convex clusters, which is where the SVC method could be of practical importance, a cluster's mean can be far outside its boundaries, which makes the

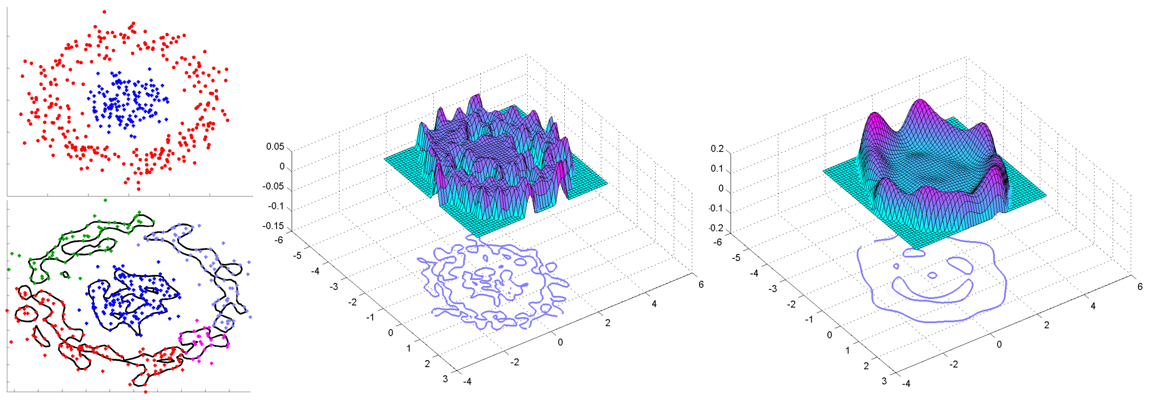

Figure 4.1: *Top left*: the original data. Support vector clustering computes a function in the feature space which is positive for the dense regions and negative elsewhere (*Middle* and *Right* figures). Small kernel widths create multiple complex contours (*Middle*), while large widths may disguise some true dense regions (*Right*). In the original space the contours naturally outline the clusters in the data (*Bottom left*). Points outside the contours are assigned to the closest cluster.

partitioning objective function inapplicable.

The effect of multiple emerging clusters can be especially strong in the presence of noise. This becomes an issue, in many practical application where the examples lie near the surface of a lower dimensional nonlinear manifold. For example, such noisy manifolds may be defined by a sample of facial images [79, 80, 93], or by the walking motions of a human [59]. Though a soft margin can be introduced to alleviate the impact of the outliers, there is again the issue of how to specify the correct parameter ν, that controls the tradeoff between the generalization performance of the learner and its tolerance to the noisy examples. Furthermore, even with an informed selection of the two parameters ν and γ, the learned decision function could still have a very large capacity, resulting in rather complex contours in the input space (Figure 4.1). Note that this is not an issue with one-class SVM as the purpose there is simply to identify regions with high density support. The problem emerges when we try to label the existing clusters, because for more complex contours it is hard to identify whether two examples x_i and x_j are enclosed by the same contour (see Section 4.3.2). As a result, a lot more clusters than are actually present are detected. Some approaches which improve the cluster labeling have been proposed [61] but the problems still persist when multiple dense regions are identified suggesting the presence of multiple clusters. Suppose also that a user wants to specify the number of clusters that they would like to be detected in their data (a natural requirement, handled easily in partitioning and agglomerative clustering schemes). A reasonable way to proceed would be to start merging the multiple contours according to their closeness to each other. Yet, due to the complex boundaries, rather nonintuitive clustering might be produced failing to capture the topology followed by the data.

To improve the performance of SVC in the case of Gaussian distributed noise and to obtain better control over the number of detected clusters, we explore the density variability of the data in very small regions. For the purpose, a Mixture of Factor Analyzers (MFA) [43] is used (see Section 4.4.1). The mixture model, when learned with large number of analyzers, implicitly detects points that deviate from the main trajectory of the data. The information about those locally deviating points is used to determine the soft margin tradeoff between the outliers and the accuracy of the one-class SVM learner, as well as, to regularize the complexity of the induced decision boundary. The latter is achieved by weighting the penalty term, imposed on the learner for mislabeling the outliers. The penalty now is set to be proportional to the distance of the outliers to the center of their factor analyzer. The regularization results in smoother contours, which are shrunk towards the dense regions in the data, rather than trying to accommodate all outliers. The subsequent clustering often allows for easier interpretation too. Because of the local dimensionality reduction performed by MFA and the nonlinear feature map Φ, the "locally constrained" SVC method is further demonstrated to correctly identify the topological structure of the data, when the clusters reside on a lower dimensional nonlinear manifold.

The rest of this chapter is organized as follows. Section 4.2 covers several related to the proposed method approaches. Section 4.3 gives a brief introduction to SVC and demonstrates some of its advantages and shortcomings. The constrained SVC approach, proposed here, is studied in Section 4.4. A discussion and evaluation of the method on synthetic and real data are presented in Section 4.5 and Section 4.6. Section 4.8 concludes with some open problems and possible future extensions.

4.2 Related work and background

Here we look into several popular directions in unsupervised learning when the data lie on or near a nonlinear manifold, and try to put the proposed approach in perspective with those directions.

A number of clustering algorithms have been demonstrated to be particularly suitable for learning of non-convex formations, e.g. spectral clustering [70], spectral graph partitioning [39], or kernel K-means [84]. A close relation between all of these approaches has been pointed out before [20]. We focus on one of these algorithms - spectral clustering. Interestingly, the algorithm shares a lot of commonalities with SVC. They both start by computing a Gaussian kernel matrix, emulating the high dimensional nonlinear feature map Φ. From here on, however, spectral clustering performs an eigen decomposition of the data in the feature space. The projected examples are then clustered, again in the feature space, using K-means clustering. This is also closer in spirit to the kernel PCA algorithm [83, 84]. Instead, SVC computes the optimal plane that separates the projected data from the origin in the feature space. In this way a simpler problem is solved by only isolating the higher density regions. This comes at the price of not knowing the actual clusters in the data, so a subsequent labeling and assignment step is carried out by SVC.

As mentioned previously, a significant disadvantage of the SVC method is that it requires as an input the width for the kernel function to be used. Spectral clustering, which also requires such a parameter while building its affinity matrix, resolves this problem by comparing the inter-cluster variability for the clusters identified by K-means in the feature space. The width that leads to smallest such variability is selected.

A different set of unsupervised learning approaches try to infer the nonlinear structure of the data by considering small regions around each example. Some popular methods following this paradigm are, for example, the Laplacian eigenmaps [11] and Isomap [93]. The general idea behind these algorithms is to compute a neighborhood graph G, where each example x_i is connected only to examples in its close proximity. The graph is then augmented to a full affinity matrix, by propagating the neighboring distances, e.g. by solving an all pairs shortest path problem (Isomap) or by applying a Laplacian operator (Laplacian eigenmaps). Both methods proceed by computing an eigen decomposition and projecting the data using a small subset of the eigenvectors. As they preserve the convexity of the data, the algorithms can easily be extended for clustering by using a partitioning scheme as K-means or Expectation Maximization (EM) for a mixture of Gaussians. While local reduction methods have been demonstrated to be unstable in the presence of noise [9], they remain to be the preferred tool for unsupervised learning from nonlinear manifolds.

In the proposed approach we combine the best features that can be obtained from global methods, such as SVC and local approximations as the ones discussed above. The underlying idea is that a global view of the data can be inferred by looking at the overall density distribution. The density estimate alone, however, provides for a very coarse reconstruction of the underlying sample space. Local methods, on the other hand, can smoothen this estimate by looking at the data statistics in some small regions. This is especially important if density fluctuations are observed in the data and yet an obvious clustering is available. In this sense, the proposed method is closest in spirit to the manifold reconstruction method proposed by Roweis et al. [80]. They use a mixture of factor analyzers to infer the local structure of the underlying manifold, but then a global constraint is imposed, so that all local models are aligned to follow a consistent trajectory.

4.3 Support vector density estimation

4.3.1 One-class classification

Let us have a set of n independent and identically distributed observations: $X = \{x_i\}_{i=1}^n$. The problem addressed by one-class classification is to find a minimal region R, which encloses the data (Figure 4.2). Assuming that the data are generated from the same distribution p, an additional to the minimization of R is the requirement that future test examples generated by p should also fall with high probability within R. Therefore, apart of being minimal, R should also generalize well on unseen data, which implies that it should be enclosed within a smoother decision boundary.

Following similar reasoning as in support vector classification, rather than exploring the nonlinear boundary in the original space, one could describe it as a hyper plane in the high dimensional feature space defined by $\Phi(X)$. All examples, which in the original space are enclosed within R, are going to be projected in the same half-space with respect to the hyper plane. If $\mathbf{w}\cdot\Phi(x) = b$ is the equation of the plane, this is equivalent to the requirement that for all examples x_i, the inequality $\mathbf{w} \cdot \Phi(x_i) \geq b$ should hold. The two parameters that define the plane uniquely, $\mathbf{w}$ and b, are its normal vector and its displacement from the origin respectively. Finally, the plane that corresponds to the smoothest boundary in the original space is the one with smallest norm of the normal vector $\mathbf{w}$ [83]. This suggests that the hyper-plane that defines the smallest region with smoothest boundary in the original space is a solution to the optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & \mathbf{w} \cdot \Phi(x_i) \geq b,\ i = 1..n \end{aligned} \tag{4.1}$$

It may be useful to restrict R to enclose only a subregion of X that has certain support μ for the probability density function p, i.e. $\int p(R)dR \geq \mu$, for some $\mu \in (0, 1]$. This will be the case, for example, if we are not interested in the noisy points on the periphery of the distribution (see Figure 4.2). In the feature space, the points that fall outside of R will satisfy $\mathbf{w} \cdot \Phi(x_i) < b$. To account for such points the constraints for them in (4.1) should be changed to $\mathbf{w} \cdot \Phi(x_i) \geq b - \xi_i$, where we have additionally introduced the *slack variables* $\xi_i \geq 0$. The regularization term that guarantees the smoothness of the boundary also changes, yielding the new formulation:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & q(\mathbf{w}, b, \xi) = \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{1}{n\nu}\textstyle\sum_{i=1}^{n} \xi_i - b \\ \text{subject to} \quad & \mathbf{w} \cdot \Phi(x_i) \geq b - \xi_i,\ \xi_i \geq 0,\ i = 1..n \end{aligned} \tag{4.2}$$

Formulations (4.1) and (4.2) produce the so called *hard* and *soft margin* decision planes respectively. The penalty parameter ν in (4.2) controls the tradeoff between the allowed slack for some of the examples and the complexity of the region boundary. It takes values in the interval $(0, 1]$ with $\nu \to 1$ allowing for a lot of examples to lie outside the region R,

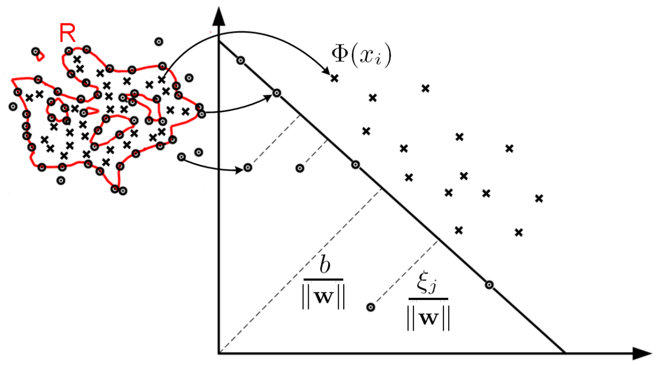

Figure 4.2: One-class SVMs detect a region R in the data with higher density support. Points inside the region are projected in the same half-space defined by the separation hyper plane $(\mathbf{w}, b)$.

and $\nu \rightarrow 0$ penalizing significantly the slack variables, converting the problem effectively into a hard margin decision problem. The latter case leads to a very tight and complex boundary for the density region R.

