Sequence Matching as Subsequence Bijection and Densification of Distance Spaces

---

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Suzan Köknar-Tezel
SEPTEMBER, 2010

ABSTRACT

Sequence Matching as Subsequence Bijection and Densification of Distance Spaces

Suzan Köknar-Tezel
DOCTOR OF PHILOSOPHY

Temple University, SEPTEMBER, 2010

Dr. Longin Jan Latecki, Chair

Time series are common in many research fields. Since both a query and a target sequence may be noisy, i.e., contain some outlier elements, it is desirable to exclude the outlier elements from matching in order to obtain a robust matching performance. Moreover, in many applications like shape alignment or stereo correspondence it is also desirable to have a one-to-one and onto correspondence (a bijection) between the remaining elements. To address the problem of noisy time series data we propose using an algorithm that determines the optimal subsequence bijection (OSB) of a query and target time series. The OSB is efficiently computed since the problem's solution is mapped to a cheapest path in a DAG (directed acyclic graph). We make several significant improvements to the original OSB algorithm and show that these improvements are theoretically and experimentally justified. We compare OSB to standard and state of the art distance measures such as Euclidean distance, Dynamic Time Warping with and without warping window, Longest Common Subsequence, Edit Distance with Real Penalty, and Time Warp Edit Distance. Moreover, we show that OSB is particularly suitable for partial matching.

In addition to noisy data, imbalanced time series data sets present a particular challenge to the data mining community. Often, it is the rare event that is of interest and the cost of misclassifying the rare event is higher than misclassifying the usual event. When the data is highly skewed toward the

usual, it can be very difficult for a learning system to accurately detect the rare event. There have been many approaches in recent years for handling imbalanced data sets, from under-sampling the majority class to adding synthetic points to the minority class in feature space. To address the problem of imbalanced data sets, we present an innovative approach to adding synthetic points (*ghost points*) to the minority class in distance space and theoretically show that these points preserve the distances. All current methods that add synthetic points to minority classes do so in feature space. However, distances between time series are known to be non-Euclidean and non-metric, since comparing time series requires warping in time. In addition, in some fields data is not available as feature vectors, but instead as pairwise distances between objects in the data set. Therefore the only recourse to augmenting the minority class is to add synthetic points in distance space. Our experimental results on standard time series using standard distance measures show that our synthetic points significantly improve the classification rate of the rare events, and in most cases also improves the overall accuracy of support vector machines. We also show how adding our synthetic points can aid in the visualization of time series data sets.

For time series classification, a large number of similarity approaches have been developed, with the main focus being the comparison or matching of pairs of time series. In these approaches, other time series do not influence the similarity measure of a given pair of time series. By using the locally constrained diffusion process (LCDP), other time series do influence the similarity measure of each pair of time series, and we show that this influence is beneficial. The influence of other time series is propagated as a diffusion process on a graph formed by a given set of time series. We use LCDP when densifying the minority class data space by adding *ghost points*. Our experimental results demonstrate that using LCDP when densifying the minority class also improves the classification rate of the minority class.

ACKNOWLEDGEMENTS

If a person is lucky, there will be many people who profoundly influence her on her journey through life. I am just such a person. I also now get to formally acknowledge them.

First I would like to thank my advisor, Longin Jan Latecki. I have learned so much from him over the past few years, and he has been a very patient and supportive advisor. He is a wonderful mentor. I am also in awe of his dedication to scholarship. Other professors at Temple that I have had the good fortune to learn from and interact with: Alexander P. Yates and Haibin Ling (my committee members), Richard Beigel, Eugene Kwatny, Rolf Lakaemper, John T. Nosek, Zoran Obradovic, Arthur T. Poe, Yuan Shi, Robert L. Stafford, and Daniel B. Szyld. I also want to thank my sometime collaborator and cubicle mate, Xingwei Yang - you're next! And two very special people I met many years ago at Temple and who will, I hope, be life-long friends - Vladimir Vacic and Nagesh Adluru. Both of them have enriched my life, and are just fun to be around.

My deepest gratitude to many people at Saint Joseph's University who have given me so much support in so many ways. I thank Father Timothy R. Lannon, President, Dr. Brice Wachterhauser, Provost, and Dr. William Madges, Dean of the College of Arts and Sciences, for the resources to be able to complete this - I am honored to be a faculty member of SJU. To Dr. Agnes Rash, former chair of the Mathematics Department for starting me on this odyssey. To Dr. Gary Laison, who will always be my role model of a great teacher. And to the members of the Department of Computer Science: Dr. Jonathan Hodgson, Chair, for being on my committee and all the other myriad ways you have helped me; Dr. Susanna Wei and Dr. F. Betul Ataly for patiently listening to me grumble and giving me such good advice; Dr. Babak Forouraghi, for the wonderful job scheduling my classes for these oh so many years to give me time for research; Dr. George Grevera, for always making me laugh; and Terry Fasy, our administrative assistant extraordinaire, who is also

a great “break buddy”. My sincerest thanks to all of you.

My family has done so much for me, it is impossible to list everything. My love and thanks to my mother, Jeannine Köknar, for easing my guilt by providing my family with many home-cooked meals; to my father H. Sezer Köknar, who is the most honest, principled, and good man I know; to my brother Kemal Köknar, who asked my husband, “So what are you doing to support my sister?” and helped him see the light; to my sister Semra Köknar, for helping me write my application essay at the very beginning; and to my youngest brother Kevin Köknar, who makes life nicer just by being himself.

Though I am dedicating this dissertation to my children, I also have to thank them. They sacrificed so much for me to do this, and learned to be very self-sufficient and independent people in the process. So often they had to arrange their life around my schedule, and usually with no complaining. I’m really grateful to them for how much easier they made things for me. They are wonderful kids!

And finally, my husband Ahmet Tezel, who, without any choice in the matter, has traveled this entire long, hard, and bumpy road with me. And threatened divorce if I EVER decide to go back to school again.

To my children,

E. Hale Can, S. Erin Can, and D. Kerem Can.

I am so proud of the people you have become, some of it because of me, some in spite of me.

I love you so very much.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Time Series

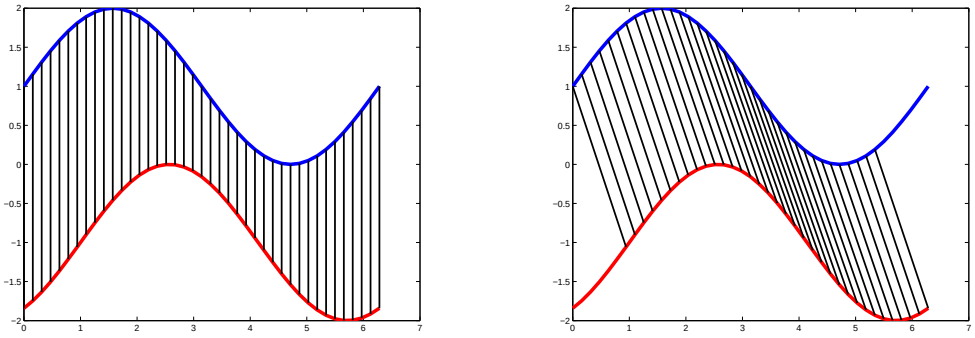
Sequences of real numbers are commonly used in all research fields. When the sequences of real numbers have a natural ordering, they are historically called *time series* even though the order imposed on the sequence may be from some dimension other than time, e.g., curvature at successive points of a contour. Time series are a ubiquitous and prevalent type of data that arise in all scientific disciplines, economics, financial forecasting, and many other domains. One of the earliest known time series plot is of planetary orbits from a tenth century monastery [47]. Because of the prevalence of time series data, there has been much research effort devoted to time series data mining in recent years. Many data mining algorithms have similarity measurements of sequences at their core. Examples include motif discovery [14], anomaly

detection [27, 44], rule discovery [24], classification [40], and clustering [1].

1.1.1 Related Work

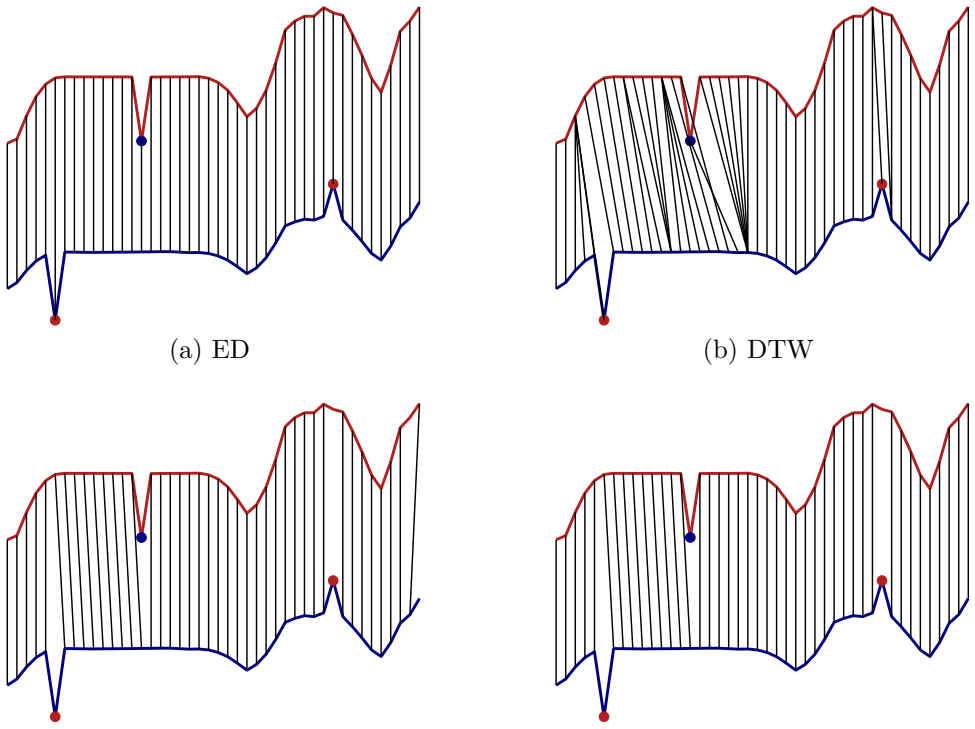
There have been many sequence similarity measures proposed in the literature. If two sequences of real numbers of equal length n are to be compared, the simplest way is to treat them as vectors in $\mathbb{R}^n$, and compute their Euclidean distance (ED). This approach is based on the assumption that both sequences are well aligned, i.e., that corresponding vector coordinates represent corresponding sequence elements. In many applications, e.g., speech recognition, this assumption is not satisfied. For example, in both graphs of Fig. 1.1, the top and bottom time series are identical except for a phase shift. It is obvious that the alignment in Fig. 1.1a, induced by the Euclidean distance, is not correct. In Fig. 1.1b, the optimal alignment is shown. Many researchers have mentioned in their work [24, 37, 40] that the Euclidean distance is not always the optimal distance measure for similarity searches of sequences. To overcome the limitations of ED, elastic measures were introduced that allow the stretching and compressing of the time series to find the best alignment.

The most well-known elastic measure is Dynamic Time Warping (DTW) [50, 42] that was first used for aligning spoken words. Identical words may be spoken at different rates, for example *no* and *nooo*, so a non-linear alignment of the speech patterns was needed. DTW allows two sequences to be stretched



(a) (b)

Figure 1.1: The graph in (a) shows two out of phase sine waves and the corresponding Euclidean distance. The graph in (b) shows the optimal alignment, which the Euclidean distance cannot find.



(a) ED (b) DTW

(c) LCSS ($\epsilon = 0.05$) (d) OSB ($\mathbf{C} = 0.1$)

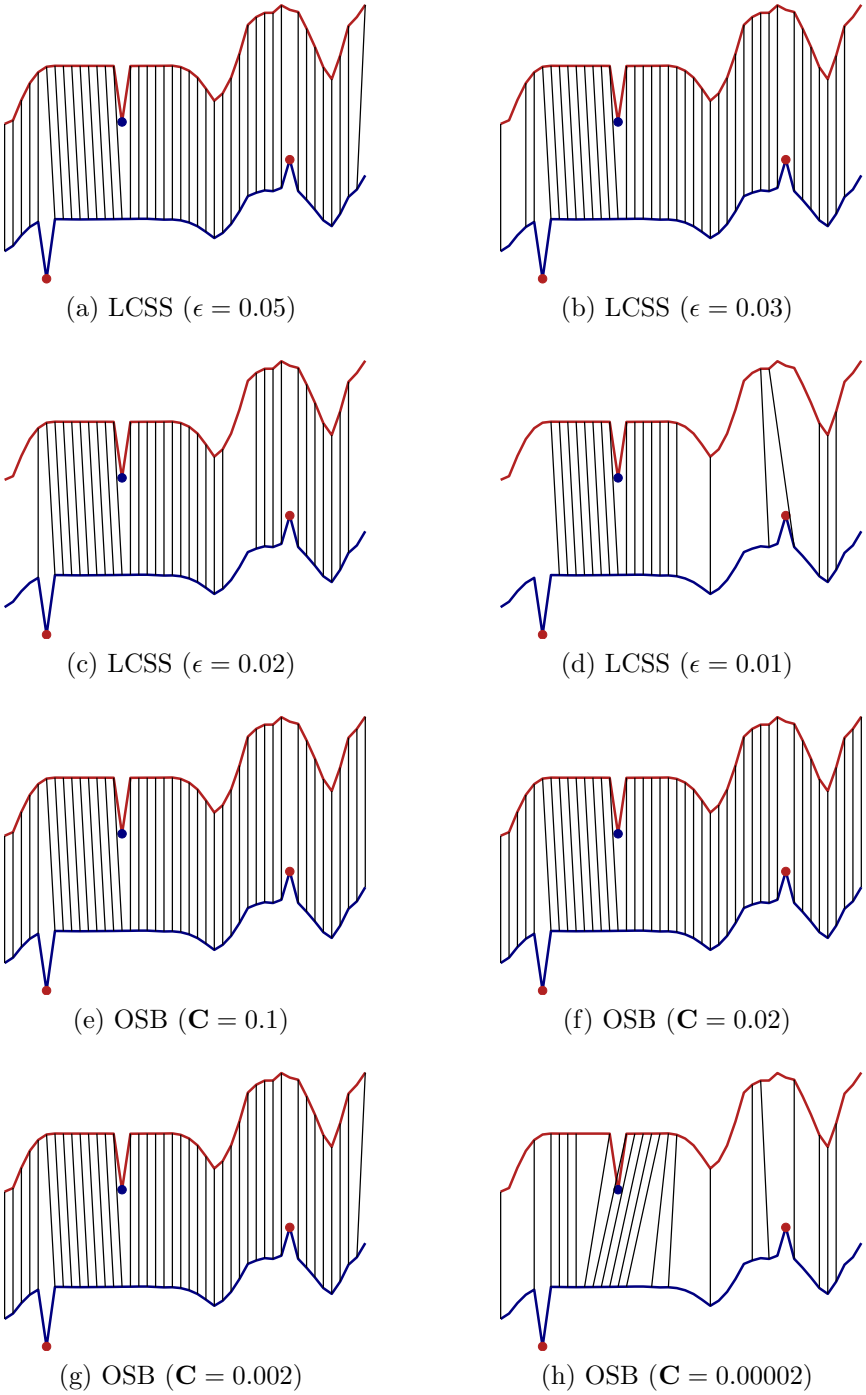
Figure 1.2: For each graph, the top and bottom sequences represent parts of contours of two different but very similar bone shapes from the MPEG-7 data set. The top sequence has one outlier manually added (the blue dot) and the bottom sequence has two outliers added (the red dots).