Minimizing the quadratic function $q(\mathbf{w}, b, \xi)$ in problem (4.2) is hard, because of the available constraints. Instead, if we write all constraints in the form $q_i(\mathbf{w}, b, \xi) \leq 0$, the solution is obtained by minimizing the Lagrangian $L(\mathbf{w}, b, \xi, \alpha) = q(\mathbf{w}, b, \xi) + \sum_i \alpha_i q_i(\mathbf{w}, b, \xi)$. To minimize L, one sets the derivatives of L with respect of $\mathbf{w}$, b and ξ to zero, which allows for expressing them as a function solely of the introduced Lagrangian multipliers α_i ($\alpha_i \geq 0,\ \sum_i \alpha_i = 1$) and the data in the feature space $\Phi(x_i)$. Substituting the values back in the Lagrangian, we obtain the dual optimization problem of problem (4.2):

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\textstyle\sum_{ij} \alpha_i \alpha_j \Phi(x_i)\cdot\Phi(x_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le \tfrac{1}{n\nu} \\ & \textstyle\sum_{i=1}^{n} \alpha_i = 1 \end{aligned} \tag{4.3}$$

The class of feature mappings $\Phi(X)$ that linearly separate the data from the origin is not available in parametric form, yet it is selected so that the dot products in the feature space correspond to a computable kernel function in the input space, i.e. $k(x_i, x_j) = \Phi(x_i)\cdot\Phi(x_j)$. In SVC the Gaussinan kernel $k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$ is used, as it defines smooth closed contours [14, 92]. All multipliers $\alpha_i > 0$ in the solution of (4.3) correspond to the support vectors, i.e. the examples which in the feature space lie on the separating hyper-plane (see Figure 4.2). For the rest of the points x_i the corresponding α_i is equal to zero. To test on which side of the hyper plane such examples are projected, one needs to substitute them in the equation of the plane as defined by the computed support vectors:

$$f(x) = sgn[\sum_{x_i \in SVs} \alpha_i k(x_i, x) - b] \tag{4.4}$$

Positive $f(x)$ implies that x falls within the dense subspace R, whereas negative values of the decision function imply a sparsely populated region. Observation x_i from the input set X for which $f(x_i) < 0$ are called *bounded support vectors*. The value of the displacement b can be computed using the fact that any support vector x_s lies on the separation plane, and thus it satisfies the equality $\mathbf{w} \cdot \Phi(x_s) = b$, which can also be expressed in terms of the kernel function as $\sum_{x_i \in SVs} \alpha_i k(x_i, x_s) = b$.

The formalization defined so far is not the only way for computing high dimensional density support. For instance, rather than looking for the optimal separation hyper plane, Ben-Hur et al. [14] study the class of spheres in the feature space that enclose the projected examples. They derive an alternative formulation of problem (4.3), which instead minimizes the volume of the enclosing hyper sphere. An equivalence of the two formulations has been demonstrated by Schölkopf and Smola in [83]. Here, the density estimation step is carried out as in the original one-class SVM formulation.

4.3.2 Support vector clustering

The one-class density estimation method can easily be extended to a clustering scheme by computing a matrix A for the data, where $A_{ij} = 1$ if x_i and x_j are enclosed within the same contour and 0 otherwise. Whether x_i and x_j lie within the same contour can be determined by computing the SVM decision function (4.4) for all points on the line that connects them. In the original SVC formulation (and also in our implementation) 20 regularly spaced points between x_i and x_j are tested. An always positive decision function guarantees that x_i and x_j are part of the same dense region. The opposite, however, is not necessarily true. For some points, on the line between two examples, f may negative, but the examples may still be within the same contour. This is often the case if the contours

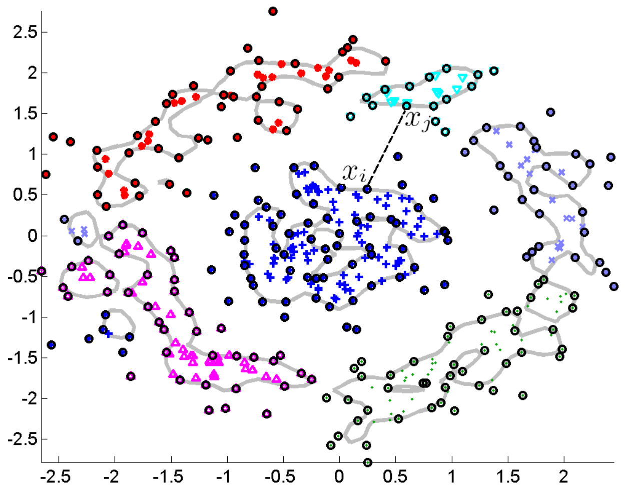

Figure 4.3: One-class SVMs can be extended to a clustering scheme, by assigning the same label to all points enclosed within the same contour. For example, x_i and x_j are within the same contour if for any point x on the line between them the decision function $f(x)$ is non-negative.

are too complex. Therefore one needs to detect the connected components in the graph induced by A. This determines the number of clusters in the data as well as the labels for each example that is enclosed by a contour. Finally, the bounded support vectors (i.e. the examples outside the contours) are assigned to their closest cluster (see Figure 4.3).

While precise parametrization is not so essential when only density estimation is required, it becomes of crucial importance in the case of clustering. Consider, for example, Figure 4.3. Selecting a large kernel width (i.e. small γ) would disguise the fact that there is large fluctuation between the density of the inner and the outer circles. Large values of γ or too small tradeoff terms ν, on the other hand, can produce decision boundary of a very high capacity, which leads to multiple tight contours in the original space. Apart of obtaining too many small and nondescriptive clusters, the complex decision function impedes the proper labeling even of elements that are within the same contour. For some examples x_p all lines connecting them to other examples x_q within the same contour, would pass through regions where the decision function has negative value. Such examples will be assumed to belong to a different cluster.

The lack of control over the number of clusters produced by different parametrizations is a significant drawback of the scheme. A common requirement in clustering is that the users provide the number of clusters that they want to be detected in their data. Such a requirement is easily handled by partition clustering (e.g. K-means), agglomerative clustering and even kernel based algorithms as spectral clustering. Unfortunately there is no clear unsupervised strategy of how such user imposed constraint can be incorporated in SVC. One reasonable way to emulate such behavior, would be to start exploring kernels with monotonically decreasing widths until at least as many clusters as users require emerge from the data. Such iterative approach is followed for example in [62]. As will be shown in the experimental evaluation, this strategy, though pretty robust in the case of well separated and dense clusters, can cause the occurrence of some rather uninformative formations when the clusters are sparse and noise is present in the data.

Next, we introduce a modification of the SVC approach, which improves on its stability in the presence of noise. The method is further demonstrated to be less sensitive to slight changes in the parametrization.

4.4 Locally constrained SVC

A leading observation in here is that both global density estimation methods as SVC, and local reconstruction methods as Isomap [93] or LLE [79] introduce some information about the data, which is somewhat complementary. For example, support vector clustering provides some very important information about the overall structure of the data. Namely, an estimate of its density. A local method can complement this with additional region boundary smoothing and can evaluate locally which points are likely to deviate from the unknown distribution that has generated the data. The method that we utilize here to obtain such local statistics is based on the Mixture of Factor Analyzers framework introduced by Ghahramani et al. in [43]. We term the algorithm derived in this section locally constraint support vector clustering (LSVC).

4.4.1 Mixture of factor analyzers

Factor analysis (FA) is a technique for linearly projecting the data $X \subset R^D$ into a lower dimensional space R^d [36]. Ghahramani et al. [43] derive an EM procedure for learning the projecting dimensions **z**. They make the simplifying assumption that the dimensions **z** are normally distributed with zero means and variance one, i.e. $\mathbf{z} \in \mathcal{N}(0, I)$ (I here marks the identity matrix). Furthermore, each example is also allowed to have some residual noise **u**, which is also assumed to be normally distributed with covariance Ψ, i.e. $\mathbf{u} \in \mathcal{N}(0, \Psi)$. To summarize, the following relation is enforced:

$$\mathbf{x} = \Lambda \mathbf{z} + \mathbf{u} \tag{4.5}$$

where Λ is the so called *factor loading matrix*. In (4.5) the noise covariance matrix Ψ is required to be diagonal. The *common factors* **z** are used as latent variables to iteratively obtain an improved likelihood estimate for the observed data **x** (E-step of the algorithm), recomputing on each iterations more optimal values for the matrices Ψ and Λ (M-step of the algorithm).

Ghahramani et al. [43] also suggest that one could have a mixture of factor analyzers, rather than a single one, where every component in the mixture can have different mean μ_j and loading matrix Λ_j. The noise term in the mixture is preserved the same across all factor analyzers, i.e. $\mathbf{z}_j \in \mathcal{N}(\mu_j, \Psi)$. The goal now becomes to find a maximum likelihood estimate for the observed data **x**, using the latent variables $\mathbf{z}_j$, and the probability that it has been projected using the j-th factor analyzer (E-step of the mixture model). On every iteration the MFA algorithm, apart of computing some more optimal estimates of the matrices Ψ and Λ_j, also improves on the estimate for the mean of the analyzers μ_j too (M-step of the mixture model).

Figure 4.4 illustrates the MFA algorithm when applied with twenty components. As a clustering approach MFA resembles significantly the simple EM for a mixture of Gaussians, and often produces clusters of similar characteristics. Apart of clustering the data,

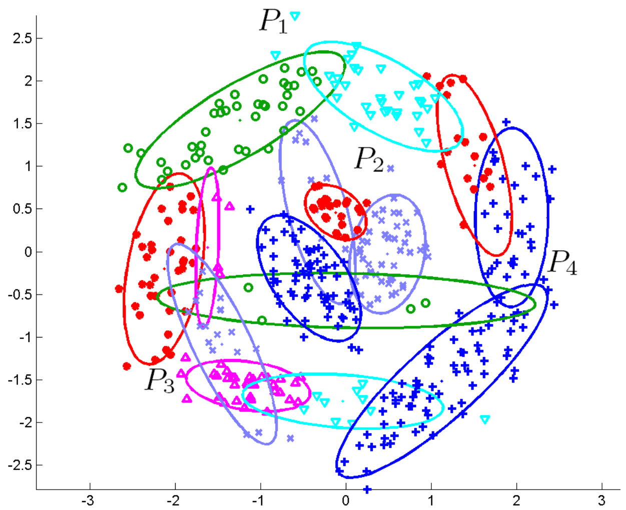

Figure 4.4: The topology of the data is closely approximated with a mixture of 20 factor analyzers. The ellipses outline two standard deviations from the center of the analyzers. The mixture can be used to detect "local" outliers, such as P_2 that may bridge the existing clusters.

however, MFA also estimates the optimal lower dimensional representation for the examples in each cluster. This is an essential characteristic when the data follow the structure of a lower dimensional manifold embedded in the original space R^D. The locally constrained SVC method suggested here exploits this property.

4.4.2 Regularization of the one-class learners

What we use in the introduced approach is the fact that MFA can single out the majority of the outliers that fall outside the main trajectory followed by the data. In Figure 4.4 the ellipses outline a two standard deviations region around the mean of the corresponding local clusters. Points, such as P_1 and P_2 that are too distant from their cluster centers, are indeed among the noisy points bridging the two global concentric clusters. Cleaning the dataset from these points can significantly improve the performance of the SVC method.

Note also, that using only the MFA method for reconstructing the underlying distribution will not provide a good enough solution either. Applied as a local method, similarly to Isomap and LLE, MFA can be instable because of the noise [9]. For instance, the two analyzers that bridge the two clusters on Figure 4.4 will impede the proper identification of the present formations. This comes to illustrate the importance of having an additional input from the global density method too.

Before we show how the information obtained through MFA can improve the one-class SVMs, it would be useful to understand how the outliers impact the detected contours. In the soft margin formulation (4.2), every example is allowed to cross the decision boundary with a penalty controlled by the slack variables ξ. This makes the decision function less complex, at the price of some misclassified examples x_i, which in this case means that the function underestimates the density around these examples. Misclassification of all such x_i is penalized proportionally to their distance to the separation plane (ξ_i), but with the same weighting factor $\frac{1}{n\nu}$. Assuming that there is an additional, possibly uncertain, knowledge about which examples are actually outliers, the procedure might instead be changed to use different weighting factors. The idea is similar to the weighted SVM classification, that has been demonstrated to be suitable in the case of imbalanced classes [35], with the difference being that the weights now should be determined based on the confidence that a certain example is an outlier.

A confidence estimate of the importance of each example can be obtained by measuring the example's deviation from the mean of the factor analyzer that it belongs too. If $\mathbf{z}_j = (z_1^j, z_2^j, \ldots, z_{r_j}^j)'$ are the projections of the examples that are assigned to the j-th mixture component, then the deviation of each example projection z_i^j can be expressed through the Mahalanobis distance:

$$\mathbf{d}_j = [(\mathbf{z}_j - \mu_j)' C_j (\mathbf{z}_j - \mu_j)]^{1/2} \tag{4.6}$$

In the above, the covariance of the j-th factor analyzer is estimated as $C_j = \Lambda_j \Lambda_j' + \Psi$ (see [43]). Now we adjust the penalty for misclassifying examples that are believed to be outliers (i.e. examples with large distance d_{ij} to their corresponding center μ_j) to be small, so that the decision function is not so influenced by them. This will smooth the separation boundary inferred by function (4.4), and hence will decrease the chance of having multiple small contours around even not so representative neighborhoods. To achieve that, each individual penalty term is modified to be inversely proportional to its Mahalanobis distance d_i. Now (4.2) is written as follows:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},\xi} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{1}{n\nu}\textstyle\sum_{i=1}^{n} \tfrac{1}{d_i}\xi_i - b \\ \text{subject to} \quad & \mathbf{w} \cdot \Phi(x_i) \geq b - \xi_i,\ \xi_i \geq 0,\ i = 1..n \end{aligned} \tag{4.7}$$

For brevity of notation in (4.7), we have omitted the indicator showing to which factor analyzer the projection of an example x_i belongs, yet it should be kept in mind that the distances d_i are computed based on the individual mixture components. Note, that the feature map Φ is applied on the original variables x_i rather than the projections z_i. The latter is done because the projecting dimensions for every factor analyzer are different. As density estimation in higher dimensional spaces has degrading effectiveness, it may still be necessary to perform a dimensionality reduction of the space X before solving the optimization problem (4.7). For that purpose, one could detect a global coordination for all factor analyzers [80], or just use a linear reduction as PCA as suggested by Ben-Hur et al. [14]. Here we use the second approach, which does not diminish the importance of MFA in the overall scheme, as the example weights have been computed based on the intrinsic dimensionality inferred by the method.

The Lagrangian now has the form:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{n\nu}\sum_{i=1}^{n}\frac{1}{d_i}\xi_i - b - \sum_{i=1}^{n}\alpha_i(\Phi(x_i) - b + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \tag{4.8}$$

Taking the derivatives with respect to the primal variables $\mathbf{w}$, b, and ξ_i and substituting in (4.8) we obtain the dual optimization problem which we now try to maximize with respect to the dual variables α_i. This yields the constraint optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2} \textstyle\sum_{ij} \alpha_i \alpha_j \Phi(z_i) \cdot \Phi(z_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \tfrac{1}{d_i n \nu} \\ & \textstyle\sum_{i=1}^{n} \alpha_i = 1 \end{aligned} \tag{4.9}$$

In [83] the one-class SVM optimization problem is demonstrated to be solvable with a fast iterative technique called sequential minimal optimization (SMO). What makes the method applicable is the special form of the objective function and the linear equality constraints $\sum_{i=1}^{n} \alpha_i = 1$. Both, the function and the equality constraints in (4.9), are similar to the ones in problem (4.3), which means that we can perform the optimization using SMO again. What differs in the two formulations (4.3) and (4.9) are only the constraints imposed on α_i, which now are allowed to be upper-bounded by different values. That upper-bound is determined based on the confidence for the corresponding examples to be outliers.

It may be argued that the described process will also identify as noisy points that are not necessarily outliers. For instance, the points P_3 and P_4 in Figure 4.4. They are part of denser regions, yet they deviate from their component centers too. In this sense we say that the feedback obtained from MFA is uncertain, yet this will not necessarily have a detrimental effect, as the collaboration with the density estimation procedure again comes into play. The decision function evaluated for the denser region where P_3 resides will be positive for a large set of kernel widths, and the optimal slack variable for this point will most likely be zero, regardless of what constraint is imposed on its weight.

The number of mixture components that we use in the evaluation procedure is set to be larger than the number of clusters that we would like to be detected in the data. In general it is a good practice to use at least several analyzers for each cluster that we want to detect. This ensures that if there are non-convex clusters present, each cluster may be covered with more than one component on average, which would better outline the cluster's topology. This may seem like very loose specification, yet we observe that even providing a relatively large number of components the LSVC algorithm still correctly detects as bounded support vectors points that are indeed outliers. We could also specify the number of analyzers as a fraction of the total number of examples. In this mode MFA would roughly approximate methods, such as Isomap or LLE which use neighborhoods of certain size to reconstruct the underlying structure. For example, if we set the number of analyzers to be equal to $\frac{n}{10}$, then most components in the mixture will on average have ten elements and will resemble the neighborhoods constructed by the local methods.

Before we conclude this section, we note another interesting estimate that can be obtained through the MFA algorithm, namely, that of the tradeoff parameter ν. [82] demonstrates that the optimal ν to be specified in the one-class optimization problem (4.2) should be an upper bound on the fraction of outliers that are assumed to be present in the data. This fact by itself is not very helpful, as the number of outliers is unknown in advance. Using the factor analyzers, however, such an estimate can be obtained for example by counting the elements which deviate significantly from the mean of their mixture component. For the purpose, we compute the empirical standard deviation of the Mahalanobis distances d_{ij} within each analyzer. Then we set $\nu = \sum_j s_j/n$, where s_j is the number of examples that are more than two standard deviations away from the mean of the j-th analyzer.

4.5 Discussion

Using an example, we will elaborate on the effect that the introduced weighting scheme has on the detected contours. We run the two algorithms, SVC and the LSVC, on the synthetic "target dataset" from Figure 4.6 (see Section 4.6 for details about its generation). The parameters used for both algorithms are $\gamma = 8$ and $\nu = 0.1$. Ten factor analyzers were used in the weight computing step for LSVC.

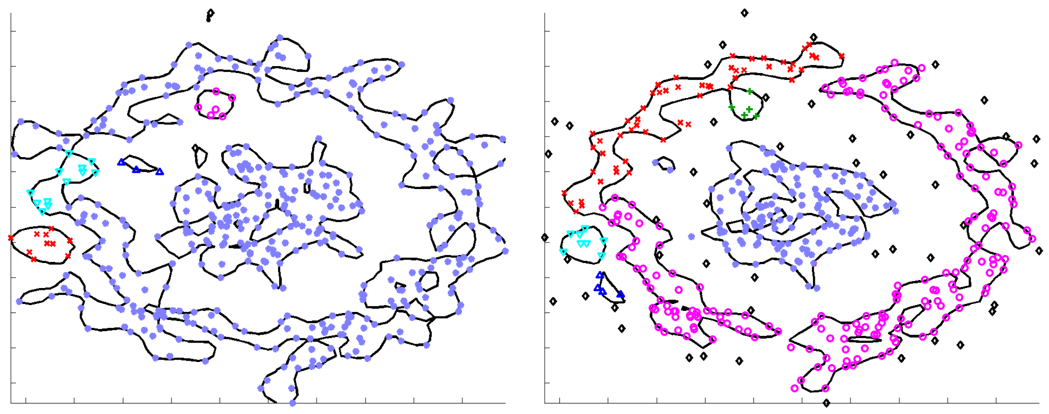

Figure 4.5: $\gamma = 8$ and $\nu = 0.1$. *Left*: SVC tries to accommodate all examples building complex contours and incorrectly bridging the two concentric clusters. *Right*: LSVC, the proposed here method, detects most outliers. The contours shrink towards the truly dense regions and the two main clusters are separated correctly.

The black diamonds on the graphs represent bounded support vectors or support vectors which were found to form no connected components with any of the other examples (i.e. they form a one point cluster). As Figure 4.6 left shows, SVC tries to learn a decision boundary that accounts for almost all of the examples. This results in bridging the two concentric clusters present in the data. For the same parameters, LSVC (see Figure 4.6 right) forms contours that are shrunk towards the means of the data distribution. Multiple points, with lower density around them, are identified as bounded support vectors. Such points are identified as noise in the MFA step, and their weights in building the decision function have been decreased. The central circle is now identified as a separate cluster, while the outer circle has approximately as many clusters as in the SVC case.

It could be argued that we give an advantage to the LSVC algorithm by allowing the penalty to vary due to the different weights, while for SVC it is fixed with the constant ν. It is true, that if we relax the penalty for all examples (i.e. increase ν), some of the noisy points will be identified as bounded support vectors by SVC too. Yet, there is the problem of how exactly ν should be determined to improve the performance of SVC. In this case the value $\nu = 0.1$ was automatically computed using the previously described procedure of counting the deviating points for the ten factor analyzers. Furthermore, a suitable value for ν may not exist for the currently selected γ. For example, increasing ν twice produces almost identical results as $\nu = 0.1$. Increasing it four times leads to the graph on Figure 4.6 left.