or compressed to optimize local alignments. The DTW distance is then computed as the sum of the distances of the corresponding elements, and dynamic programming is used to find corresponding elements so that this distance is minimal. The DTW distance has been shown to be superior to the Euclidean distance in many cases [1, 15, 52, 61], however, DTW is particularly sensitive to outliers, since it is not able to skip any elements of the sequences. In DTW, each element of the query sequence must correspond to some element of the target sequence and vice versa. Thus the optimal correspondence computed by DTW is a relation on the set of indices of both sequences, i.e., a one-to-many and many-to-one mapping. The fact that outlier elements must participate in the correspondence optimized by DTW often leads to an incorrect correspondence of other sequence elements. This fact is illustrated in Fig. 1.2b, where the query sequence on top has one outlier element (spike), and the target sequence at the bottom has two outlier elements. The correspondence of the elements is illustrated with straight lines. Observe that the outliers corrupt the correspondence computed by DTW. In particular, this is reflected in the fact that single elements of one sequence correspond to a large number of elements of the other sequence.

The main difference between DTW and the proposed algorithm, Optimal Subsequence Bijection (OSB), is that OSB can skip outlier elements of the query and target sequences when computing the correspondence while

DTW requires that each element of the query sequence is matched to each element of the target sequence. This makes the performance of OSB robust in the presence of outliers. Moreover, OSB defines a bijection on the remaining subsequences, which means that we have a one-to-one correspondence of the remaining elements.

The Longest Common Subsequence (LCSS) measure has been used in time series [18, 51] to deal with the alignment and outlier problems. Given a query and a target series, LCSS determines their longest common subsequence, i.e., LCSS finds subsequences of the query and target that best correspond to each other. The distance is based on the ratio between the length of longest common subsequence and the length of the whole sequence. The subsequence does not need to consist of consecutive points, the order of points is not rearranged, and some points can remain unmatched. However, when LCSS is applied to sequences of numeric values, one needs to set a threshold ϵ that determines when values of corresponding points are treated as equal [51]. The performance of LCSS depends heavily on the correct setting of ϵ, which may be a particularly difficult problem for many applications. For example, we run LCSS on the sequences shown in Fig. 1.2, with ϵ ranging from 0.01 to 0.10. The best correspondence found is shown in Figs. 1.2c and 1.3a where $\epsilon = 0.05$. Figs. 1.3c and 1.3d show how easily the correspondence can degenerate with very small changes in ϵ. In addition, no known algorithm exists to optimally



(a) LCSS ($\epsilon = 0.05$)

(b) LCSS ($\epsilon = 0.03$)

(c) LCSS ($\epsilon = 0.02$)

(d) LCSS ($\epsilon = 0.01$)

(e) OSB ($\mathbf{C} = 0.1$)

(f) OSB ($\mathbf{C} = 0.02$)

(g) OSB ($\mathbf{C} = 0.002$)

(h) OSB ($\mathbf{C} = 0.00002$)

Figure 1.3: LCSS is very sensitve to the value of the threshold parameter. The correspondence obtained by OSB is much more stable in spite of huge differences in the value of the jump cost.

set ϵ, though in the literature ϵ is often set to one-quarter the standard deviation of the normalized time series [13, 39]. We observe that the threshold ϵ is static, i.e., it is used for all pairs of elements of two sequences.

One of the key features of the proposed OSB is the fact that the equality of two elements is dynamic and it depends on other elements in their neighborhoods in both sequences. The main difference between LCSS and OSB is that LCSS optimizes over the length of the longest common subsequence, while OSB directly optimizes the sum of distances of corresponding elements. Moreover, the OSB penalty for skipping consecutive elements of the query sequence is proportional to the number of elements skipped, thus skipping one outlier costs less than skipping a consecutive subsequence of several elements. There is no direct penalty for skipping elements in LCSS, which often leads to accidental matches.

OSB has a parameter called the *jump cost*, or $\mathbf{C}$, which is the penalty for skipping elements in the sequences. Unlike ϵ, $\mathbf{C}$ is not a threshold for pairwise equality of sequence elements, which makes the sequence matching results significantly less sensitive to changes in its value. As an example, Figs. 1.2d and 1.3e show the best correspondence found by OSB for the two sequences. When $\mathbf{C}$ is changed from 0.1 to 0.02 (a change of 80%), still the correspondences are identical (Fig. 1.3f). Contrast that with the correspondences from LCSS after a change of 80% in ϵ in Figs. 1.3a and 1.3d. When $\mathbf{C}$ is decreased

98% from 0.1 to 0.002 (Fig. 1.3g), again the correspondences are identical except for the extreme right tail of the sequences. In order to get the degenerate correspondence equivalent of that shown for LCSS in Fig. 1.3d, **C** must be set to 0.00002 (Fig. 1.3h).

The edit distance, originally used as a distance measure between strings, has been adapted to work with sequences of real numbers. The edit distance between two strings is the smallest number of primitive operations (insertions, deletions, and substitutions) needed to transform one string into the other. Two recent edit distance algorithms for sequences of real numbers, Edit Distance with Real Penalty (ERP) [12] and Time Warp Edit Distance (TWED) [35] have been proposed. In ERP, a deletion from one sequence is treated as an insertion into the other sequence, and this inserted element is called a *gap* element. When calculating the distance between two elements, a constant value is used for gap elements and the L_1 distance is used if both elements are non-gap elements. Note that ERP can be viewed as a variant of L_1-norm except that it handles local time shifting, and as a variant of DTW except that it is a metric. TWED also combines L^p-norms with edit distance but uses the time stamps of the sequences when calculating the distance between elements. Using the time stamps of the elements controls the elasticity of the measure.

DTW [42] and LCSS [23] can control the elasticity by setting a threshold

that determines if two elements may be matched or not; if the difference between the time stamps (indices) is less than this threshold, the elements may be matched, otherwise the elements can not be matched. Though this constraint increases the efficiency of the algorithms, if the optimal solution lies outside this window, it will not be found. TWED instead uses the difference in the time stamps to linearly penalize the matching elements. This then favors matching elements that have close time stamps.

Both ERP and TWED induce distances that satisfy the triangle inequality, and are therefore metrics. Though OSB is not a metric, as discussed in [26, 32] there are clear arguments from human perception that the distances induced by human judgments are frequently nonmetric. Many well-established machine learning methods require the data to be metric, so non-metric distance spaces are forced to be metric by embedding them into Euclidean spaces. The distortion of the data that occurs with this embedding is assumed to be noise, but little is known about the real information loss. So working directly with non-metric distance spaces may better represent the real distances between objects and not suffer from loss of information.

1.2 Imbalanced Data Sets

Most traditional learning systems assume that the class distribution in data sets is balanced, an assumption that is often violated. There are many

real-world applications where the data sets are highly imbalanced, such as oil spill detection from satellite images [29], credit card fraud detection [9], medical diagnostics [38], and predicting telecommunication equipment failure [56]. In these data sets, there are many examples of the "normal" (the majority/negative class), and very few examples of the "abnormal" (the minority/positive class). But often it is the rare occurrence, the "abnormal", which is the interesting or important occurrence, e.g. an oil spill. In data mining, the rare occurrence is usually much more difficult to identify since there are so few examples and most traditional learning systems are designed to work on balanced data. These learning systems are biased towards the majority class, focus on improving overall performance, and usually perform poorly on the minority class. If a data set has say 999 examples of the normal event and only one example of the abnormal event, a learning system that predicts all examples as "normal" will be 99.9% accurate, but misclassify the very important abnormal example.

1.2.1 Synthetic Data Points

Mining imbalanced data sets has been the focus of much research recently [6, 16, 55], and one important direction is sampling strategies. Sampling methods may include removing majority class data points (under-sampling) or inserting minority class data points (over-sampling) in order to improve accu-

racy. Two well-known techniques for increasing the number of minority examples are random resampling and SMOTE (Synthetic Minority Over-sampling TEchnique) [10]. In random resampling, minority class examples are randomly replicated, but this can lead to overfitting. The SMOTE algorithm inserts synthetic data into the original data set to increase the number of minority class examples. The synthetic points are generated from existing minority class examples by taking the difference between the corresponding feature values of a minority class example x and one of its nearest neighbors in the minority class, multiplying each feature difference by a random number between 0 and 1, and then adding these amounts to the feature vector of x.

SMOTE and its variations, for example [11, 3, 21], have shown that they can improve overall classification accuracy and also improve the learning of the rare event. But SMOTE and its variations work only in feature space, i.e., each example is represented as a point in n-dimensional space where n is the number of features of each example. In these methods, synthetic points are added as a weighted average of the Euclidean coordinates of two existing points. However, the Euclidean distance is known to be unsuitable as a shape dissimilarity measure even if shapes are represented as vectors of their contour sample points. For example, the horse in Fig. 1.4c is computed as the average of the Euclidean coordinates of the two horses in Fig. 1.4a and Fig. 1.4b. The Euclidean coordinates were obtained as sequences of 2D coordinates of 100

(a) (b) (c)

Figure 1.4: (c) The mean horse computed by averaging corresponding sample contour points of the aligned shapes in (a) and (b).

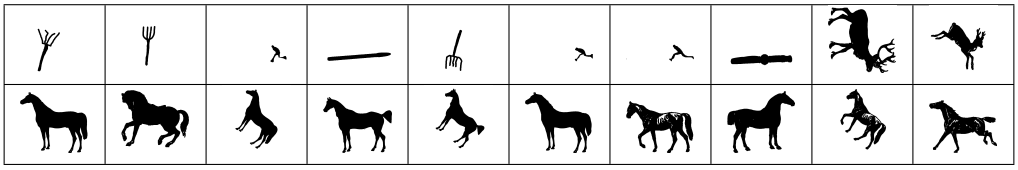


Figure 1.5: First row: the retrieval results of the mean horse from Fig. 1.4c. Second row: the retrieval results of the ghost horse created by the averaging in distance space of the two shapes in Figs. 1.4a and 1.4b.

aligned contour sample points. Although the feature points of both horses correspond, it is difficult to recognize the shape in Fig. 1.4c as a horse. To demonstrate the problem, we submitted the mean horse as a query to the MPEG-7 CE-Shape-1 part B data set [30]. The top ten retrieval results are shown in the first row of Fig. 1.5, ordered from left to right. Obviously none of the retrieval results is correct, but they are similar to the mean horse. For example, the tines of the forks are similar to the 'legs' of the average horse. The second row of Fig. 1.5 shows the retrieval results of the 'synthetic horse' generated by the proposed approach, which are all correct. We used the Inner Distance Shape Context (IDSC) [33] as the shape distance in both cases.

In addition, for some fields such as bioinformatics, image analysis, and

cognitive psychology, often the feature vectors are not available. Instead, in these domains the data may be represented as a matrix of pairwise comparisons where typically each element of the matrix is the distance (similarity or dissimilarity) between the corresponding original data points. This matrix represents the *distance space* of the data. Often, this distance space is non-metric because the distance function used to calculate the similarities or dissimilarities between the pairs of data points does not satisfy the mathematical requirements of a metric function. For example, the distances between time series are often non-metric due to warping. When only pairwise scores are available, the feature space based approaches to adding synthetic points cannot be used. In our experiments, we do not compare our synthetic points with SMOTE or random resampling because SMOTE and random resampling do not work in distance spaces, while the distinct advantage of the proposed approach is that it can be used in distance spaces. Our approach to balancing the data sets is to use supervised learning to increase the size of the minority class by inserting synthetic points directly into the distance space. Our synthetic points do not have any coordinates, i.e., they are not points in any vector space, which is why we call our synthetic points *ghost points*. But our ghost points *are* points in distance space.

To show the flexibility of our approach for unbalanced data sets, we insert ghost points into the distance spaces induced by two different distance

measures, Dynamic Time Warping (DTW) [7, 43] and Optimal Subsequence Bijection (OSB) [31]. We choose support vector machines (SVMs) to perform the classification because they are a fundamental machine learning tool and they have a strong theoretical foundation [49], though ghost points can also be uses with newer methods like LotkaVolterra derived models [25]. SVMs have been very successful in pattern recognition and data mining applications on balanced data sets. But when data sets are unbalanced, the SVM's accuracy on the minority/positive examples is poor. This is because the class-boundary learned by the SVM is skewed towards the majority/negative class [59]. This may lead to many positive examples being classified as negative (false negatives), which in some situations can be very costly (e.g. missing an oil spill, missing a cancer diagnosis). There are cost-sensitive SVMs that assign different costs to different classification errors and the SVM attempts to minimize misclassification costs instead of maximizing accuracy. Often though, in many real-world situations, the misclassification costs are unknown. Also, how to assign the costs is still an active research area and has not been solved. For example, [63] discusses two major approaches to converting traditional classifiers into cost-sensitive classifiers, and [53] combines modifying the classifier with changing the data distribution. Using ghost points eliminates the need for cost-sensitive classifiers and our experimental results show that inserting ghost points in both DTW distance spaces and OSB distance spaces can sig-

nificantly increase the SVM's ability to learn the rare events. Furthermore, in most cases, the addition of ghost points increases the SVM's overall classification accuracy.