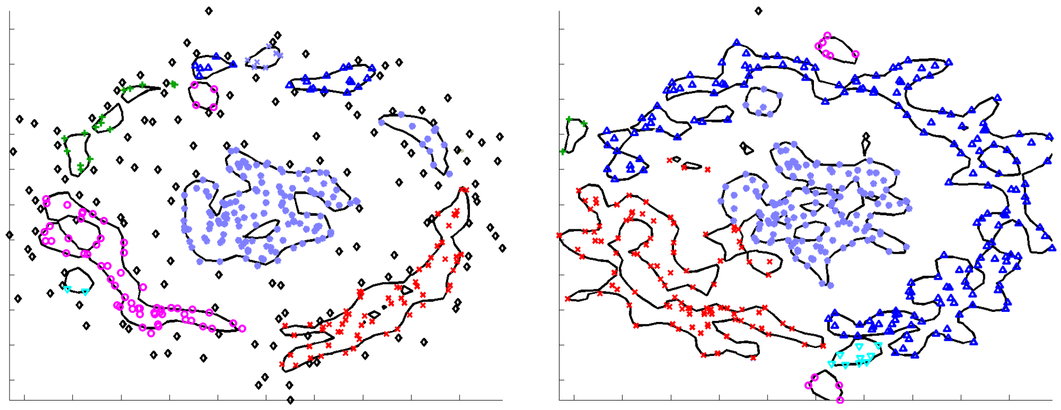

Figure 4.6: *Left*: SVC for $\gamma = 8$ and $\nu = 0.4$. Many outliers are now correctly identified, but the rest of the points are split into multiple uninformative clusters. *Right*: SVC for $\gamma = 9$ and $\nu = 0.1$. Increasing γ also cannot achieve the LSVC effect. The contours become very tight and complex and start splitting into multiple clusters.

SVC detects the internal circle as a separate cluster now, but the outer circle is split into multiple nonintuitive clusters. Another alternative to isolating the noisy points would be to keep ν unchanged and decrease the kernel width instead. However, there is again the issue of what kernel width would be more accurate. Decreasing the width, though, increases the complexity of the boundary forming some rather tight contours (see Figure 4.6 right) that at some point may also split into multiple clusters.

4.6 Experimental evaluation

To demonstrate the performance of the proposed method we employ the following unsupervised procedure, which we run with both algorithms SVC and LSVC. For every dataset we specify the number of clusters k that we would like the algorithm to detect. For all experiments the number of factor analyzers in LSVC is set to 10. The value of ν is determined as the fraction of outliers detected in the MFA step. The same value of ν is used in parameterizing SVC too. We vary $\log \gamma$ within the interval $[-16, 16]$ starting with -16 and incrementing it with step 1 at a time. This gradually increases γ (i.e. decreases the kernel width) and causes for more clusters to emerge. We stop the procedure when the number of clusters detected by the algorithm $\hat{k}$ surpasses k (i.e. $\hat{k} \geq k$). The procedure is suitable for comparing the robustness of the two algorithms, as the rate with which the clusters emerge when slowly decreasing the kernel width is highly correlated to the stability of the density estimation procedure in the presence of noise.

Though SVC and LSVC are primarily density estimation methods, rather than clustering algorithms for detection of fixed number of classes, we also check which would be the k clusters that the algorithms will return to the users. For the purpose, if $\hat{k}$ is larger than k, we start appending smaller clusters to the k largest clusters. The merging is done based on the minimal pairwise distance between the different clusters. Though not formal enough, and prone to certain errors, this merging step is suitable for detecting whether the clusters identified by the algorithms are well separated or there are dense regions that bridge them. The bounded support vectors are also assigned to their closest cluster.

4.6.1 Synthetic datasets

We first study the performance of SVC and LSVC on the synthetic dataset used throughout this exposition. The data represents two concentric circles (see Figure 4.7), and is generated similarly to one of the datasets used by Ben-Hur et al. in [14]. The inner concentric circle contains 150 points from a Gaussian distribution with identical variance along both dimensions. The outer circle is composed of 300 points from a radial Gaussian distribution and a uniform angular distribution.

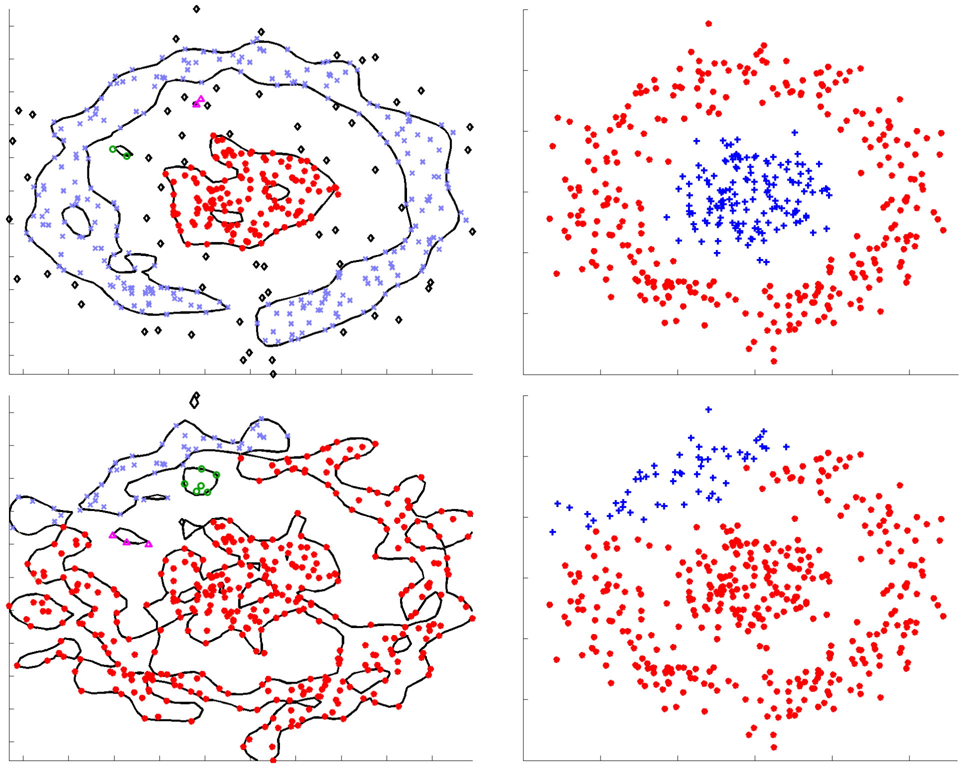

Figure 4.7: *Top*: the proposed LSVC algorithm; *left*: the contours and the clusters identified by the automatic procedure (the black diamonds indicate the bounded support vectors detected as noise); *right*: merging to obtain only two clusters. *Bottom*: the SVC algorithm; *left*: identified contours and clusters; *right*: merging to obtain only two clusters.

We set $k = 2$ and run the described automatic procedure. The ν value is computed to be 0.1. For $\log \gamma < 2$ both SVC and LSVC detect only one cluster. For $\log \gamma = 2$ LSVC and SVC detect four clusters (see Figure 4.7 left) and as $\hat{k} > k$ the procedure terminates. LSVC identifies 62 bounded support vectors (the black diamonds on the graph) against only 2 for SVC. The merging of the detected clusters results in 99% accuracy for LSVC and only 54% for SVC (see Figure 4.7 right). Manually probing among a larger set of (γ, ν)-pairs

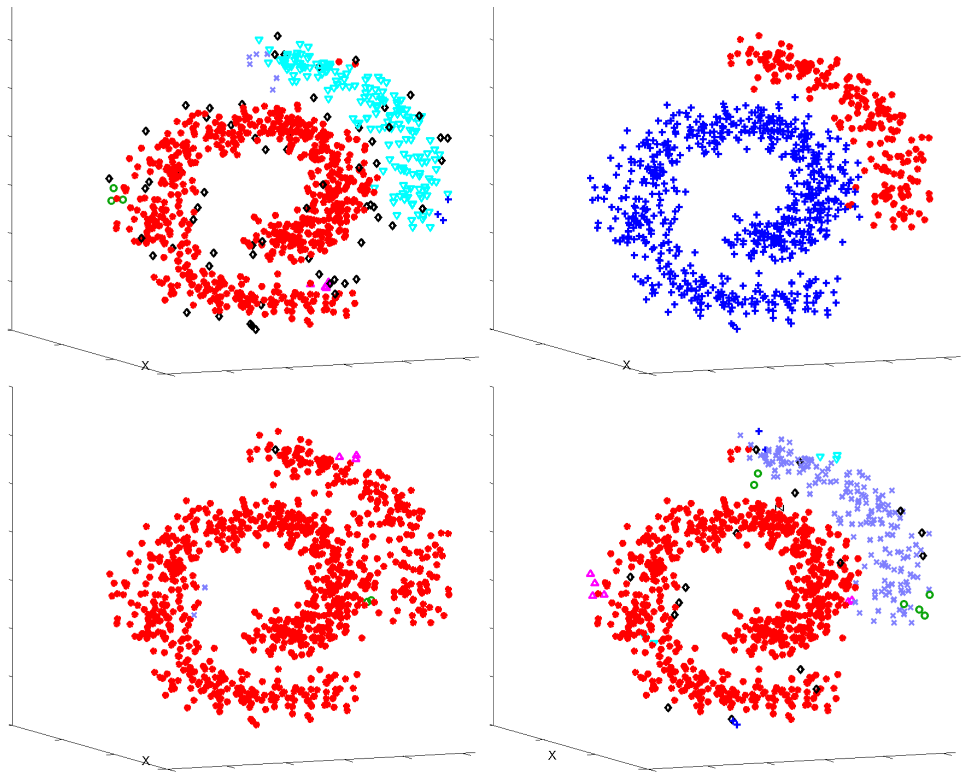

Figure 4.8: *Top*: the proposed LSVC algorithm; *left*: clusters identified by the automatic procedure; *right*: merging to obtain only two clusters. *Bottom*: the SVC algorithm; *left*: 5 small nonrepresentative clusters are identified with the automatic procedure; *right*: using supervision we detect parameters that lead to better clustering, which still fails to isolate the noise.

we managed to identify values for SVC that also produced high accuracy after the merging procedure, but for those values there were multiple nonintuitive clusters detected by the algorithm and some rather complex contour boundaries.

The *Swiss roll* dataset is a standard benchmark data for evaluating local unsupervised techniques for clustering and dimensionality reduction [79, 93]. We have removed some of the examples from the original dataset to obtain two disconnected non-convex clusters (see Figure 4.8). The data is three dimensional and contains 900 examples to which we have additionally added some Gaussian noise.

For this experiment, the MFA step of the LSVC algorithm is set to use a two dimensional projection $\mathbf{z}$. The number of required clusters is set to $k = 2$. The tradeoff term is computed as $\nu = 0.07$. $\log \gamma = -1$ is the first value for which the LSVC method detects more than one cluster ($\hat{k} = 9$). The number of bounded support vectors is 65 (see Figure 4.8 left top). Note that the bounded support vectors are positioned on the periphery of the two clusters, detecting much of the bridging noise that could degrade the clustering approach. Applying the merging procedure yields the clustering presented on Figure 4.8 top right. The accuracy is again approximately 99%.

The SVC algorithm detects $\hat{k} = 5$ clusters for $\log \gamma = -2$, and thus the automatic procedure terminates. Four of the clusters, however, correspond to some small dense neighborhoods and do not detect the two large point formations in the data (see Figure 4.8 bottom left). Only one bounded support vector was found, underestimating significantly the amount of noise present. The accuracy after merging is 78% with most points from the smaller cluster being assigned to the larger one. We again manually probe for other possible parameters that can produce a more accurate merging step for SVC. We find that the pair $(\log \gamma = -1, \nu = 0.07)$ identifies 14 clusters and 16 bounded support vectors (see Figure 4.8 bottom right), which after merging do lead to high accuracy as in the LSVC algorithm. Again, in this case, the detection of the suitable values required additional supervision and still produced larger number of not very representative small clusters.

4.6.2 Time series extracted from facial images

The *Frey face* images have been demonstrated by Roweis et al. [80] to reside on a smooth two dimensional manifold. Several examples of the images in different representation are shown in Figure 4.9 and Figure 4.10. In the evaluation here we randomly select 1000 examples from the original dataset.

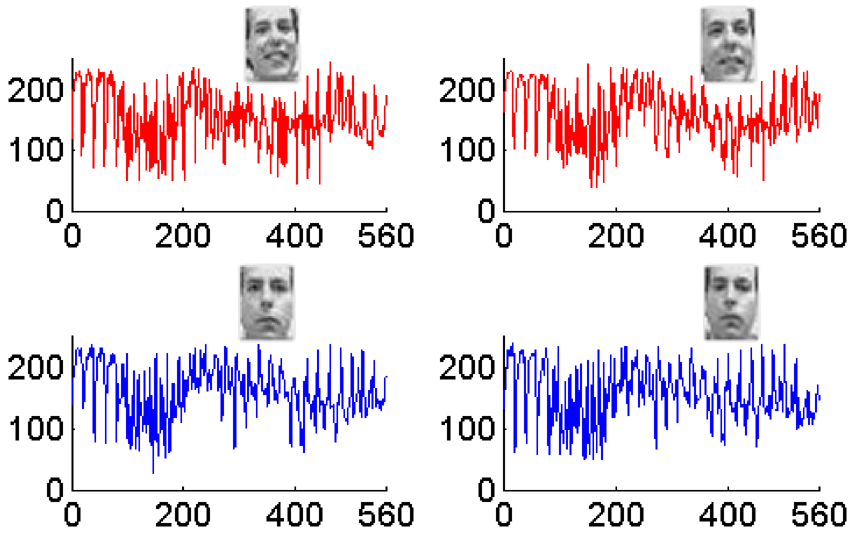

Figure 4.9: Time series representation of the Frey face images. Each point corresponds to the greyscale intensity of a pixel.

Every example is recorded as a 560 dimensional vector (the images are 20x28 pixels), where the dimensions correspond to the greyscale intensities of each pixel. This naturally converts the images into a time series representation as shown with the four examples in Figure 4.9. The representation is quite expressive, preserving much of the information available in the images. For example, while the dependence of neighboring pixels within a row of an image is directly preserved through the correlation of neighboring points in the corresponding time series, the dependence of neighboring pixels within a column is also captured by the periodicity in the time series representation.

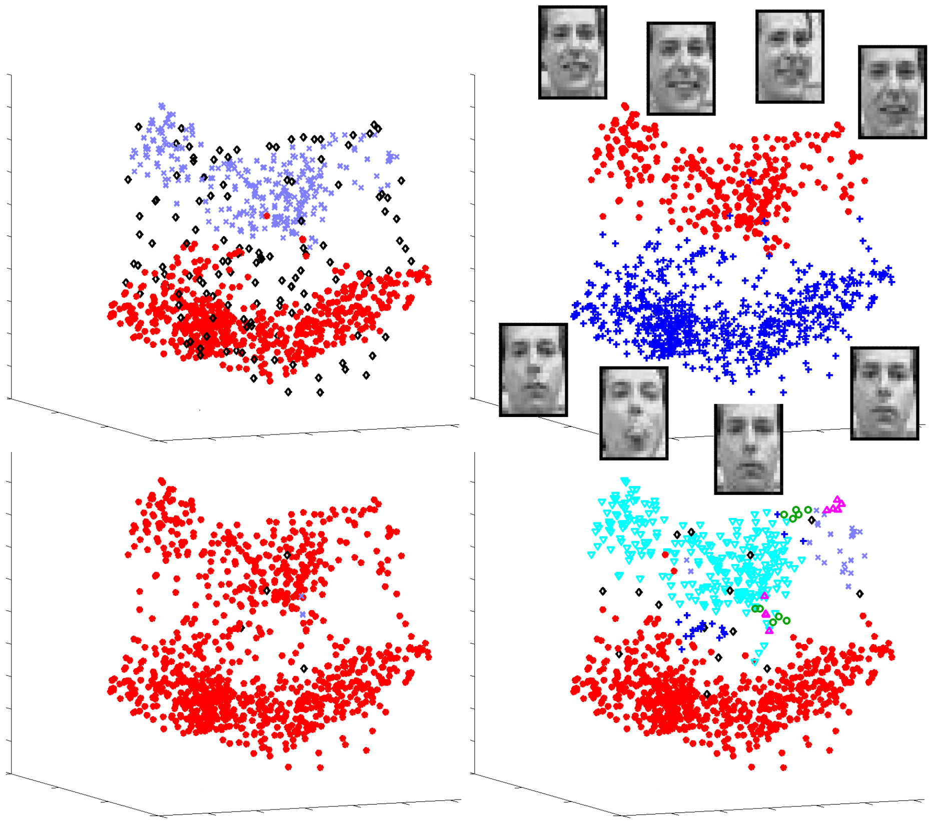

Figure 4.10: *Top*: the proposed LSVC algorithm; *left*: clusters identified by the automatic procedure; *right*: merging clusters to obtain only two clusters. *Bottom*: the SVC algorithm; *left*: 1 large and 1 small nonrepresentative cluster are identified with the automatic procedure; *right*: using supervision we detect parameters that lead to better separation, but still with some nonrepresentative clusters.

As mentioned above the data are known to reside on a smooth manifold, where the position of the examples on the manifold is determined by the expression of the face and the rotation of the head. Those are the features that separate the data into the two dense clouds seen in the figure. To facilitate the density estimation procedure rather than using the high dimensional representation space we first apply an off-the-shelf dimensionality reduction technique (PCA) and project the data along the top three eigenvectors. The MFA step of the algorithm is again set to use two dimensional projections **z**. We also set $k = 2$, aiming to detect the two dense formations that can be observed on the PCA projection in Figure 4.10.

The tradeoff ν is computed to be 0.07 and $\log \gamma = -14$ is the first γ for which LSVC detects more than one cluster. The algorithm identifies exactly $\hat{k} = 2$ clusters and 129 bounded support vectors which again outline correctly the bridging noise between the two distributions (see Figure 4.10 top left). Assigning the bounded support vectors to the closest dense region results in the clustering demonstrated in Figure 4.10 top right.

For the SVC algorithm $\log \gamma = -13$ yields the kernel width that first detects more than one cluster ($\hat{k} = 2$). One of the clusters, however, is a small region of just a few elements (see Figure 4.10 bottom left). The merging step does not change this result either. Increasing $\log \gamma$ twice did lead us to better cluster assignment (see Figure 4.10 bottom right), which after merging the multiple clusters produced two clusters similar to the ones identified with LSVC. However, the value required additional supervision and also detected multiple non-representative clusters. Moreover, very few of the scattered examples between the two dense formations were detected as noise (i.e. bounded support vectors).

4.6.3 Time series representing shapes

The *Arrowheads* dataset was earlier studied in Chapter 3. Here we use a subset of it containing the time series extracted from the shape contours of 600 projectile images. There

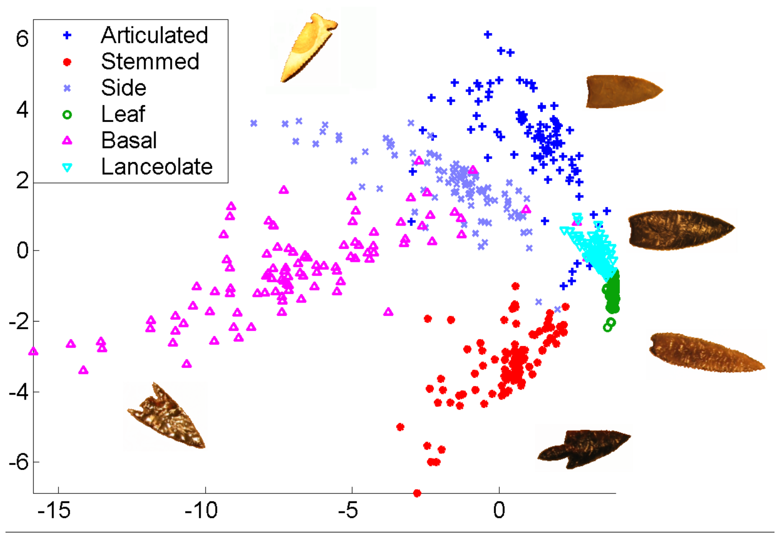

Figure 4.11: *Arrowheads* dataset. 2D MDS projection with representative examples for the six classes present in the data.

are six classes of projectiles labeled in the subset. The time series are formed again as described in Chapter 3. We have further aligned and resampled all time series in the dataset, representing them with 340 dimensional vectors.The data is then projected using the two largest eigenvectors (see Figure 4.11).

The dataset is rather difficult to discriminate, with many bridging elements between the available classes, and with some classes (*leaf* and *lanceolate*) significantly overlapping.

We run the SVC and the LSVC algorithms with $k = 6$. The MFA projection $\mathbf{z}$ is again two dimensional. The value for ν is computed to be 0.09. The contours detected by the two methods and the clusters after the merging procedure are presented in Figure 4.12.

Both methods detect less than six clusters for $\log \gamma < 1$. For $\log \gamma = 1$, LSVC finds 19 clusters and isolates 60 bounded support vectors (see Figure 4.12 top left). After the

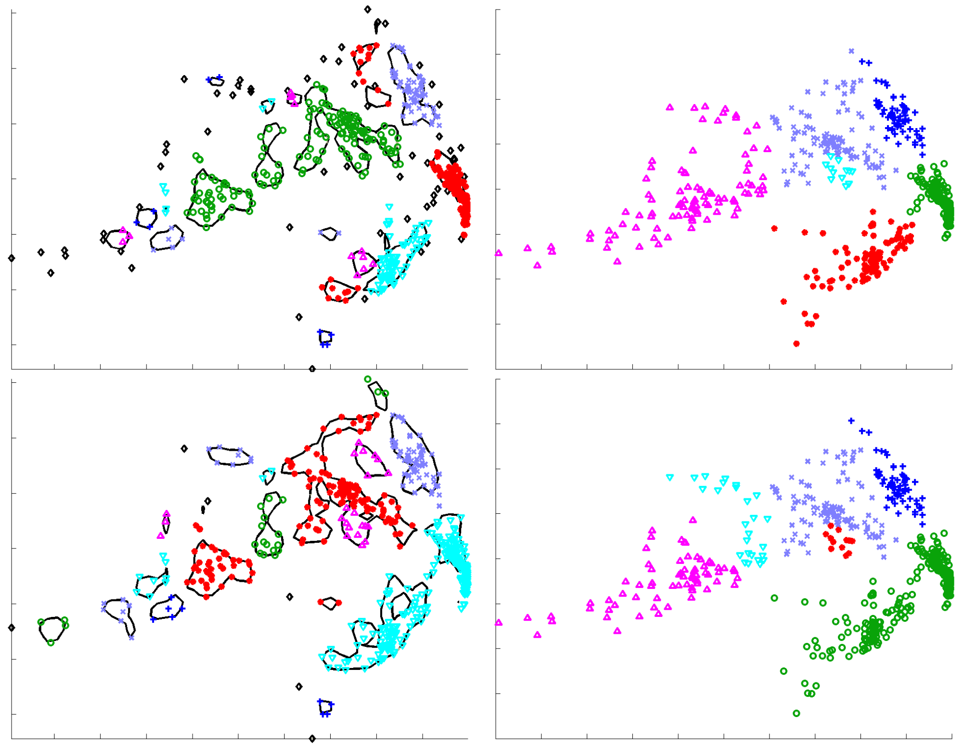

Figure 4.12: *Top*: the proposed LSVC algorithm; *left*: the contours and clusters identified by the automatic procedure (colors are assigned agnostically); *right*: merging to obtain six clusters. *Bottom*: the SVC algorithm; *left*: the identified contours and clusters (colors are assigned agnostically). The method tries to accommodate much of the noise building more complex boundaries; *right*: merging to obtain six clusters. The accuracy is significantly lower compared to the LSVC algorithm: 60% vs 73%.

merging procedure, we map the six clusters that we identify to the original labels that yield highest accuracy. The result is presented in Figure 4.12 top right. The accuracy of the method is $\sim 73\%$. In summary, the LSVC method performs well and succeeds in capturing the objectively dense regions in the data.