1.3 Diffusion Process

Suppose there is a space describing some data set. Since differences between data in the same class can be very large and differences between data in different classes can be very small, pairwise data comparison sometimes has difficulty describing the dissimilarity correctly. Therefore, the distance between two data point can be correctly described only if it is considered in the context of other data points similar to them, which is the motivating idea of using the diffusion process in time series classification.

In our approach, the influence of other data points is propagated as a diffusion process on a graph formed by a given set of data points. However, as the data space is sparse, in some cases the diffusion process can not propagate properly. It is obvious that adding more data points to the data space would make the estimation of the data manifold more accurate. In other words, if the data space is properly densified, a diffusion process is able to better reveal its underlying manifold structure. To the best of our knowledge, this is the first time researchers try to solve the problem of densifying non Euclidean data manifolds, and as our experimental results illustrate, the diffusion process

performs significantly better on the densified manifolds.

1.3.1 Relation to Other Approaches

In [60], a graph transduction learning approach based on label propagation is introduced. It is the first approach in which the shape similarity of a pair of shapes is computed in the context of other shapes as opposed to considering only pairwise relations between two shapes. There are two key differences in the proposed approach. First, our diffusion process framework does not require label clamping as is the case in label propagation. Consequently, we need less than 10 iterations while label propagation requires on the order of 10,000 iterations. Second, they key step to improve the performance of a diffusion process is the introduction of ghost points, which densify the manifold structure allowing the diffusion process to better propagate.

In order to find relevant structures in complex geometries for classification and clustering, Markov chain techniques have been combined with graph-based methods. In [45], the L^1 distance between probabilities of transition is used as a metric between data points, and this metric is then employed to induce class labels. Zhou et al [64] assume a metric to arbitrarily convey a probability transition metric (proportional to weights) over the data to cluster it; [65] also assumes a metric from the beginning to arbitrarily claim a probability transition metric based on the original metric which is then used to propagate

labels using accumulated correlations; In [4], Azran assumes a metric over the data to induce a probability transition metric over an M-NN graph (where labeled points are absorbing), which is used to produce a probability distribution over the labels of each unlabeled point. While our setting is similar to these approaches, we treat the Markov random walks in a unsupervised setting. In order to make it more suitable for shape space, the transition matrix is focused on k nearest neighbors and point set correspondence is constructed for robustness.

For metric embedding, one of the most popular dimensionality reduction algorithms is Principal Component Analysis (PCA) [34]. PCA performs dimensionality reduction by projecting the original n-dimensional data onto the $d(\ll n)$-dimensional linear subspace spanned by the leading eigenvectors of the data's covariance matrix. However, in many real world problems, there is no evidence that the data is sampled from a linear subspace. Various researchers have considered the case then the data lives on or close to a low dimensional sub-manifold from random points lying on this unknown sub-manifold. Along this direction, many subspace learning algorithms have been proposed, such as Locality Preserving Projection [22], Locally Linear Embedding [41], and ISOMAP [46]. The proposed synthetic points directly densify the data manifold in the original distance space instead of the metric embedding. In other words, we use metric embedding to add new synthetic data points in the orig-

inal distance space. A key feature of our approach is that we can do this without distorting the original distances.

CHAPTER 2

Optimal Subsequence Bijection

2.1 Introduction

The *Optimal Subsequence Bijection* (*OSB*) works for the elastic matching of two sequences of different lengths m and n:

$$a = (a_1, \ldots, a_m) \text{ and } b = (b_1, \ldots, b_n).$$

The goal of OSB is to find subsequences a' of a and b' of b such that a' best matches b'. Skipping (not matching) some elements of a and b is necessary because both sequences may contain some outlier elements. However, skipping too many elements of either sequence increases the chance of accidental matches. To prevent this from happening, we introduce a penalty for skipping which we call *jump cost* and denote it with $\mathbf{C}$.

To formally define OSB, we need to first augment the sequences a and b

by first and last elements. To differentiate the original sequences a and b from the augmented sequences, we will use the notation

$$\bar{a} = (a_0, a_1, \ldots, a_m, a_{m+1}) \text{ and } \bar{b} = (b_0, b_1, \ldots, b_n, b_{n+1}).$$

The subsequences found with OSB using $\bar{a}$ and $\bar{b}$ will be denoted $\bar{a}'$ and $\bar{b}'$; $\bar{a}'$ will always contain the elements a_0 and a_{m+1} and $\bar{b}'$ will always contain b_0 and b_{n+1}. As explained below, these added elements do not contribute to the computed distance between the optimal subsequences a' and b'.

2.2 The Algorithm

We assume that the distance function d used to compute the dissimilarity value between elements of sequences a and b is given for $(i, j) \in \{1 \ldots m\} \times \{1 \ldots n\}$. We do not have any restrictions on the distance function d other than non-negativity, i.e., $d(a_i, b_j) \geq 0$, and therefore any distance function is possible. Usually, for sequences of real numbers we simply have the distance $d(a_i, b_j) = (a_i - b_j)^2$, which is also the case for our experimental results reported in Section 2.6. For the elements added to a and b, namely a_0, a_{m+1}, b_0, and b_{n+1}, we set $d(a_0, b_0) = 0$ and $d(a_{m+1}, b_{n+1}) = 0$. In particular, we define

$$d_{osb}(a_i, b_j) = \begin{cases} (a_i - b_j)^2 & \text{if } 1 \leq i \leq m \wedge 1 \leq j \leq n \\ 0 & \text{if } (i = 0 \wedge j = 0) \vee \\ & \quad (i = m + 1 \wedge j = n + 1) \\ \infty & \text{otherwise} \end{cases} \tag{2.1}$$

We want to select a subsequence a' of the query sequence a by skipping some outlier elements of a, so that each element of a' matches to some element of b in an order preserving manner with possibly skipping some outliers in b as well. The optimal correspondence is obtained by optimizing the balance between the dissimilarity of a' to its image subsequence of b and the penalties of skipping elements of a and of b.

The optimal correspondence can be found with a shortest path algorithm on a DAG (directed acyclic graph), e.g., see Fig. 2.2. The nodes of the DAG are all index pairs $(i, j) \in \{0 \dots m+1\} \times \{0 \dots n+1\}$ and the edge cost w is defined as

$$w((i,j)(k,l)) = \begin{cases} ((k-i-1)+(l-j-1)) \cdot C + d(a_k, b_l) & \\ & \text{if } i < k \wedge j < l \\ \infty & \text{otherwise} \end{cases} \tag{2.2}$$

Thus, the cost of an edge from node (i, j) to node (k, l) is the L_1 distance of vertices (i, j) and (k, l) in the matrix $\{0 \dots m+1\} \times \{0 \dots n+1\}$ times the jump cost $\mathbf{C}$ plus the dissimilarity measure of elements a_k and b_l. In the example DAG in Fig. 2.2, the purpose of the added nodes $(0, 0)$ and $(m+1, n+1)$ is to have distinct *source* and *destination* vertices for the shortest path algorithm and to allow the skipping of elements at the beginning and the end of a and b.

The output of OSB yields a *correspondence* defined as a monotonic injec-

tion

$$f : \{i_0, \ldots i_{m'}\} \rightarrow \{0, 1 \ldots n+1\}$$

such that $(i_0, \ldots i_{m'}) \subseteq (0, 1 \ldots m+1)$ is a subsequence, with $i_0 = 0$, $i_{m'} = m+1$, $f(i_0) = f(0) = 0$, and $f(i_{m'}) = f(m+1) = n+1$. Thus we require that the first and the last elements of $\bar{a}$ and $\bar{b}$ correspond. The sets of indices $\{i_0, \ldots i_{m'}\}$ and $\{f(i_0), \ldots f(i_{m'})\}$ define subsequences $\bar{a}'$ of $\bar{a}$ and $\bar{b}'$ of $\bar{b}$, such that f restricted to these sequences is a bijection, which explains the phrase "subsequence bijection" in *Optimal Subsequence Bijection* (*OSB*). Our goal is to find a subsequence bijection f that minimizes the function

$$\frac{1}{m'} \sum_{k=0}^{m'} w((i_k, f(i_k)), (i_{k+1}, f(i_{k+1}))). \tag{2.3}$$

This means, we need to find subsequences $\bar{a}' = (a_{i_0}, \ldots a_{i_{m'}})$ of $\bar{a}$ and $\bar{b}' = (b_{f(i_0)}, \ldots b_{f(i_{m'})})$ of $\bar{b}$ with a minimal total weight for the w defined in Eq. (2.2), which explains the word "optimal" in *Optimal Subsequence Bijection*. Note that we normalize the score by $\frac{1}{m'}$ to get the average matching score per matching pair of elements.

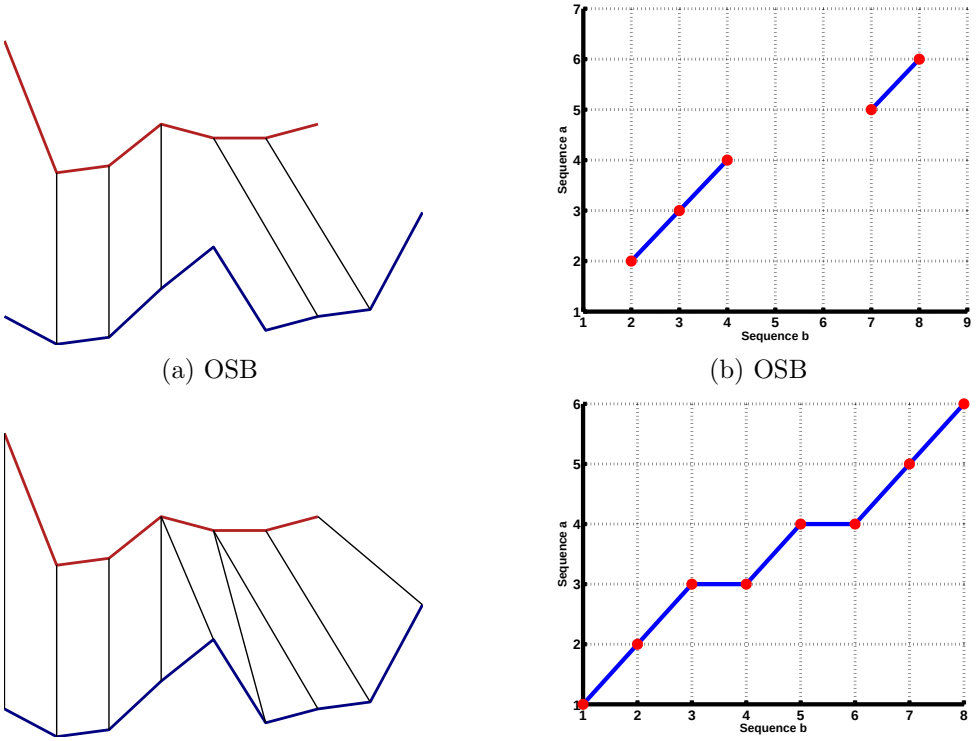
The OSB distance between two time series $\bar{a}$ and $\bar{b}$, obtained as the minimum of (2.3), is denoted as $OSB(\bar{a}, \bar{b})$. As stated above, we minimize Eq. (2.3) by computing a shortest-path algorithm on the DAG. Pseudo code of our dynamic programming algorithm is given in Algorithm 1.

We illustrate the proposed method on a simple example. Fig. 2.1a demonstrates the optimal correspondence found with OSB for two sequences $a =$

$(20, 1, 2, 8, 6, 6, 8)$ and $b = (5, 1, 2, 9, 15, 3, 5, 6, 20)$ with the jump cost $C = 1$. The corresponding sequence indices computed by OSB (not the values) are $(2, 2)$, $(3, 3)$, $(4, 4)$, $(5, 7)$, and $(6, 8)$ as seen in Fig. 2.1b. The shortest path on the DAG for these sequences is shown in Fig. 2.2. Observe that the outliers $a_1 = 20$, $b_4 = 15$, $b_5 = 3$, and $b_8 = 20$ are skipped, which results in a small distance between a and b, specifically, the OSB distance between a and b is 8. Moreover, we also find an optimal one-to-one correspondence of the remaining elements. For comparison, all the outliers are forced to participate in the matching if DTW is used as illustrated in Figs. 2.1c and 2.1d, which leads to a large distance between the two sequences; in this case the DTW distance between a and b is 14.28.

2.3 OSB and Existing Distance Functions

A recursive definition of OSB that is suitable for dynamic programming computation is given in Eq. (2.6). Eq. (2.6) optimizes the target function in Eq. (2.3) by recursively finding the distance between the optimal subsequences for $(a_0, \ldots, a_{m+1})$ and $(b_0, \ldots, b_n)$ or $(a_0, \ldots, a_m)$ and $(b_0, \ldots, b_{n+1})$ while including the penalty for the skipped elements of $\bar{a}$ and $\bar{b}$ when matching a_{m+1} and b_{n+1}, then adding in the distance between elements a_{m+1} and b_{n+1}. Formulas (2.4) - (2.9) compare the recursive definition of OSB to definitions of well-known sequence similarity measures. All definitions except that of the



(a) OSB (b) OSB

(c) DTW (d) DTW

Figure 2.1: Time series alignment example

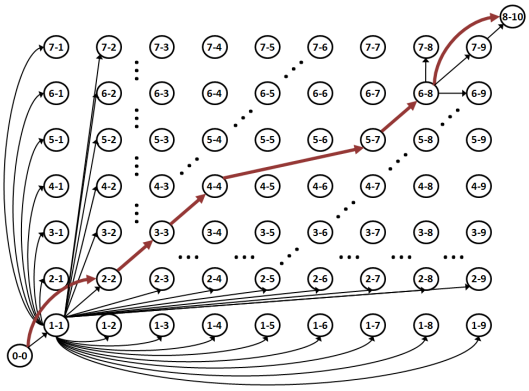


Figure 2.2: The directed acyclic graph (DAG) that is created from the sequences shown in Fig. 2.1. The numbers in the nodes are the corresponding row-column indices of the elements in the distance matrix.

squared Euclidean Distance (ED) in Eq. (2.4) are recursive. ED is a rigid measure since it does not allow local shifting on the time axis and both time series must be the same length.