The SVC approach fails to separate the stemmed class, and hence the worse accuracy of the clustering $\sim 60\%$ (see Figure 4.12 bottom right). The number of clusters detected by the method is 18 and the number of bounded support vectors is six (see Figure 4.12 bottom left). SVC also identifies some objectively dense regions in the data, but the contours are again more complex and tend to accommodate most of the bridging elements between the different classes.

4.7 Improving the efficiency of transductive learning

Transductive learning is a form of semi-supervised learning where apart of the labeled data we are also given a set of unlabeled examples, whose position in space might be explored in order to improve the inductive learners alone [28, 50]. More formally, The observations $X = \{x_i\}_{i=1}^n$ now give rise to two separate sets: a set $L = \{(x_i, y_i)\}_{i=1}^m$ of labeled examples, where $y_i = \pm 1$; and a set of unlabeled examples $U = \{x_i^* = x_{m+i}\}_{i=1}^p$, where $m + p = n$. The goal is to improve the binary inductive classifiers trained only on L. Unlike other semi-supervised approaches, where trained on L and U classifiers are constructed in a manner that would increase their generalization performance on other unseen examples outside of X, in transductive learning the goal for the trained classifier is simply to achieve optimal performance only on X, i.e. derive maximally accurately the labels $\{y_i^*\}_{i=1}^p$ of the examples x_i^*. This subtle difference between transduction and semi-supervised learning in general, turns out to be important, as it allows constructing methods, which have at least theoretically better properties, meaning that better performance bounds can be derived for them [28].

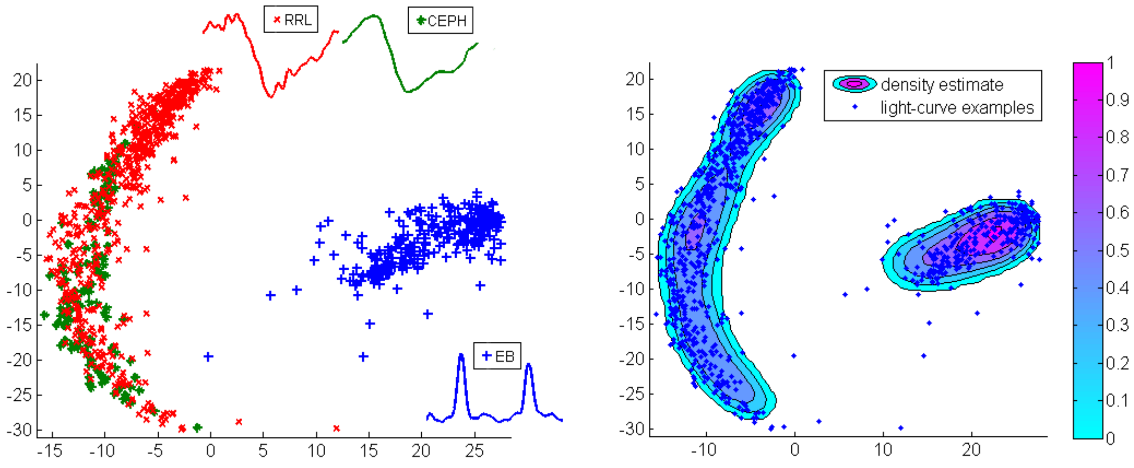

Figure 4.13: *Left:* Light-curves - PCA projection with three random time series examples from each of the three classes. *Right:* Density estimate of the LSVC method.

A drawback of transductive learning is that the algorithms become extremely inefficient with increasing the number of unlabeled examples. Here we look into an interesting application of the proposed constrained one-class SVM method for improving the efficiency of transductive inference. Only the density estimation part of the method will be used, and the cluster labeling part of LSVC will be ignored in the current application. For evaluation we use the light-curves dataset described in Chapter 2. A two dimensional PCA projection of the three classes of light-curves is presented in Figure 4.13 left. As pointed previously, the *Cepheids* (*CEPH*) have very similar shape with the *RRLyrae* (*RRL*) class, which is reflected in the large overlap of their projections. Separating them is hard, and instead the binary classification problem that we try to solve here is to recognize the *Eclipsed Binaries* (*EB*) time series from the time series of the other two classes.

Figure 4.13 right shows the density estimate of the proposed constrained SVC method. This noise resilient density estimate is an important information that we utilize in the current result. Intuitively, we would like a binary classifier not to intersect the curved manifold formed by the *non-EB* examples, nor the dense cloud of the *EB* time series.

Now assume that we have a sample of 1000 light-curves, only 70 of which are labeled as indicated on Figure 4.14 top left. The labels are respectively 50 *non-EB* and 20 *EB* examples, which corresponds roughly to the ratio of positive vs. negative among the unlabeled examples too. Knowing this ratio, though not always feasible, is a key requirement for semi-supervised approaches not to infer trivial solutions that assign all examples to only one of the classes, and therefore for the sake of the example we assume that we can closely satisfy it. The example is further manipulated to the extent that we have restricted the regions from which we have randomly sampled the labeled examples, and thus the two labeled groups are not representative of the entire *EB* and *non-EB* distributions. We find this to be a commonly encountered in practice scenario and one of the main reason for inductive approaches to achieve lower accuracy rates even when large number (an order of few thousands) of labeled examples are available.

As seen in Figure 4.14 top right, due to the biased labeled sample, the inductive linear SVM classifier is shifted closer to the *EB* class. The decision boundary that maximizes the margin to the two labeled sets, intersects part of the periphery of the *EB* set and thus classifies incorrectly some of its examples.

We would like despite of the bias in the labeled examples to come out with a classifier that is as far away from the dense distributions as possible. This is achieved with the

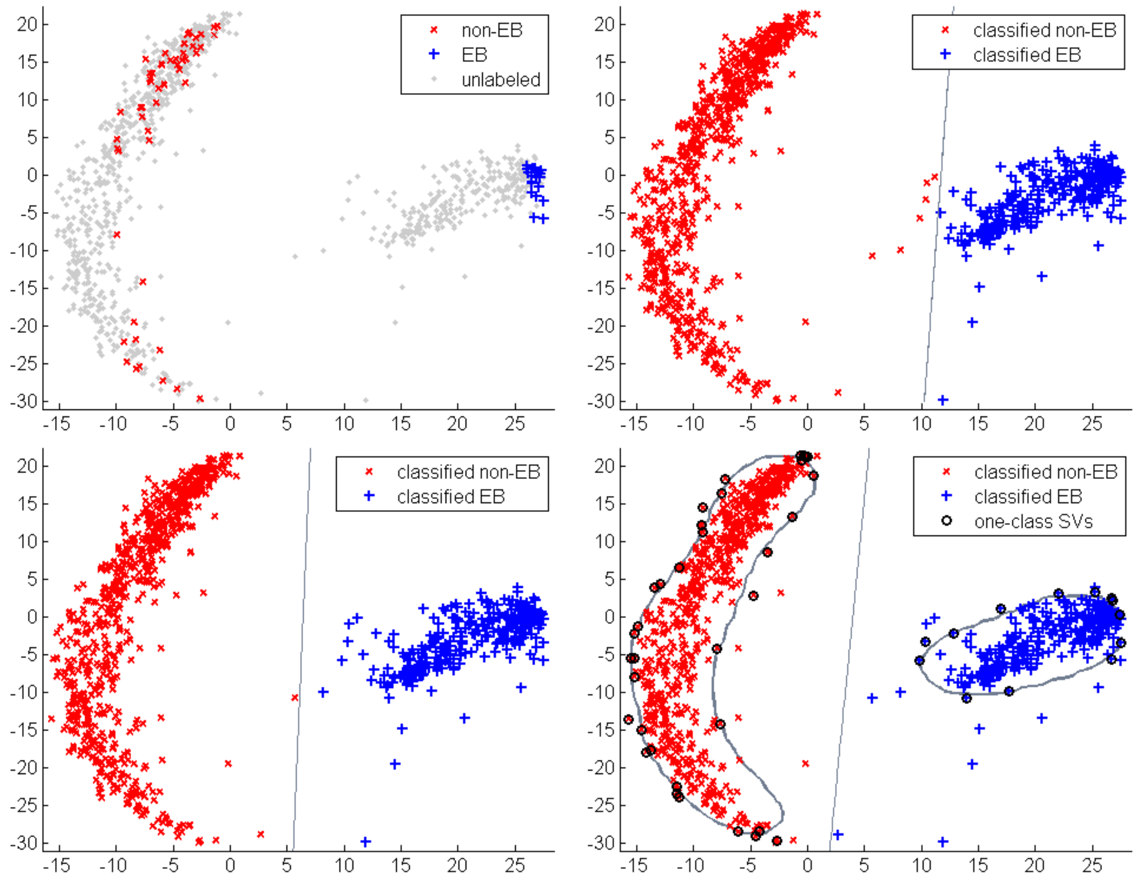

Figure 4.14: *Top left:* light-curves - PCA projection, labeled and unlabeled examples. *Top right:* performance of the inductive linear SVM classifier. *Bottom left:* performance of the transductive SVM classifier using all unlabeled examples. *Bottom right:* performance of the transductive SVM classifier using as unlabeled examples only the support vectors identified with the LSVC method.

tranductive SVM formulation which takes into account the position in the data space of the unlabeled data too (see Figure 4.14 bottom left). For clarity of the discussion we list this formulation, pointing for many of the details and how it is derived naturally from the

Structural Risk Minimization principle to [50, 94]:

$$\begin{aligned}
\min_{(\mathbf{w},\xi,\xi^*,b,\mathbf{y}^*)} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i + C^*\sum_{i=1}^{p}\xi_i^* \qquad (4.10)\\
\text{subject to} \quad & y_i((\mathbf{w}\cdot x_i)+b) \geq 1-\xi_i \quad i=1,m\\
& y_i^*((\mathbf{w}\cdot x_i^*)+b) \geq 1-\xi_i^* \quad i=1,p\\
& y_i^* = \pm 1\\
& \xi_i, \xi_j^* \geq 0 \quad i=1,m;\; j=1,p
\end{aligned}$$

In 4.10 ξ_i and ξ_j^* are the slack margins with which labeled and unlabeled examples respectively can be misclassified. For the evaluation in this section we use the hard margin formulations that do not allow slack for any of the examples in the inductive or the transductive settings. Therefore, no additional parameter adjustment is required for the tradeoff parameters C and (C, C^*) in the inductive or transductive linear SVM classifiers.