OSB can be viewed as extension of DTW (though OSB also computes a bijection of matched sequences). To see this, observe that the edge weight dtw for DTW is defined as

$$dtw((i,j)(k,l)) = \begin{cases} d(a_k, b_l) & \text{if } i < k \wedge j < l \wedge ((k-i)+(l-j) = 1 \vee 2) \\ \infty & \text{otherwise} \end{cases}$$

This means that if i maps to j, then either $k = i$ maps to $l = j + 1$ or $k = i + 1$ maps to $l = j + 1$ or $k = i + 1$ maps to $l = j$ in the DTW correspondence. Thus, in comparison to DTW, OSB allows penalized jumps. The jump penalty is included in the edge cost w, which is a distinctive feature of OSB. Analogous to DTW, the edge cost w of OSB can be easily extended to impose a warping window constraint, i.e., we can limit the number of elements that can be jumped over in one step by setting a maximal value for the index differences $k - i - 1$ and $l - j - 1$.

For DTW, some nonnegative, local dissimilarity function d must be defined for every pair of elements a_i and b_j, but that is the only restriction on the distance space. Thus DTW is suitable to use on many types of sequences: univariate, multivariate, continuous, nominal, etc., as long as an appropriate d is defined. In practice though, for univariate time series, the L_1-norm is

used, i.e. $d(a_i, b_j) = |a_i - b_j|$. The DTW distance between two time series, denoted $DTW(a, b)$, is defined in Eq. (2.5) (where $d(a_i, b_j)$ is assumed to be the L_1-norm). Like DTW, OSB may use any pairwise, nonnegative dissimilarity function d to define the distance space between elements, though in all of our experiments we use the squared L_2-norm between the original elements, which is known to strengthen the influence of outliers. This allows us to better identify the outliers and to jump over them.

LCSS does not optimize directly on the distance between the sequences but instead on the length of the optimal subsequence, thus there is no "distance" between two elements. Instead, if two elements match (according to some threshold ϵ), then the subsequence length is increased by 1. Thus, the length of the optimal subsequence between a and b, denoted $LCSS(a, b)$ is computed as in Eq. (2.7). To compute the distance between the two sequences, use $1 - LCSS(a, b)/min(m, n)$.

For the two edit distance measures defined in (2.8) and (2.9), the only edit operation supported is deletion of an element from a sequence, but it is treated as an insertion of a null element into the other sequence. This null element is indicated by Λ. For Edit Distance with Real Penalty (ERP) [12], the distance between two elements is defined as

$$ED(a,b) = \sum_{i}^{m} |a_i - b_i|^2 \tag{2.4}$$

$$DTW(a,b) = \begin{cases} 0 & \text{if } m = n = 0 \\ \infty & \text{if } m = 0 \vee n = 0 \\ |a_m - b_n| + \min\{DTW(a_{1:i}, b_{1:j}) & \text{otherwise} \\ \quad |i = m-1 \vee m,\ j = n-1 \vee n, \\ \quad i + j < m + n\} \end{cases} \tag{2.5}$$

$$OSB(\bar{a},\bar{b}) = \begin{cases} 0 & \text{if } m = n = 0 \\ \infty & \text{if } m = 0 \vee n = 0 \\ d_{osb}(a_{m+1}, b_{n+1}) + \min\{OSB(a_{0:i}, b_{0:j}) & \text{otherwise} \\ \quad + (|m-i| + |n-j|) \cdot C \\ \quad |i = 0 : m+1,\ j = 0 : n+1, \\ \quad i + j < m + n + 2\} \end{cases} \tag{2.6}$$

$$LCSS(a,b) = \min \begin{cases} 0 & \text{if } m = 0 \vee n = 0 \\ 1 + LCSS(a_{1:m-1}, b_{1:n-1}) & \text{if } |a_m - b_n| < \epsilon \\ \max\{LCSS(a_{1:m-1}, b), LCSS(a, b_{1:n-1}) & \text{otherwise} \end{cases} \tag{2.7}$$

$$ERP(a,b) = \begin{cases} \sum_1^n |b_i - g| & \text{if } m = 0 \\ \sum_1^m |a_i - g| & \text{if } n = 0 \\ \min\{ERP(a_{1:m-1}, b_{1:n-1}) + d_{erp}(a_m, b_n), & \text{otherwise} \\ \quad ERP(a_{1:m-1}, b) + d_{erp}(a_m, \Lambda), \\ \quad ERP(a, b_{1:n-1}) + d_{erp}(\Lambda, b_n)\} \end{cases} \tag{2.8}$$

$$TWED(a,b) = \begin{cases} 0 & \text{if } m = n = 0 \\ \infty & \text{if } m = 0 \vee n = 0 \\ \min\{TWED(a_{1:m-1}, b_{1:n-1}) + d_{twed}(a_m, b_n), & \text{otherwise} \\ \quad TWED(a_{1:m-1}, b) + d_{twed}(a_m, \Lambda), \\ \quad TWED(a, b_{1:n-1}) + d_{twed}(\Lambda, b_n)\} \end{cases} \tag{2.9}$$

$$d_{erp}(a_i, b_j) = \begin{cases} |a_i - b_j| & \text{if } a_i, b_j \text{ match} \\ |a_i - g| & \text{if } b_j = \Lambda \\ |b_j - g| & \text{if } a_i = \Lambda \end{cases} \tag{2.10}$$

where g is a constant gap penalty. The authors of ERP use $g = 0$ and give two justifications: firstly, when $g = 0$, the distance between sequences a and b corresponds to the difference between the area under the curve of a and the area under the curve of b; and secondly, then $\sum_i a_i = \sum_j a'_j$ where a is the original sequence and a' is the transformed sequence.

Unlike DTW, LCSS, and ERP, Time Warp Edit Distance (TWED) uses the time stamps of the elements, along with the values of the elements, when computing the distance between two elements. For TWED, the distance between two elements is defined as

$$d_{twed}(a_i, b_j) = \begin{cases} d_{match} & \text{if } a_i, b_j \text{ match} \\ d_{dela} & \text{if } b_j = \Lambda \\ d_{delb} & \text{if } a_i = \Lambda \end{cases} \tag{2.11}$$

with

$$\begin{aligned} d_{match} &= dist(a_i, b_j) + dist(a_{i-1}, b_{j-1}) + \\ &\qquad \nu \cdot (|t_{a_i} - t_{b_j}| + |t_{a_{i-1}} - t_{b_{j-1}}|) \\ d_{dela} &= dist(a_i, a_{i-1}) + \nu \cdot (t_{a_i} - t_{a_{i-1}}) + \lambda \\ d_{delb} &= dist(b_j, b_{j-1}) + \nu \cdot (t_{b_j} - t_{b_{j-1}}) + \lambda \end{aligned}$$

where $dist(a_i, b_j)$ is any L^p-norm, $\lambda \geq 0$ is a constant penalty for deletion, $\nu \geq 0$ is a constant that characterizes the stiffness of the elasticity, and $|t_{a_i} -$

$t_{b_j}|$ is the time-stamp difference of elements a_i and b_j respectively. Using Eq. (2.11), the formal definition of the TWED distance, denoted $TWED(a, b)$, is given in Eq. (2.9).

Of the six time series distance measures shown here, only ED, ERP, and TWED are metrics, but as discussed in Section 1.1.1 and in [26, 32], distances induced by human judgement are often non-metric, so non-metric distance spaces may better represent the actual space.

2.4 OSB Parameters and Time Complexity

There are up to four parameters for OSB. The first is the *warping window* which is similar to the warping window for DTW. The warping window constant r restricts the distance between matching elements. In other words, element a_i from the query sequence can only match to an element b_j from the target sequence if $|i-j| < r$. The next two OSB parameters we call *query skip* q and *target skip* t, which bound the number of elements that can be skipped in the query and target, respectively. In the experimental results in Section 2.6 we set $r = 5$, $q = 5$, and $t = 5$ for all data sets. We do not train these three parameters for any of our experiments, since OSB seems to be insensitive to setting of these parameters. In contrast, the performance of DTW is evaluated with respect to the best possible warping window r restriction.

The last parameter for OSB is the *jump cost*, the penalty for skipping

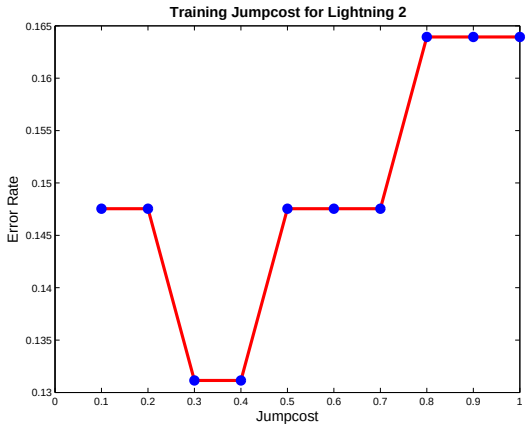


Fig. 2.3: An example of training the jump cost **C** and the resulting error rates for the data set *Lightning-2*.

elements, and is the only parameter that we train. Because all the sequences are normalized before matching, we can train in discrete steps from 0 to 4 using multiscale steps of 1.0, then 0.2, then 0.01, then 0.001. Fig. 2.3 shows an example of training the jump cost for the data set *Lightning-2*. As discussed in Section 1.1.1 and as seen in the figure, the resulting error rates are not very very sensitive to changes in the value of **C**. Though the strategy for finding the jump cost is very simple and the results are very good, the best jump cost is found empirically. How to choose the optimal jump cost is an open question that we will be addressing in the future.

It is well known that the time complexity of DTW is $O(mn)$ for two sequences with m and n elements respectively. Since OSB considers all possible jumps, its time complexity is $O(m^2n^2)$. It is also well known that in practice DTW is run with a warping window restriction r. Because the value of r can

be quite significant (up to 10% of m), the time complexity of DTW with this warping window restriction is $O(mr)$. We also impose the warping window restriction r on OSB. In addition, we can limit the number of elements that can be jumped over in one step by setting maximal values for the index differences q and t. With these restrictions, the time complexity of OSB is $O(mrqt)$. Since we set $r = 5$, $q = 5$, and $t = 5$ in all our experiments, our practical time complexity of OSB is $O(125m)$.

2.5 Implementation Details and Comparison to OSB-07

This paper is an extension of work that was published in [31] (which we will call OSB-07). Though OSB-07 performed quite well, there were some weaknesses. The first improvement we make is that in Eq. (2.2) we now use a weighted L_1 distance between the vertices in the distance matrix instead of a weighted Euclidean distance. There are two reasons for this. The first is that the L_1 distance has a natural interpretation: it is the number of elements skipped in the query sequence plus the number of elements skipped in the target sequence. The Euclidean distance has no such natural interpretation. The second reason is that when compared to the L_1 distance, the Euclidean distance gives too much weight to large differences in the values of

the indices [19]. This favors moving along the diagonal, which in our case corresponds to skipping elements in both sequences. Having simultaneous outliers in both sequences is much less likely than having outliers in only one sequence, and the Euclidean distance is more apt to skip elements in both the query and target sequences, instead of just skipping the outlier elements in one sequence. Therefore, L_1 distance is preferred.

A simple example is given in Fig. 2.4 where the numbers in the squares are the indices in the distance matrix. Assume that the value in $(4, 4)$ equals the value in $(2, 5)$, so that the choice of matching an element in the query sequence with one in the target sequence lies solely on the distance between the indices of the matrix. The Euclidean distance between $(1, 1)$ and $(4, 4)$ is 2.83, and between $(1, 1)$ and $(2, 5)$ is 3.0, thus query element 4 would match with target element 4, skipping elements 2 and 3 in both sequences. On the other hand, the L_1 distance between $(1, 1)$ and $(4, 4)$ is 4.0, and between $(1, 1)$ and $(2, 5)$ is 3.0, thus query element 2 would match with target element 5, skipping elements only in the target sequence.

The second improvement of OSB over OSB-07 is that OSB-07 required that either the first element of the query or the first element of the target be matched, and also that the last element of the query be matched with a target element. We have removed this restriction in OSB; now the first few and last few elements of both the query and target may be skipped. This ability to

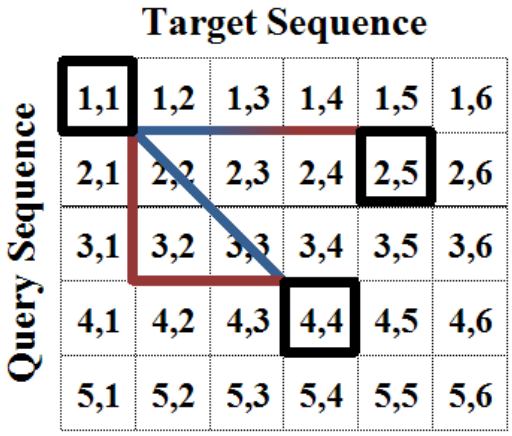


Fig. 2.4: The L_1 distance (in red) versus Euclidean distance (in blue).

skip the first and last elements of the sequences of course makes OSB more robust to outliers.