Problem 4.10 is hard to solve, because it has to optimize along the integer labels y_i^*. A full search would involve computing all possible ± 1 label assignments, which is equivalent to checking 2^p combinations. Even for modest number of unlabeled examples p, this is merely intractable. *Mixed-integer* programming solutions like [15] are also shown to be extremely inefficient [50]. An approximate solution that scales "up to 100,000 examples in reasonable time" has been suggested in [50]. The solution starts with the inductive solution (see Figure 4.14 top right) and in a gradient descend procedure tries to improve it by iteratively switching the label assignment for some of the examples x_i^*. We use the *SVMLight* transductive implementation from the author of this approach [50], and obtain the result in Figure 4.14 bottom left. The dense regions now start “pushing” away the inductive decision boundary through the help of the unlabeled examples. The effect of unlabeled examples causing the decision bound to pass through sparser regions of the data space has been observed before and is believed to be the main cause for semi-supervised approaches to achieve better accuracy [29].

The idea that we develop here is sparked by the question: are all unlabeled examples equally important for pushing the decision function away from the dense clusters, and if not, how can we select the examples that matter? The answer naturally arises from the techniques that we have developed so far in this chapter. Figure 4.14 bottom right shows the contours of the dense regions identified with the LSVC method. Obviously we do not want the decision function to cross those, and therefore it seems as a reasonable idea to remove from consideration the unlabeled examples within those contours, i.e. examples for which the density support function (4.4) is positive. Including them will only make the learner try to evaluate decision functions that pass through them and separate them with large margins, which is precisely what we would like to avoid. We also do not want to be influenced by examples in the sparse areas too. Those can be identified as the bounded support vectors with largely deviating negative values of the decision function (4.4). By removing the examples with high density support and the deviating bounded support vectors, we are left only with the support vectors outlining the contours of the one-class SVM and some of the bounded support vectors close to this decision boundary. For our light-curve example we are thus left with 47 such support vector examples (see the circled examples in Figure 4.14 bottom right), from which we remove three that happen to be among the labeled subset too. Finally, we run again the transductive SVM implementation, but this time only with the remaining 44 bounded support vectors as unlabeled examples x_i^*, rather than all 930 as in the previous transductive evaluation. As shown in Figure 4.14 bottom right, this largely reduced set of unlabeled examples again achieves the effect of moving the inductive decision away from the dense regions. The accuracy of both transductive approaches is 99.68% which is a slight improvement over the 99.14% classification accuracy of the inductive learner. The gain in running time of the improved transductive learner though is enormous - for the small dataset of 1000 light-curves the original transductive SVM requires 74 secs. to converge to a solution, while on the same computer configuration and again with the *SVMLight* transductive implementation our improved solution is computed in less than 3 secs. This time also includes the running time of the *MFA* weight computation and the time for the one-class density estimate.

4.8 Concluding remarks

The chapter presents a method for improving the stability of the support vector clustering (SVC) algorithm in the presence of noise and bridging elements between the available clusters. The introduced algorithm uses a mixture of factor analyzers (MFA) to learn a weighting, representing the confidence that a certain example is an outlier. The weights are later used to regularize the complexity of the decision function computed for the clustering. On synthetic and real datasets, we demonstrate that our method produces superior results than SVC alone. The results also indicate that complementing the best features from local and global clustering approaches can provide for a powerful tool for learning of clusters sampled from nonlinear manifolds. The developed techniques are also demonstrated to introduce several orders of magnitude speed-up over one of the most popular semi-supervised learning approaches, the transductive support vector machines, without sacrificing their effectiveness.

A possible future direction of research could study the effect of automatically inferring the suitable number of analyzers to be used with the model. Such non-parametric Bayesian extensions of the MFA model have been proposed before in [42], but whether they will be effective and efficient for the purpose of nonlinear manifold reconstruction needs to be further verified.

Chapter 5

Conclusion

There is an amazing wealth of domains where data have natural time series representation. What is surprising though is not merely the fact that data can be expressed as time series, but also that this yields high rates of accuracy across multiple data mining tasks. Coupled with appropriate similarity measures, we demonstrated effective time series representations for datasets as diverse as star light signatures, web queries, surveillance footages, object and facial images etc. The natural question here is: can we go even further? Can we extend this ordered real value encoding for domains where discrete descriptions are assumed as a norm? For certain datasets, such as DNA sequences reasonable attempts have already been made [27, 111], yet may be even better time series encoding can be defined for them. There are other areas, however, like the widely emerging social networks on the web, that still lack reasonable time series representation. If such is found, it may lead to solving some of the major problems associated with the enormous graphs representing these networks.

It could allow, for example, the parallelization and the efficient mining of patterns observable for many users within the network, or for building descriptive "profiles" that allow us differentiate between the behavior of individual users.

Interestingly, the biggest strength of the time series representation, its expressiveness, turns out to be its most problematic feature too. To achieve higher accuracy, we often use higher dimensional time series. The higher dimensionality leads to higher risks of accumulating some amounts of noise along certain dimensions. The source of this can be different. For example, some of the telescope images that astronomers use to extract light curves might be of degraded quality, or the preprocessing techniques that are applied might turn out to be erroneous in their own. Thus, even if the data are of intrinsically lower dimensionality, the amount of noise accumulated in the original, feature-rich space, prohibits from detecting the accurate low dimensional embedding. This illustrates the need of deriving time series techniques robust in the presence of noise. Here we demonstrated such unsupervised techniques for both of the possible scenarios - when the noise itself is of interest and the severe outliers need to be detected and studied; and when the noise should be isolated from any structured signal. From here on we think there are three interesting research paths that can be followed:

- Currently, a major constraint for some of the presented unsupervised algorithms, such as the b-Isomap reconstruction and the constraint support vector clustering, is their efficiency. The efficiency and the memory requirements are a drawback of all subspace reconstruction techniques, yet for the noisy settings that we work in it might be possible to sparsify the affinity matrices used in the reconstruction. This will bring our manifold methods closer to the real world standards of terabyte-sized datasets, used for example by the leading web search companies.

- An important aspect that has long intrigued the machine learning community is whether having more data is always helpful. May be removing the noisy examples will lead to better efficiency and better, or at least comparable, accuracy as demonstrate in [6]. The effect of removing examples has not been thoroughly studied neither for the discord detection algorithm, nor for the manifold reconstruction techniques.

- An important issue, especially in the presence of noise, is what contributes to the better accuracy of our methods. Is the prediction error decreasing only on average (i.e. there is less bias), or we can expected an improvement for every single example (i.e. there is smaller variance). In a previous work [108] we have demonstrated that one can construct time series learners that improve both the bias and the variance of the prediction error. We believe that algorithms with such properties will become more and more important in the light of the sparse datasets with long tail distributions that arise with the web search phenomena, where people realize that simply improving the average results and ignoring many elements in the tail is not a winning approach. The bias-variance performance remains to be studied for the presented algorithms.

Lastly, we demonstrated how even small amounts of labeled data can improve the accuracy of the proposed time series learners. Supervision turns out to be especially beneficial in the presence of noise where unsupervised methods alone can hardly infer the accurate structure. Examples were demonstrated when label information helps detecting class specific significant discords. Labels were also used in selecting the more suitable method for reconstructing the shape manifold subspace, as well as for improving the inductive learners with a constraint density estimate which can also speed-up significantly transductive inference. Semi-supervised methods, and transduction in particular, have recently emerged as a major learning area and are expected to dominate the field within the years to come. To support this an inspiring discussion in [28] states “transduction stresses new philosophical ideas related to noninductive inference... in ten years the concept of noninductive inference will be much more popular than inductive inference”. For this “prophesy” to fulfill a major breakthrough in the computational efficiency of transduction needs to be achieved. We see the developed here ideas as a step towards this direction.

Bibliography

[1] http://bulge.astro.princeton.edu/∼ogle/.

[2] http://www.cia.gov/cia/publications/factbook/.

[3] http://www.pmel.noaa.gov/tao/index.shtml.

[4] T. Adamek and N. O'Connor. Efficient contour-based shape representation and matching. *Multimedia information retrieval*, pages 138–143, 2003.

[5] J. Ameen and R. Basha. Mining time series for identifying unusual sub-sequences with applications. *1st International Conference on Innovative Computing, Information and Control*, 1:574–577, 2006.

[6] A. Angelova, Y. Abu-Mostafa, and P. Perona. Pruning training sets for learning of object categories. In *CVPR '05: Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 494–501, 2005.

[7] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *CIKM '07: Proc. of the sixteenth ACM conference on Conference on information and knowledge management*, pages 811–820, 2007.

[8] F. Angiulli and F. Fassetti. Very efficient mining of distance-based outliers. In *CIKM '07: Proc. of the sixteenth ACM conference on Conference on information and knowledge management*, pages 791–800, 2007.

[9] M. Balasubramanian and E. Schwartz. The Isomap algorithm and topological stability. *Science*, 295(5552):7, 2002.

[10] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03: Proc. of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, 2003.

[11] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[12] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Mach. Learn.*, 56(1-3):209–239, 2004.

[13] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[14] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. Support vector clustering. *J. Mach. Learn. Res.*, 2:125–137, 2002.

[15] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Proc. of the 1999 conference on Advances in neural information processing systems (NIPS)*, pages 368–374, 1999.

[16] S. Berchtold, C. Böhm, D. Keim, and H. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proc. of the 16th ACM Symposium on Principles of database systems (PODS)*, pages 78–86, 1997.

[17] B. Bhanu and X. Zhou. Face recognition from face profile using dynamic time warping. pages IV: 499–502, 2004.

[18] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.*, 7(1):108–116, 1995.

[19] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[20] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. *Proc. of the 9-th International Workshop on Artificial Intelligence and Statistics*, 2003.

[21] Markus Breitenbach and Gregory Z. Grudic. Clustering through ranking on manifolds. In *Proc. of the 22nd international conference on Machine learning (ICML)*, pages 73–80, 2005.

[22] M. Breunig, H. Kriegel, R. Ng, and J. Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000.

[23] H. Buf, M. Bayer, S. Droop, R. Head, S. Juggins, S. Fischer, M. Bunke, J. Roerdink, J. Pech-Pacheco, G. Christobal, H. Shahbazkia, and A. Ciobanu. Diatom identification: A double challenge called ADIAC. *In Proceedings ICIAP*.

[24] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236, 1973.

[25] A. Cardone, S. Gupta, and M. Karnik. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *J. Comput. Inf. Sci. Eng.*, 3(2):109–118, 2003.