There is one exception to allowing the beginning and ending elements of the sequences to be skipped. When one sequence is significantly shorter than the other sequence, then the first and last elements of the query must participate in the matching, otherwise there is too much flexibility and the likelihood of accidental matches increases. As stated in Section 2.4, we put an upper bound on the number of adjacent elements to search for a matching element (warping window r) and an upper bound on the number of elements in the target that can be skipped (target skip t). Let m be the length of the query sequence and n be the length of the target sequence. Then if $m + r + t < n$, i.e., the query is a partial sequence and significantly shorter than the target, we force the first and last elements of the query to match with an element in the target

Input : Sequences a and b, $warpWin$, $querySkip$, $targetSkip$, $jumpCost$
Output: $optDist$ - the distance between the optimal subsequences

```
1  m ← Length(a);
2  n ← Length(b);
   // Create original distance matrix
3  for i ← 1 to m do
4  |   for j ← 1 to n do
5  |   |   d(i, j) ← Distance(a_i, b_j);
6  |   end
7  end
   // Create augmented distance matrix by adding an extra row to
      beginning and end
   // and an extra column to beginning and end.  Then set very first and
      very last
   // element to zero
8  dist(1 : m + 2, 1 : n + 2) ← Inf;
9  dist(2 : m + 1, 2 : n + 1) ← d(1 : m, 1 : n);
10 dist(0, 0) ← 0;
11 dist(m + 2, n + 2) ← 0;
   // Initialize weight matrix
12 weight(1 : m + 2, 1 : n + 2) ← Inf;
13 weight(1, 1 : n + 2) ← dist(1, 1 : n + 2);
14 weight(1 : m + 2, 1) ← dist(1 : m + 2, 1);
15 for i ← 1 to m + 1 do // Index over rows
   |   // Find index of last query element that may be skipped
16 |   stopRowJump = Min(i + querySkip, m + 2);
17 |   for j ← 1 to n + 1 do // Index over cols
   |   |   // Find index of last target element that may be skipped
18 |   |   stopColJump = Min(j + targetSkip, n + 2);
   |   |   // Difference between i and j must be smaller than warpWin
19 |   |   if Abs(i-j)≤ warpWin then
20 |   |   |   for rowJump ← i + 1 to stopRowJump do // Second index over
   |   |   |   rows
21 |   |   |   |   for colJump ← j + 1 to stopColJump do // Second index over
   |   |   |   |   cols
   |   |   |   |   |   // We favor shorter jumps by multiplying by jumpCost
22 |   |   |   |   |   newWeight = weight(i, j) + dist(rowJump, colJump) +
   |   |   |   |   |   [(rowjump − i − 1) + (k − j − 1)] ∗ jumpCost;
23 |   |   |   |   |   if newWeight < weight(rowJump, colJump) then
24 |   |   |   |   |   |   weight(rowJump, colJump) = newWeight;
25 |   |   |   |   |   end
26 |   |   |   |   end
27 |   |   |   end
28 |   |   end
29 |   end
30 end
31 optDist = weight(m + 2, n + 2);
```

Algorithm 1: Calculating the OSB for Two Sequences a and b

sequence. In this case, to find the distance between the optimal subsequences, two changes need to be made to Algorithm 1. To force the first element of the query to participate in the matching, change line 17 to **for** $j \leftarrow i$ **to** $n+1$ **do** . To force the last element of the query to participate in the matching, change line 31 to $optDist = \text{Min}(weight(m, 2:n))$.

In [31], we reported the performance of OSB-07 on nineteen of the UCR data sets [28]. OSB improves the results of OSB-07 on sixteen of the data sets by as much as 17.7 percentage points, with a mean improvement of 4.0 percentage points. On one data set (*Two Patterns*) the performance is identical, and on two data sets (*Face All* and *Coffee*) OSB does worse than OSB-07 (by 3.4 and 3.6 percentage points respectively). On partial sequences from the MPEG-7 data set [30], the performance of OSB-07 and OSB is similar.

2.6 Experimental Results

To demonstrate the effectiveness of OSB compared to other similarity measures (both metric and non-metric), we test OSB on the benchmark time series data sets published on the UCR Time Series Classification/Clustering Page [28] and on the MPEG-7 Core Experiment CE-Shape-1 data set [30]. These data sets are online to serve the data mining/machine learning community as an effort to encourage reproducible research for time series classification and clustering, and for shape matching.

2.6.1 The UCR Time Series Data Sets Results

The UCR Time Series data repository has available twenty data sets from various domains. The time series lengths range from 60 to 637 and the number of classes in a data set ranges from 2 to 50. Each data set is divided into a fixed training set and testing set. The number of examples in a training set ranges from 24 to 1000, and the number of testing examples ranges from 28 to 6174. See Table 2.1 for the characteristics of the data sets. For these twenty data sets, we perform classification based on the distance-wise 1-nearest-neighbor (1NN). For each time-series in the test set, we calculate its distance to every time series in the training set. The label of the training sequence that is closest to the test sequence is assigned to the test sequence. To calculate the error rate of the classifier, let c be the number of test time series correctly labeled, and t be the total number of test time series. Then $errorrate = 1 - c/t$; hence the smaller the value, the better the result. The table published on [28] reports the 1NN classification results for three methods: Euclidean distance (ED), Dynamic Time Warping (DTW), and DTW with best scoring warping window (DTW-WW). The table published in [35] also reports the 1NN classification results for Longest Common Subsequence (LCSS), Edit Distance with Real Penalty (ERP), and optimized Time Warp Edit Distance (OTWED). The results comparing these six classifiers to OSB is shown in Table 2.1.

The table shows that OSB outperforms the other six distance measures on

DATASET	**Number of Classes**	**Size of Training Set**	**Size of Testing Set**	**Time Series Length**	**ED**	**DTW WW**	**DTW**	**LCSS**	**ERP**	**OTWED**	**OSB**
Synthetic Control	6	300	300	60	0.120	0.017	**0.007✓***	0.047	0.036	0.023	0.020
Gun-Point	2	50	150	150	0.087	0.087	0.093	**0.013✓**	0.040	**0.013✓**	0.020
CBF	3	30	900	128	0.148	0.004	**0.003✓**	0.009	**0.003✓**	0.007	0.004
Face (all)	14	560	1690	131	0.286	0.192	0.192	0.201	0.202	**0.189✓***	0.190
OSU Leaf	6	200	242	427	0.483	0.384	0.409	**0.202✓***	0.397	0.248	0.409
Swedish Leaf	15	500	625	128	0.213	0.157	0.210	0.117	0.120	0.102	**0.085✓***
50 Words	50	450	455	270	0.369	0.242	0.310	0.213	0.281	**0.187✓***	0.257
Trace	4	100	100	275	0.240	0.010	**0.000✓***	0.020	0.170	0.050	0.030
Two Patterns	4	1000	4000	128	0.090	0.002	**0.000✓**	**0.000✓**	**0.000✓**	0.001	**0.000✓**
Wafer	2	1000	6174	152	0.005	0.005	0.020	**0.000✓***	0.009	0.004	0.001
Face (four)	4	24	88	350	0.216	0.114	0.170	0.068	0.102	**0.034✓***	0.045
Lightning2	2	60	61	637	0.246	**0.131✓**	**0.131✓**	0.180	0.148	0.213	**0.131✓**
Lightning7	7	70	73	319	0.425	0.288	0.274	0.452	0.301	0.247	**0.192✓***
ECG	2	100	100	96	0.120	0.120	0.230	**0.100✓**	0.130	**0.100✓**	**0.100✓**
Adiac	37	390	391	176	0.389	0.391	0.396	0.425	0.378	0.376	**0.358✓***
Yoga	2	300	3000	426	0.170	0.155	0.164	0.137	0.147	**0.130✓***	0.142
Fish	7	175	175	463	0.267	0.233	0.267	0.091	0.120	**0.051✓***	0.103
Beef	5	30	30	470	0.467	0.467	0.500	0.533	0.500	0.533	**0.433✓***
Coffee	2	28	28	286	0.250	**0.179✓**	**0.179✓**	0.214	0.250	0.214	0.286
OliveOil	4	30	30	570	0.133	0.167	0.133	0.800	0.167	0.167	**0.100✓***
Total Number of Best Scores per Method					0	2	6	5	2	7	8
Total Number of UNIQUELY Best Scores per Method					0	0	2	2	0	5	5

Table 2.1: The 1NN classification results for various time series distance measures on the UCR data sets. Bolded, checked results indicate best scores. Asterisks indicate uniquely best scores.

eight of the twenty data sets, and is uniquely the best on five. The next best performer is OTWED, which is best on seven of the data sets and uniquely best on five. DTW is best on six of the data sets and uniquely best on two. LCSS achieves the lowest error rate on five data sets and is uniquely the lowest on two. ERP and DTW-WW are the best on two data sets each, though neither have uniquely best scores. Of the five data sets where OSB has the uniquely best performance, OSB improves the error rate on *Lightning 7* by 5.5 percentage points with an error rate of 0.192, OSB improves the error rate of *Beef* and *Olive oil* by more than 3 percentage points with error rates of 0.433 and 0.100 respectively, and it improves the error rate of *Swedish Leaf* and *Adiac* by more than 1.5 percentage points with error rates of 0.085 and 0.358 respectively.

As these experimental results clearly demonstrate, OSB has the best overall performance, although it is not always superior to DTW, LCSS, ERP, or OTWED. Consequently, skipping outlier elements as done by OSB can significantly improve matching results for many real data sets.

2.6.2 The MPEG7 Data Set Results

MPEG-7 is widely used to test shape classification and retrieval methods. It contains 1400 binary images consisting of 70 object classes (e.g. "Bird") and within each class there are 20 shapes, for a total of 1400 shapes. For our

experiments, each shape is represented with 100 equidistant sample points on the contour, and these points are converted into sequences by calculating the curvature of each point with respect to its five neighbors on each side. This yields 1400 sequences of real numbers, each of length 100. This particular transformation makes the sequence representation invariant to rotation and scale changes. In other words, the shape of a cell phone with its antenna pointing up can still match with the same cell phone shape scaled and rotated so that is now smaller and the antenna is pointing down. To obtain our results, we compute the distance between every shape in the data set and all other shapes. Because in the shape domain both classification and retrieval are important, we report the number of shapes from the same class (based on the computed distance) among the 1, 5, 10, and 20 most similar shapes. In addition, we report the bull's-eye retrieval rate. Among the 40 most similar shapes for each query, the bulls-eye retrieval rate is the ratio of the total number of shapes from the same class to the highest number possible (which is 1400×20), thus the best possible score is 1.0.

Because the starting point of the shape contour is not known when calculating the representative sequence, one sequence representing say a horse may begin on the nose, while another horse sequence may begin on the tail. For this reason, for all experiments reported in this section, we use a sliding window of size 5 when calculating the distance between two shapes. To use the sliding

	OSB $\mathbf{C = 0.03}$	**LCSS** $\epsilon = 0.45$	**DTW** $r = 3$
1NN	**0.963✓**	0.955	0.912
5NN	**0.872✓**	0.847	0.780
10NN	**0.779✓**	0.752	0.678
20NN	**0.651✓**	0.627	0.557
Bulls-eye	**0.724✓**	0.719	0.624

Table 2.2: The retrieval results on the MPEG-7 data set for various distance measures. Bolded, checked results indicate best scores.

window, we first calculate the distance between the query and the elements 1 - 100 of the target shape. Then we circularly shift the target sequence by 5 points to the left and again calculate the distance between the query and target, and so on until the target has been circularly shifted 20 times. Of these 20 distances between a query and target pair, we choose the minimum distance.

Full Sequences

In Table 2.2 we report the results of OSB, LCSS, and DTW when both the query sequence and the target sequence are of length 100. Each method has one parameter to train: the jump cost $\mathbf{C}$ for OSB, the threshold ϵ for LCSS, and the warping window r for DTW. The best value for each parameter is shown in the table. As can be seen from the table, OSB outperforms LCSS and DTW

	Full-length Targets			Targets using Corresp. Window			
	OSB	**LCSS**	**DTW**	**OSB**	**LCSS**	**DTW**	**ED**
1NN	**1.00✓**	0.40	0.10	**1.00✓**	0.80	0.50	0.70
5NN	**0.86✓**	0.26	0.06	**0.86✓**	0.70	0.38	0.66
10NN	**0.82✓**	0.28	0.05	**0.82✓**	0.59	0.33	0.46
20NN	**0.69✓**	0.23	0.04	**0.69✓**	0.47	0.28	0.31
Bulls-eye	**0.79✓**	0.33	0.09	**0.79✓**	0.57	0.37	0.40

Table 2.3: The retrieval results on the MPEG-7 data set for ten partial query sequences. Bolded, checked results indicate best scores.

in all classification and retrieval results. OSB attains a 1NN classification rate of 96.3% while the next best, LCSS, attains a 1NN classification rate of 95.5%, and DTW attains 91.2%. For the bull's-eye score, OSB has a 72.4% retrieval rate, while LCSS and DTW have a retrieval rate of 71.9% and 62.4% respectively.

Partial Sequences

The goal of this section is to test OSB's ability as a measure for partial sequence similarity. Our query sequences represent significant contour parts of shapes in the MPEG-7 data set. We manually select ten query sequences as contour segments representing shapes from ten different classes. They are represented as the black parts of the contours in the first column in Fig. 2.5. The number of points used for each partial sequence ranges from 30 to 57. We then convert the query segments into sequences by calculating the curvature of

each point with respect to its 5 neighbors on each side, giving query sequences of length 10 less than the original number of points for each segment.