[26] C. Chang, S. Hwang, and D. Buehrer. A shape recognition scheme based on relative distances of feature points from the centroid. *Pattern Recognition*, 24(11):1053–1063, 1991.

[27] H. Chang, N. Lo, W. Lu, and C. Kuo. Visualization and comparison of dna sequences by use of three dimensional trajectories. *First Asia-Pacific bioinformatics conference on Bioinformatics*, 2003.

[28] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[29] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. *In Proc. of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[30] D. Charalampidis. A modified k-means algorithm for circular invariant clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1856–1865, 2005.

[31] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'03)*, pages 493–498, 2003.

[32] M. Chuah and F. Fu. ECG anomaly detection via time series analysis. Technical Report LU-CSE-07-001, 2007.

[33] S. Davies and A. Moore. Mix-nets: Factored mixtures of gaussians in bayesian networks with mixed continuous and discrete variables. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 168–175, 2000.

[34] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *OSDI'04: Proc. of the 6th conference on Symposium on Opearting Systems Design & Implementation*, 2004.

[35] S. Du and S. Chen. Weighted support vector machine for classification. In *Proc. Internation Conference on Systems, Man and Cybernetics*, volume 4, pages 3866–3871, 2005.

[36] B. Everitt. *An Introduction to Latent Variable Models (Monographs on Statistics and Applied Probability)*. Chapman & Hall, 1984.

[37] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Record*, 23(2):419–429, 1994.

[38] U. Fayyad, C. Reina, and P. Bradley. Initialization of iterative refinement clustering algorithms. In *Knowledge Discovery and Data Mining*, pages 194–198, 1998.

[39] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[40] A. Fu, O. Leung, E. Keogh, and J. Lin. Finding time series discords based on Haar transform. In *Proc. of the 2nd International Conference on Advanced Data Mining and Applications*, pages 31–41, 2006.

[41] K. Fukunaga and P. Narendra. A branch-and-bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comp.*, 24(7):750–753, 1975.

[42] Z. Ghahramani and M. Beal. Variational inference for Bayesian mixtures of factor analysers. *Advances in neural information processing systems*, 12:449–455, 2000.

[43] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, 1996.

[44] A. Ghoting, S. Parthasarathy, and M. Otey. Fast mining of distance-based outliers in high dimensional datasets. In *Proc. of the 6th SIAM International Conference on Data Mining*, 2006.

[45] W. Hodges and K. Zamudio. Horned lizard (*p*hrynosoma) phylogeny inferred from mitochondrial genes and morphological characters: understanding conflicts using multiple approaches. *Molecular Phylogenetics and Evolution*, 31:961–971, 2004.

[46] E. Hung and D. Cheung. Parallel mining of outliers in large database. *Distrib. Parallel Databases*, 12(1):5–26, 2002.

[47] N. Iwabe. Sister group relationship of turtles to the bird-crocodilian clade revealed by nuclear DNA-coded proteins. *Molecular Biology and Evolution*, 22:810–813, 2004.

[48] H. Jagadish, N. Koudas, and S. Muthukrishnan. Mining deviants in a time series database. In *Proc. of the 25th International Conference on Very Large Data Bases*, pages 102–113, 1999.

[49] A Jalba, M. Wilkinson, J. Roerdink, M. Bayer, and S. Juggins. Automatic diatom identification using contour analysis by morphological curvature scale spaces. *Machine Vision and Applications*, 16(4):217–228, 2005.

[50] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proc. of the 16th International Conference on Machine Learning*, pages 200–209, 1999.

[51] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM '01: Proc. of the 2001 IEEE International Conference on Data Mining*, pages 289–296, 2001.

[52] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 102–111, 2002.

[53] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177, 2005.

[54] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proc. of the 5th IEEE International Conference on Data Mining*, pages 226–233, 2005.

[55] E. Keogh, L. Wei, X. Xi, S. Lee, and M. Vlachos. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. *VLDB*, 2006.

[56] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pages 607–614, 2001.

[57] E. Klassen, A. Srivastava, M. Mio, and S. Joshi. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):372–383, 2004.

[58] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. of the 24rd International Conference on Very Large Data Bases (VLDB)*, pages 392–403, 1998.

[59] C. Lee and A. Elgammal. Simultaneous inference of view and body pose using torus manifolds. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR)*, pages 489–494, 2006.

[60] J. Lee. *Introduction to Topological Manifolds (Graduate Texts in Mathematics)*. Springer, 2000.

[61] J. Lee and D. Lee. An improved cluster labeling method for support vector clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):461–464, 2005.

[62] S. Lee and K. Daniels. Gaussian kernel width generator for support vector clustering. In *Proc. of the International Conference on Bioinformatics and its Applications. Series in Mathematical Biology and Medicine*, volume 8, pages 151–162, 2005.

[63] D. Li and S. Simske. Shape retrieval based on distance ratio distribution. *HP Tech Report. HPL-2002-251*, 2002.

[64] N. Logothetis and D. Sheinberg. Visual object recognition. *Annu. Rev. Neurosci.*, 19:577–621, 1996.

[65] E. Lozano and E. Acuna. Parallel algorithms for distance-based and density-based outliers. In *ICDM '05: Proc. of the Fifth IEEE International Conference on Data Mining*, pages 729–732, 2005.

[66] K. Malatesta, S. Beck, G. Menali, and E. Waagen. The AAVSO data validation project. *Journal of the American Association of Variable Star Observers (JAAVSO)*, 78:31–44, 2005.

[67] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. *In Proc. International Workshop on Image Databases and MultiMedia Search*, pages 35–42, 1996.

[68] R. Mollineda, E. Vidal, and F. Casacuberta. Cyclic sequence alignments: Approximate versus optimal techniques. *Approximate versus optimal techniques. International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):291–299, 2002.

[69] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Multimedia Syst.*, 12(3):227–238, 2006.

[70] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2001.

[71] C. Olson and D. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1):103–113, 1997.

[72] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002.

[73] C. Papadimitriou and V. Vazirani. On two geometric problems related to the travelling salesman problem. In *J. Algorithms*, pages 231–246, 1984.

[74] D. Pokrajac, A. Lazarevic, and L. Latecki. Incremental local outlier detection for data streams. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, 2007.

[75] P. Protopapas, J. Giammarco, L. Faccioli, M. Struble, R. Dave, and C. Alcock. Finding outlier light-curves in catalogs of periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, 369:677–696, 2006.

[76] C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping. In *SIAM International Conference on Data Mining (SDM '05)*, pages 506–510, 2005.

[77] R. Ravi, M. Marathe, S. Ravi, D. Rosenkrantz, and H. Hunt III. Many birds with one stone: Multi-objective approximation algorithms. In *ACM Symposium on Theory of Computing*, pages 438–447, 1993.

[78] M. Riedewald, D. Agrawal, A. Abbadi, and F. Korn. Accessing scientific data: Simpler is better. In *Proc. of the 8th International Symposium in Spatial and Temporal Databases*, pages 214–232, 2003.

[79] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[80] S. Roweis, L. Saul, and G. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2002.

[81] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[82] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[83] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[84] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998.

[85] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in Neural Information Processing Systems (NIPS)*, pages 582–588, 2000.

[86] M. Shapiro. The choice of reference points in best-match file searching. *Commun. ACM*, 20(5):339–343, 1977.

[87] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

[88] R. Souvenir and R. Pless. Manifold clustering. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)*, pages 648 – 653, 2005.

[89] A. Srivastava, S. Joshi, W. Mio, and X. Liu. Statistical shape analysis: Clustering, learning, and testing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):590–602, 2005.

[90] D. Stoyan. On estimators of the nearest neighbour distance distribution function for stationary point processes. *Metrica*, 64(2):139–150, 2006.

[91] Y. Tao, X. Xiao, and S. Zhou. Mining distance-based outliers from large databases in any metric space. In *KDD '06: Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 394–403, 2006.

[92] D. Tax and R. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, 2004.

[93] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[94] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[95] R. Veltkamp and L. Latecki. Properties and performance of shape similarity measures. *IFCS Conference: Data Science and Classification*, 2006.

[96] R. Veltkamp and M. Tanase. Content-based image retrieval systems: a survey. *Technical Report*, 2001.

[97] M. Vlachos, Z. Vagena, P. Yu, and V. Athitsos. Rotation invariant indexing of shapes and line drawings. In *Proc. of the 14th ACM international conference on Information and knowledge management(CIKM)*, pages 131–138, 2005.

[98] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML '00: Proc. of the 17th International Conference on Machine Learning*, pages 1103–1110, 2000.

[99] C. Wang and X. Wang. Multilevel filtering for high dimensional nearest neighbor search. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 37–43, 2000.

[100] D. Wang, P. Fortier, H. Michel, and T. Mitsa. Hierarchical agglomerative clustering based t-outlier detection. *6th International Conference on Data Mining - Workshops*, 0:731–738, 2006.

[101] Z. Wang, Z. Chi, D. Feng, and Q. Wang. Leaf image retrieval with shape features. In *4th International Conference on Advances in Visual Information Systems*, pages 477–487, 2000.

[102] L. Wei and E. Keogh. Semi-supervised time series classification. In *KDD '06: Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753, 2006.

[103] L. Wei, E. Keogh, and X. Xi. SAXually explicit images: Finding unusual shapes. In *Proc. of the 6th International Conference on Data Mining*, pages 711–720, 2006.

[104] A. Weigend and N. Gershenfeld. *Time Series Prediction. Forecasting the Future and Understanding the Past*. Addison-Wesley Publishing Company, 1994.

[105] Y. Wu and K. Chan. An extended Isomap algorithm for learning multi-class manifold. *ICMLC*, 6:3429–3433, 2004.

[106] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML '06: Proc. of the 23rd international conference on Machine learning*, pages 1033–1040, 2006.

[107] L. Yang. Building k-connected neighborhood graphs for isometric data embedding. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 28:827–831, 2006.

[108] D. Yankov, D. DeCoste, and E. Keogh. Ensembles of nearest neighbor forecasts. In *European Conference of Machine Learning*, pages 545–556, 2006.

[109] D. Yankov and E. Keogh. Manifold clustering of shapes. In *ICDM '06: Proc. of the 6th International Conference on Data Mining*, pages 1167–1171, 2006.

[110] D. Yankov, E. Keogh, and K. Kan. Locally constrained support vector clustering. In *ICDM '07: Proc. of the 7th International Conference on Data Mining*, 2007.

[111] D. Yankov, E. Keogh, S. Lonardi, and A. Fu. Dot plots for time series analysis. In *ICTAI '05: Proc. of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pages 159–168, 2005.

[112] D. Yankov, E. Keogh, J. Medina, B. Chiu, and V. Zordan. Detecting time series motifs under uniform scaling. In *KDD '07: Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853, 2007.

[113] D. Yankov, E. Keogh, and U. Rebbapragada. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *Knowledge and Information Systems*, 15, 2008.

[114] D. Yankov, E. Keogh, L. Wei, X. Xi, and W. Hodges. Fast best-match shape searching in rotation-invariant metric spaces. *IEEE Transactions on Multimedia*, 10(2):230–239, 2008.

[115] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *4th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 311–321, 1993.

[116] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.

[117] J. Zunic, P. Rosin, and L. Kopanja. Shape orientability. In *ACCV(2)*, pages 11–20, 2006.