We compare each partial query sequence against all 1400 targets of the data set. We also use the same query and target sequences with three other algorithms: LCSS, DTW, and ED. We compute the distances between the queries and targets in two ways. First, we compare each query against targets of length 100 (full-length), using the sliding window described above. The first three columns of results in Table 2.3 report the retrieval rates using these full-length targets. As seen in the results, this evaluation puts DTW at a severe disadvantage since it cannot skip any elements in the target, and therefore all 100 elements of the target must participate in the matching of the 20 to 47 elements of the query. In order to eliminate this problem, we also compute the distances using a correspondence window. If the length of the partial query sequence is m, we use only the first $m + 10$ elements of the target sequence in the correspondence window. Because we also use the sliding window, the query does get compared to the entire target, and again we choose the minimum distance among the 20 distances computed for the query/target pair. The results using the correspondence window are reported in the next three columns of results in Table 2.3. The last column reports the results for ED on the partial query sequences. Since ED requires that the query and target sequence be the same length, the size of the correspondence window matches

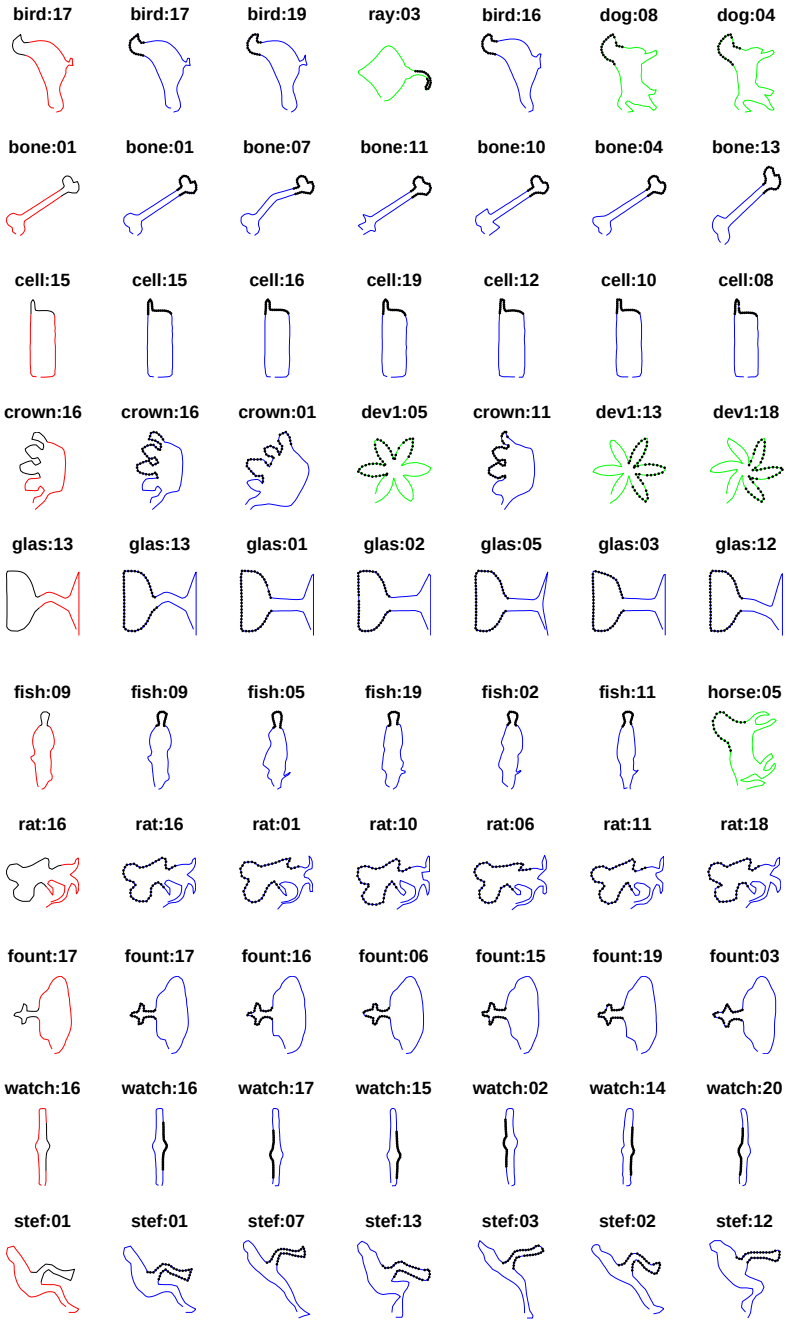


Fig. 2.5: OSB results on the MPEG-7 data set for 10 partial query sequences on full-length targets. Each contour is identified by label:id. The first column shows the query part (in black) in its original contour (red). Columns 2 through 7 are the top six matches out of the 1400 target shapes. The black points on the targets indicate the corresponding points as computed by OSB.

the length of the query.

To avoid bias in the classification results, we exclude from the 1NN to 10NN results any self-matching (see Fig. 2.5 for the top six matches). Because OSB always matches a partial sequence to its original sequence, the 1NN result that we report in Table 2.3 is actually the second best match; the 5NN result is actually the number of correct classifications from the second to sixth best match; and the 10NN results come from the second to eleventh best matches. We use this reporting method for DTW, LCSS, and ED also. From the table, it is clear that OSB performs significantly better than any of the other algorithms. OSB is the only algorithm to achieve 100% retrieval rate, with or without correspondence window, in the 1NN classification (refer to third column in Fig. 2.5); using a correspondence window, LCSS's best result is 80%, DTW's is 50%, and ED's is 70%. OSB has an 86% retrieval rate for 5NN, which is 16 percentage points better than LCSS, 48 percentage points better than DTW, and 20 percentage points better than ED. What is extremely interesting is that OSB has the exact same results whether using the full-length targets or the correspondence window. For both LCSS and DTW, their performance is significantly improved by using the correspondence window. LCSS's 1NN rate goes from 40% to 80% after using the window, and DTW goes from 10% to 50%, but neither can compare with the results obtained by OSB.

Fig. 2.5 shows, for each of the partial sequence queries, the 6 most similar shapes after running OSB. It can be observed that the performance of OSB is actually better than is reflected by the classification based on class labels. For example, the query contour segment representing a bird head in the first row matches to two dog heads, which actually does have a very similar shape in these contours. The best match for each segment (shown in the second column) is the same shape from which we take the query segment. This holds for all the queries and is not a trivial result. Because of the sliding window of size 5 we use to position the starting point of the target sequence, the query and target sequences are not identical. In fact, DTW without the correspondence window can find only 1 of the 10 queries' original shapes, DTW with the window is able to find only 5 of the queries' original shapes, and ED can find only 6 of the 10. LCSS, like OSB, is able to find all 10 original query shapes.

We highlight that the performance of OSB for 5NN retrieval is 80 percentage points higher than the performance of DTW without a correspondence window, and using a correspondence window, OSB is 48 percentage points higher than that of DTW, 16 percentage points higher than that of LCSS, and 20 percentage points higher than ED. It may be possible to further improve the performance of LCSS, DTW, and ED, by varying the size of the correspondence window, but this would add one more parameter to train and increase the time complexity of the algorithms. The excellent performance

of OSB illustrates that such a window size parameter is not needed, since OSB computes a bijective embedding of the query sequence. This fact clearly demonstrates that OSB is more suitable for partial sequence similarity.

CHAPTER 3

Ghost Points

3.1 Introduction

In many applications, only distance (or equivalently similarity) information is available, in which case operations in vector space cannot generate synthetic points. This is the case when the data points do not have any coordinates, or the data points have coordinates but the Euclidean distance does not reflect their structure. Consequently, a distance measure is used that is not equivalent to the Euclidean distance, e.g., [31, 7]. For this type of data, researchers usually utilize embeddings to low dimensional Euclidean spaces. However, embedding implies distance distortion. It is known that not every four point metric space can be isometrically embedded into an Euclidean space $\mathbb{R}^k$, e.g., see [36].

Definition 3.1. A *metric* on a set X is a distance function $\rho : X \times X \to \mathbb{R}$,

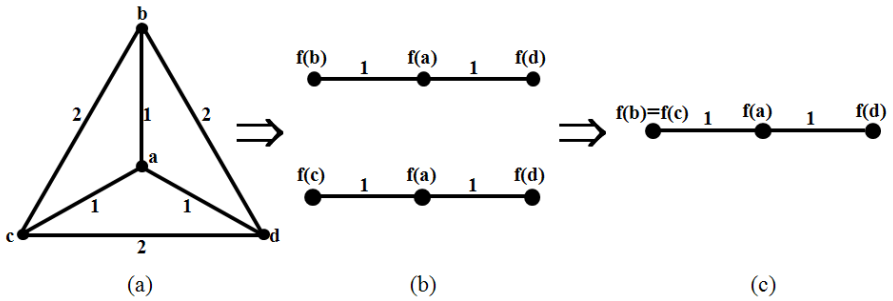


Fig. 3.1: (a) is an example of a 4-point metric space that cannot be embedded into a Euclidean space. (b) shows that the points b, a, and d are collinear after embedding, and so are the points c, a, and d. Thus points b and c are the same point after embedding as shown in (c).

such that the following axioms hold:

1. $\rho(x, y) \geq 0$ (non-negativity)

2. $\rho(x, y) = \rho(y, x)$ (symmetry)

3. $\rho(x, y) = 0 \Leftrightarrow x = y$ (positive definiteness)

4. $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$ (triangle inequality)

for any $x, y, z \in X$.

Definition 3.2. A *metric space* is an ordered pair (X, ρ), where X is a set of points, and ρ is metric on X, that is, a distance function $\rho : X \times X \to \mathbb{R}$.

Definition 3.3. Let Y and Z be two metric spaces. We say that a mapping f of the space Y into Z is an *isometric embedding* if $dist_Z(f(y_1), f(y_2)) = dist_Y(y_1, y_2)$.

A simple example where distances are not preserved when mapping a four point metric space to $\mathbb{R}^k$ is presented in [20]. Given the metric space (X, ρ) defined in Figure 3.1a, assume there exists a mapping $f : X = \{a, b, c, d\} \to \mathbb{R}^k$ for some k where f preserves the distances. The triangle inequality holds for the elements a, b, and d; in fact $\rho(b, d) = \rho(b, a) + \rho(a, d)$ and because of the equality, the mapped points $f(b), f(a)$, and $f(d)$ are collinear in the space $\mathbb{R}^k$. This also holds for points a, c, and d, i.e., they are collinear in $\mathbb{R}^k$ (Figure 3.1b). Since both lines have two points in common, they must be the same line (Figure 3.1c). But then $f(b) = f(c)$ contradicting the fact that the original distance between b and c is 2. Therefore the assumption that f preserves the distances is false.

Definition 3.4. In this paper, a *distance space* is an ordered pair (X, ρ), where X is a set of points and $\rho : X \times X \to \mathbb{R}$ is a distance function that satisfies the first two axioms and the $\Leftarrow$ direction of axiom 3 from Definition 3.1.

Clearly, we would like ρ to be as close as possible to a metric, but this is not always possible, e.g., there are clear arguments from human visual perception that the distances induced by human judgments are often non-metric [32].

3.2 Definition of Ghost Points

The key observation of the proposed approach is that although not every four point metric space can be embedded into a Euclidean space, every three point metric space can be isometrically embedded into the plane $\mathbb{R}^2$. Let (Δ, ρ), where $\Delta = \{x, a, b\} \subseteq X$, be a metric space with three distinct points. Then it is easy to map Δ to the vertices of a triangle on the plane. For example, we can construct an isometric embedding $h : \Delta \to \mathbb{R}^2$ by setting $h(a) = (0, 0)$ and $h(b) = (\rho(a, b), 0)$. Then $h(x)$ is uniquely defined as a point with nonnegative coordinates such that its Euclidean distance to $h(a)$ is $\rho(x, a)$ and its Euclidean distance to $h(b)$ is $\rho(x, b)$. $h : \Delta \to \mathbb{R}^2$ is an isometric embedding, since for any two points $y, z \in \Delta$, $\rho(y, z)^2 = ||y - z||^2$, where $|| \cdot ||$ is the standard L_2 norm that induces the Euclidean distance on the plane. We stress that this construction does not require that (X, ρ) be a metric space, but it does require that the three point space (Δ, ρ) be a metric space. Below we will generalize this construction to the case when Δ is not a metric space.

Definition 3.5. Given any two points a, b in a distance space X, we define a *ghost point* e induced by a and b using the construction $e = \mu(a, b) = h^{-1}(\frac{1}{2}(h(a) + h(b))$. For every $x \in X$, the distance from x to e, $\rho(x, \mu(a, b))$, is computed as follows:

1. If the three point subspace $\Delta = \{x, a, b\}$ is a metric, then use Equa-

tion 3.1 below.

2. If $\rho(a,b) > \rho(x,a) + \rho(x,b)$, then use Equation 3.2 below.

3. (a) If $\rho(x,a) > \rho(x,b) + \rho(a,b)$, then use Equation 3.3 below or

   (b) If $\rho(x,b) > \rho(x,a) + \rho(a,b)$, then use Equation 3.4 below.

Cases 2 and 3 in this definition apply when Δ is not a metric space.

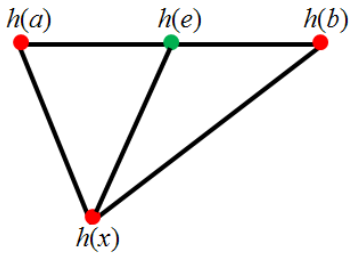
Let $\mu(a,b)$ denote the mean of two points a, b. If $a, b \in \mathbb{R}$, then we have the usual formula $\mu(a,b) = \frac{1}{2}(a+b)$ (see Figure 3.2a, where red points are original data, the green point e is the ghost point and $e = \mu(a,b)$).

Our first key contribution is the definition of $\mu(a,b)$ for any two points a, b in a distance space X. To define $\mu(a,b)$, we need to specify $\rho(x, \mu(a,b))$ for every $x \in X$. There are three cases depending on whether the three point subspace $\Delta = \{x, a, b\} \subseteq X$ is a metric or not.

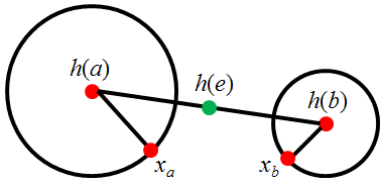
Case 3.1. Type1: $\Delta = \{x, a, b\} \subseteq X$ is a metric subspace

We first isometrically embed Δ into the plane $\mathbb{R}^2$ by h. We define $\mu(a,b) = h^{-1}(\frac{1}{2}(h(a) + h(b))$. Since $h(\Delta)$ defines vertices of a triangle on the plane, we can easily derive that

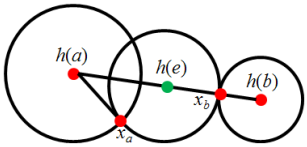
$$||h(x) - \frac{h(a)+h(b)}{2}||^2 = \frac{||h(x)-h(a)||^2}{2} + \frac{||h(x)-h(b)||^2}{2} - \frac{||h(a)-h(b)||^2}{4}$$



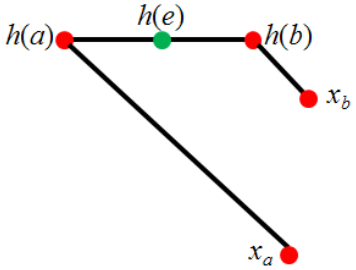
(a) Type 1 ghost points.



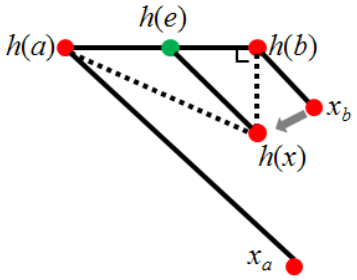
(b) Type 2 ghost points



(c) Type 2 ghost points



(d) Type 3 ghost points



(e) Type 3 ghost points

Fig. 3.2: (a) shows the construction of $\rho(x, e)$ for $e = \mu(a, b)$ for a triple of points that satisfy the triangle inequality. (b) shows a triple of points that cannot construct a triangle. The way to calculate $\rho(x, e)$ for (b) is shown in (c). Another way in which the triangle inequality is violated is shown in (d) and the approach to calculating $\rho(x, e)$ is shown in (e).

Since h is an isometry and $\mu(a,b) = h^{-1}(\frac{1}{2}(h(a) + h(b))$, we obtain (see Figure 3.2a)

$$\rho(x,\mu(a,b))^2 = \frac{1}{2}\rho(x,a)^2 + \frac{1}{2}\rho(x,b)^2 - \frac{1}{4}\rho(a,b)^2 \tag{3.1}$$

Consequently, Equation 3.1 defines the distance of every point $x \in X$ to the new point $\mu(a,b)$, which we call the mean of a and b. By computing the distances of $\mu(a,b)$ to all points in X, we define a new point $\mu(a,b)$, and the augmented set $X' = X \cup \{\mu(a,b)\}$ is also a distance space. We stress that to add a new point $\mu(a,b)$ to X we do not need to compute the embedding h. We use h only to derive Equation 3.1. Moreover, since the embedding h is an isometry, Equation 3.1 defines locally correct distances from $\mu(a,b)$ to all points in X. Since we can compute the correct distances without explicitly computing the mapping h, this is similar to the kernel trick [2].

Case 3.2. Type 2: $\Delta = \{x,a,b\} \subseteq X$ is not a metric subspace and $\rho(a,b) > \rho(x,a) + \rho(x,b)$

In Equation 3.1 we assume that the three point space (Δ,ρ) is a metric space. Thus, we assume that the local structure of any distance space X can be locally approximated by the metric space, which is also the assumption for embedding approaches [41, 46]. However, for some point triples $\Delta = \{x,a,b\} \subseteq X$, (Δ,ρ) is not a metric space, which may lead to a negative distance in Equation 3.1. This is the case if $\rho(a,b) > \rho(x,a) + \rho(x,b)$. Then a

triangle with vertices $h(a)$, $h(b)$, and $h(x)$ cannot be constructed on the plane, as illustrated in Figure 3.2b. Since a single point $h(x)$ on the plane does not exist, we map $h(x)$ to two different points denoted x_a and x_b such that $\rho(x,a) = ||h(a) - x_a||$ and $\rho(x,b) = ||h(b) - x_b||$. Without loss of generality we assume that $\rho(x,a) > \rho(x,b)$. Then it is possible to position points x_a and x_b on the plane such that (see Figure 3.2c): $\rho(x,a) = ||h(a) - x_a||$, $\rho(x,b) = ||h(b) - x_b||$, and $||h(\mu(a,b)) - x_a|| = ||h(\mu(a,b)) - x_b||$.

Thus, both points x_a and x_b are the same distance away from $h(\mu(a,b))$, and this distance is equal to $\frac{1}{2}||h(a) - h(b)|| - ||x_b - b||$. Therefore, we define $h(x) = \{x_a, x_b\}$ and

$$\rho(x, \mu(a,b)) = \frac{1}{2}\rho(a,b) - \rho(x,b) \tag{3.2}$$

Formally, h maps x to a single point in a quotient space $\mathbb{R}^2/\{x_a, x_b\}$, and h remains an isometric embedding but to the quotient space.

Case 3.3. Type 3: $\Delta = \{x, a, b\} \subseteq X$ is not a metric subspace and either $\rho(x,a) > \rho(x,b) + \rho(a,b)$ or $\rho(x,b) > \rho(x,a) + \rho(a,b)$

In this case, as in Case 3.2, (Δ, ρ) is not a metric space and again may lead to a negative distance in Equation 3.1. This occurs if either $\rho(x,a) > \rho(x,b) + \rho(a,b)$ or $\rho(x,b) > \rho(x,a) + \rho(a,b)$. Then a triangle with vertices $h(a), h(b), h(x)$ cannot be constructed on the plane, as illustrated in Figure 3.2d. Since a single point $h(x)$ on the plane does not exist, we again map

$h(x)$ to two different points denoted x_a and x_b such that $\rho(x,a) = ||h(a) - x_a||$ and $\rho(x,b) = ||h(b) - x_b||$.

Without loss of generality we assume that $\rho(x,a) > \rho(x,b) + \rho(a,b)$. In this case, we first position point x_b on the plane so that the angle $h(a)h(b)x_b$ is straight without changing the distance from $h(b)$ to x_b (see Figure 3.2e). Then we use the triangle $h(a)h(b)x_b$ to define the ghost point. When doing so we ignore the distance $\rho(x,a)$ in this construction or equivalently, only consider the assignment $h(x) = x_b$. Unlike Case 3.2, it is impossible to make the assignments $h(x) = x_a$ and $h(x) = x_b$ consistent, hence we need to ignore one of them. Since $\rho(x,b)$ is significantly smaller than $\rho(x,a)$, and small distances are less likely to be the result of noise, we rely only on $h(x) = x_b$. We can then use the right triangle $h(a)h(b)x_b$ to define

$$\rho(x,\mu(a,b))^2 = \rho(x,b)^2 + \frac{1}{4}\rho(a,b)^2. \tag{3.3}$$

Similarly, if $\rho(x,b) > \rho(x,a) + \rho(a,b)$, we define

$$\rho(x,\mu(a,b))^2 = \rho(x,a)^2 + \frac{1}{4}\rho(a,b)^2. \tag{3.4}$$

The so defined distances to ghost points are guaranteed to be nonnegative and symmetric by their construction. Hence the space augmented by ghost points remains a distance space. However, it may happen that two different points have distance zero, and this is possible even if X is a metric space. For example, assume that X is a sphere of radius 1 and that points a and b

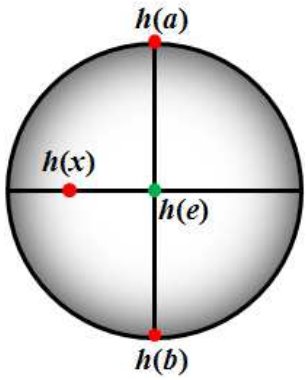


Fig. 3.3: Example of a unit sphere where $\rho(h(e), h(x)) = 0$.

are on the north and south poles (see Figure 3.3). For any point $x \in X$ on the equatorial line the distance between $\mu(a, b)$ and x becomes $\rho(x, \mu(a, b))^2 = 0.5(\pi/2)^2 + 0.5(\pi/2)^2 - 0.25\pi^2 = 0$. Therefore, every point on the equatorial line has a distance of 0 to the ghost point $\mu(a, b)$. This example also shows that adding ghost points to a metric space may lead to a non-metric space. We stress however that the intended application of the proposed method is to densify distance spaces that are non-metric, since such spaces are common in many cognitively motivated tasks such as distances between images, shapes, text documents, and so on. We also stress that though global metricity is not necessary, local metricity is preferred. If the triple of points $a, b,$ and x is close to a metric, then the embedding of the three points is uniquely defined and Equation 3.1 can be used to calculate the distance between x and the ghost point.

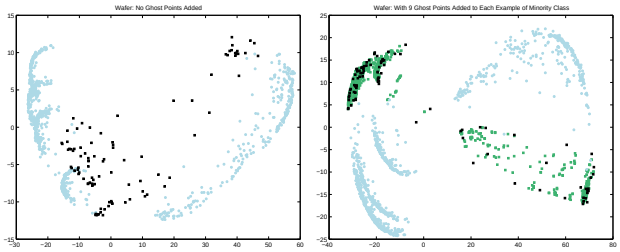
If the space X is finite, i.e., $X = \{x_1, \ldots, x_n\}$, then the distance function

$\rho : X \times X \rightarrow \mathbb{R}_{\geq 0}$ is represented by a square matrix $M_\rho(X)$. Each row of the square distance matrix $M_\rho(X)$ is the distance of one data point x to all data points in the data set, i.e., for all $y \in X$, $M_\rho(x, y) = \rho(x, y)$. The matrix for $X \cup \{\mu(a, b)\}$ is obtained by simply adding one row and one column to $M_\rho(X)$, with each entry computed using Equations 3.1, 3.2, 3.3, or 3.4.

Thus, the proposed approach can be applied to metric and non-metric distance spaces, and our construction guarantees that the distances to all ghost points are nonnegative and symmetric. In Section 3.4.3, we show the results of experiments that count the number of Type 1, Type 2, and Type3 computations performed on eighteen data sets using distance spaces induced by two different distance functions, OSB and DTW.

3.3 Visualizing Data

High-dimensional data, such as time series, are often hard to visualize, though visualization can help in the analysis of trends, periodicity, motifs, and the like. When the actual time series sequences are available, line graphs can be a very effective tool to visualize and analyze time series. One of the earliest known time series plot is of planetary orbits from a tenth century monastery [47]. There have been some advances in the visualization of time series (for example, [54]), but the line graph is still the most prevalent. But if instead of sequences, the data is represented as a distance space (pair-wise



(a) (b)

Fig. 3.4: After using PCA on the distance matrix to reduce the dimensionality from 1000 to 2, (a) the 1000 examples are plotted (the majority class as gray circles and the minority class as black squares). (b) is the same data set with 9 ghost points (green squares) added per minority example. In (a), it is impossible to distinguish the minority class from the majority class as the minority class has no structure. However, in (b) there are 5 distinct clusters, 2 of which belong to the minority class. (Best viewed in color.)

distances between each time series), then the visualization becomes much more difficult as line graphs are not sufficient. In addition, even when plotted in a graph, if the data set is imbalanced, the minority class is often undiscernible among all the points of the majority class.

Adding ghost points to the minority class before plotting can change the structure of the underlying points so that minority class clusters become visible. For example, Figure 3.4 shows the Wafer training set before and after adding ghost points to the distance matrix induced by Optimal Subsequence Bijection (OSB). The training set has 903 samples of the majority class and 97 samples of the minority class for a total of 1000 samples. To create Figure 3.4a, we first take the original 1000×1000 distance matrix and use principal component analysis (PCA) to reduce the dimensionality to two dimensions.

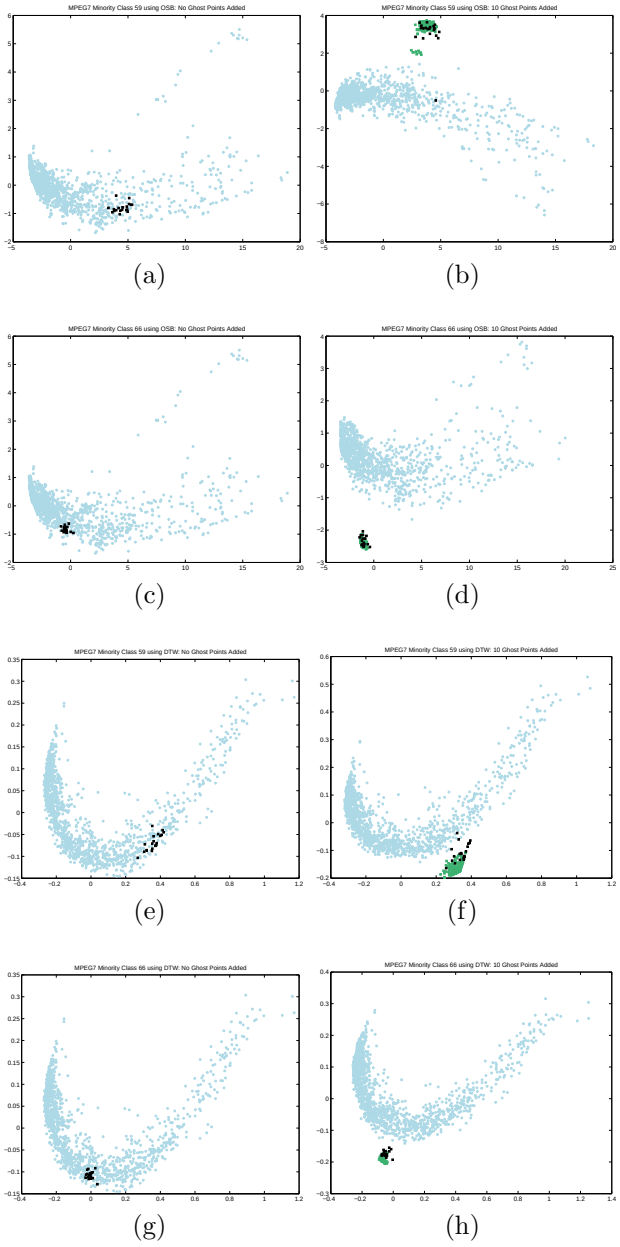


Fig. 3.5: The MPEG-7 data set. Rows 1 and 2 use the distance matrix induced by OSB, rows 3 and 4 by DTW; rows 1 and 3 show the minority class rat and rows 2 and 4 show the minority class teddy. Column 1 shows the data set without ghost points; column 2 after adding ghost points. The 1400 examples of the data set are plotted (majority class as gray circles, minority class as blue squares, and ghost points as green squares). In the first row, without ghost points, it is impossible to distinguish the minority class from the majority class as the minority class completely overlaps the majority class. However, in the second row the minority class now forms a distinct cluster.

For Figure 3.4b we add 9 ghost points per minority sample to the distance matrix (to create a 1873×1873 matrix) and again run PCA. The majority class is plotted as light blue circles, the minority class as black squares, and the ghost points as green squares. In Figure 3.4a, without ghost points, it is impossible to distinguish the minority class from the majority class since the minority class forms no cluster and many of the minority class points overlap the majority class clusters. In Figure 3.4b, after ghost points are added to the training set, the underlying shape of the data changes to form five discernable clusters. It is clear that two of the clusters belong to the minority class (the upper-left cluster and the lower-right cluster).

The next four examples are from the MPEG-7 image data set (see Section 3.4.3 for a description of the data set). In Figure 3.5, the first two rows show the MPEG-7 data set before and after adding ghost points to the distance matrix induced by OSB for two different minority classes, *rat* and *teddy*; the second two rows show the MPEG-7 data set before and after adding ghost points to the distance matrix induced by Dynamic Time Warping (DTW) for the same two minority classes, *rat* and *teddy*. We divide the data set into 1380 samples of a majority class (69 classes of 20 images each, collapsed into a single class) and 20 samples of a minority class for a total of 1400 samples. As before, for Figures 3.5a, 3.5c, 3.5e, and 3.5g we take the original 1400×1400 distance matrix and use PCA to reduce the dimensionality to two and then

plot the 1400 points. For Figures 3.5b, 3.5d, 3.5f, and 3.5h we add 10 ghost points for each of the 20 minority class samples to the distance matrix (creating a 1600×1600 matrix) and again run PCA. Again, a majority class is plotted as light blue circles, a minority class as black squares, and the ghost points as green squares. In Figures 3.5a, 3.5c, 3.5e, and 3.5g, without ghost points, the minority class points overlap the majority class points and cannot be differentiated from the majority class points. In Figures 3.5b, 3.5d, 3.5f, and 3.5h, after ghost points are added to the data set, the underlying shape of the minority class changes to form distinct and visible clusters, with very few points, if any, overlapping the majority class points.

3.4 Experimental Evaluation

In many real-world situations, the minority class, the class with the fewest examples, is by far the most important class. Take for example the Mammography data set [58], which consists of non-calcification (non-cancerous) and calcification (cancerous) examples. The data set has 11183 examples of which only 260 (2.32%) are examples of cancer. A trivial classifier that classifies all examples as non-cancerous will achieve an accuracy of 97.68%, though its error rate for the minority class is 100%. For this data set, there are also uneven costs associated with misclassifying a normal example and misclassifying a cancerous example. If a healthy patient is incorrectly diagnosed with hav-

ing breast cancer, there is a cost associated with this error (fear, unnecessary tests) but eventually the misdiagnosis will be found. On the other hand, if a patient who does have breast cancer is incorrectly diagnosed as being healthy, then the cost could be her life since she will not get appropriate treatment. When the performance on the minority class is as important or more important than overall accuracy, other performance measures must be used. A common measure is F_β-*measure* [48] which is defined below in Section 3.4.1.

Unlike other techniques that add synthetic points, ghost points have the advantage that they can be added in distance space. To show that they will work with different distance measures, we use both Dynamic Time Warping (DTW) and Optimal Subsequence Bijection (OSB) as distance measures on the UCR Time Series data sets [28] and on the MPEG-7 Core Experiment CE-Shape-1 data set [30] for our experiments.

The UCR Time Series data repository has available 20 data sets from various domains. The time series lengths range from 60 (Synthetic Control) to 637 (Lightning-2) and the number of classes in a data set ranges from 2 to 50. Each data set is divided into a fixed training set and testing set. The number of examples in a training set ranges from 24 (FaceFour) to 1000 (Wafer), and the number of testing examples ranges from 28 (Coffee) to 6174 (Wafer). In our experiments, we use seventeen of the data sets and their characteristics are described in Table 3.1.

Data Set	Number of Classes	Number of Classes Used as Minority Class	Training Set			Testing Set		
			Total Number of Examples	Number of Minority Examples	Number of Majority Examples	Total Number of Examples	Number of Minority Examples	Number of Majority Examples
SyntheticControl	6	6	300	50	250	300	50	250
CBF	3	2	30	8 - 10	20 - 22	900	300 - 302	598 - 600
FaceAll	14	14	560	40	520	1690	8 - 287	1403 - 1682
OSULeaf	6	6	200	15 - 53	147 - 185	242	23 - 55	187 - 219
SwedishLeaf	15	15	500	26 - 42	458 - 474	625	33 - 49	576 - 592
50Words	50	49	450	2 - 52	398 - 448	455	1 - 57	398 - 454
Trace	4	4	100	21 - 31	69 - 79	100	19 - 29	71 - 81
TwoPatterns	4	4	1000	237 - 271	729 - 763	4000	959 - 1035	2965 - 3041
Wafer	2	1	1000	97	903	6174	665	5509
FaceFour	4	4	24	3 - 8	16 - 21	88	14 - 26	62 - 74
Lightning2	2	1	60	20	40	61	28	33
Lightning7	7	7	70	5 - 19	51 - 65	73	6 - 19	54 - 67
ECG	2	1	100	31	69	100	36	64
Adiac	37	37	390	4 - 15	375 - 386	391	6 - 16	375 - 385
Fish	7	7	175	21	147 - 154	175	22 - 29	146 - 153
Beef	5	5	30	6	24	30	6	24
OliveOil	4	3	30	4 - 8	22 - 26	30	4 - 9	21 - 26

Table 3.1: The characteristics of the 17 UCR data sets used in our experiments.

	Predicted Positive	Predicted Negative
Actual Positive	**TP**	**FN**
Actual Negative	**FP**	**TN**

Table 3.2: Confusion Matrix.

MPEG-7 is a standard data set and is widely used to test shape classification and retrieval methods. It contains 1400 binary images consisting of 70 object classes (e.g. "Rats") and within each class there are 20 shapes, for a total of 1400 shapes. For our experiments, each shape is represented with 100 equidistant sample points on the contour, and these points are converted into sequences by calculating the curvature of each point with respect to its five neighbors on each side. This yields 1400 sequences of real numbers, each of length 100. This particular transformation makes the sequence representation invariant to rotation and scale changes. In other words, the shape of a cell phone with its antenna pointing up can still match with the same cell phone shape scaled and rotated so that the phone is now smaller and its antenna is pointing down.

3.4.1 Evaluating Performance

Most studies on the class imbalance problem concentrate on two-class problems since multi-class data sets can easily be reduced to two classes (see Sec. 3.4.2). In an imbalanced data set, one class, the *majority* class or the *negative* class, has many examples, while the other class, the *minority* class or *positive* class, has few examples. These imbalances in real world data sets can be 2:1, 1000:1, or even 10000:1. When a data set is imbalanced, the usual forms of evaluating performance do not work. For classification, generally the overall accuracy (the fraction of examples that are correctly classified) or the error rate ($1 - accuracy$) is reported, but this does not have much value if the interest lies in the minority class. It has been empirically shown that accuracy can lead to poor performance for the minority class [57]. Another problem with using accuracy as the performance metric is that different classification errors are given the same importance, whereas in actuality their costs might differ significantly. One solution commonly used is to have a weighted loss function with higher loss for the minority class [8], but it requires knowing the loss weights, which is often impossible in real applications.

For imbalanced data sets when the minority class is the important class, performance metrics borrowed from the information retrieval community [48] are often used. They are based on a *confusion matrix* (see Table 3.2), that reports the number of true positives (**TP**), true negatives (**TN**), false positives

(**FP**), and false negatives (**FN**). These are then used to define metrics that evaluate the performance of a learner on the minority class, such as *recall*, *precision*, and F_β*-measure*. The formulas for these metrics are given below. The precision of a class (Equation 3.6) is the number of TPs divided by the total number of examples predicted as positive. A precision score of 1.0 means that every example predicted as a positive example *is* a positive example, though there may be some positive examples that were labeled as negative. The recall of a class (Equation 3.5) is the number of TPs divided by the number of examples that are actually positive. A recall score of 1.0 means that every positive example is labeled correctly, though some negative examples may have also been labeled as positive. There is always a trade-off between precision and recall, but for data sets where the cost of false negatives is high, a high recall value is preferable. The F_β-measure [48] (Equation 3.8) is the weighted harmonic mean of precision and recall and merges recall and precision into a single value. The best F_β score is 1 and the worst is 0. The β parameter controls the relative weight given to recall and precision. F_β "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision" [48]. If correct classification of the minority class is important, when false negatives have similar costs to false positives, then the F_1-measure ($\beta = 1$) is used because precision and recall are weighted equally. When the cost of false negatives is more than that of false positives,

then the F_2-measure ($\beta = 2$) is better because it weights recall twice as heavily as precision.

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.6}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.7}$$

$$F_\beta = (1 + \beta^2)\frac{Recall \times Precision}{\beta^2 \times Precision + Recall} \tag{3.8}$$

3.4.2 Methodology

Of the twenty-one data sets we have available (twenty UCR data sets and the MPEG-7 data set), only three have training sets that contain a true minority class (a two-class data set with one class comprising less than 35% of the total number of examples). These data sets are Wafer, Lightning-2, and ECG. In order to evaluate ghost points further, we also create artificially imbalanced data sets. To create artificial minority classes for the fourteen data sets from the UCR repository that have more than two classes, we take each class that comprises less than 35% of the total number of examples as a minority class, and then collapse the remaining classes into one. If in a data set there is more than one class that meets our criteria as a minority class, we treat each class as minority class in turn and average the results. For the MPEG-7 data set, to create a training and testing set, we randomly choose ten shapes from

each class (for a total of 700 shapes) for the training set, and the remaining ten shapes from each class (again 700 shapes total) become the testing set. Then we take each class as a minority class in turn, collapse the remaining 69 classes into one class, and average the results over the 70 minority classes. See Tables 3.3, 3.4, and 3.5 for a summary of the results.

Once we create a minority class in the training set, we add ghost points to the minority class of the training and testing sets and perform classification in the following manner:

1. The training set

   (a) Given a training set consisting of m time series examples with sequence length s, create the $m \times m$ distance matrix by calculating the OSB or DTW distance between each pair of examples.

   (b) For each minority class example x, add k-many ghost points by inserting one ghost point between x and each of its $k$nn. This gives us a total of p new points.

   (c) Calculate the distance from the p ghost points to every other point in the training set using Equation 3.1, 3.2, 3.3, or 3.4; we now have an $(m+p) \times (m+p)$ matrix.

   (d) Convert both the original and augmented OSB or DTW score matrix to affinity matrices using the approach in [60] and Equation 3.9.

(e) Use these affinity matrices as the *user-defined* or *precomputed* kernels for the SVM to get two models: one that includes ghost points and one that does not.

(f) Run SVM to train.

2. The testing set

(a) Given a testing set consisting of n time series examples with sequence length s, and a training set consisting of m time series of length s, create the $n \times m$ OSB or DTW distance score matrix.

(b) Calculate the distance from each test data point to each of the p ghost points using Equation 3.1, 3.2, 3.3, or 3.4; we now have an $n \times (m + p)$ distance matrix.

(c) Convert both the original and augmented OSB or DTW score matrix to an affinity matrix as in step 1d above.

(d) Use these affinity matrices as the *user-defined* or *precomputed* kernels for the SVM as in step 1e above.

(e) Run SVM to test.

There are two critical parameters to set when we convert the distance matrices to kernels that modify the σ for the Gaussian Kernel function, A and K. As stated in [62], the scaling parameter σ is some measure of when two points are considered similar. We use the method in [60] to calculate the

local scaling parameter σ_{ij} for each pair of data points x_i and x_j. The affinity between a pair of points can be written as:

$$k(x_i, x_j) = exp(\frac{-d(x_i, x_j)^2}{\sigma_{ij}}) \tag{3.9}$$

where $\sigma_{ij} = A \cdot mean\{k\text{nn } d(x_i), k\text{nn } d(x_j)\}$, $mean\{k\text{nn } d(x_i), k\text{nn } d(x_j)\}$ is the the mean distance of the K-nearest neighbors of points x_i, x_j, and A is an extra scaling parameter. For the SVM, there is a third parameter to set, which is the cost parameter C. For all UCR experiments we used $A = 0.5$, $K = 5$, and $C = 0.5$ and for all MPEG-7 experiments we used $A = 0.36$, $K = 25$, and $C = 0.5$. For each of the eighteen data sets, we run SVM on the four matrices (after converting them to kernels): OSB score matrix without ghost points; OSB score matrix with ghost points; DTW score matrix without ghost points; and DTW score matrix with ghost points.

The final parameter to set is the number of ghost points to add per minority example, as the final results can be sensitive to the number of ghost points added. Two good heuristics are: 1. to balance the the classes and 2. add one ghost point per minority example, but neither of these always give the best results. The strategy we use in our experiments is a modified version of balancing the classes. Let m be the number of minority examples in the training set, n be the number of majority examples, and k be the maximum number of ghost points to add per minority example such that $k \cdot m = n$.

We then choose the number of ghost points per example $g \in \{1, \ldots, k\}$ that gives the best results. Though the strategy for finding g is very simple and the results are very good, g is found empirically. How to choose the *optimal* number of ghost points is an open question that we will be addressing in the future.

3.4.3 Results

Of the eighteen data sets we test, only three of the training sets had natural minority classes. For the other fifteen, we created artificial minority classes, and if necessary, averaged the results. See Section 3.4.2 for the methodology we used to create the imbalanced data sets. We compare the results of SVM on OSB with and without ghost points on the UCR data sets in Table 3.3, the results of SVM on DTW with and without ghost points on the UCR data sets in Table 3.4, and finally, the results of SVM on OSB and DTW on the MPEG-7 data set in Table 3.5. Because we are interested in the performance on minority classes, specifically minimizing the number of false negatives, we measure the overall accuracy (Equation 3.7), the F_1-measure (Equation 3.8 with $\beta = 1$) which weights precision and recall equally, and the F_2-measure (Equation 3.8 with $\beta = 2$) which weights recall twice as heavily as precision.

As the results show in Table 3.3, for the OSB score matrix on the UCR data sets, adding ghost points improve SVM's overall accuracy rate on sixteen

Data Set	Num GP Added per Minority Example	Overall Accuracy		F_1-Measure Minority Class		F_2-Measure Minority Class	
		SVM	SVM-GP	SVM	SVM-GP	SVM	SVM-GP
SyntheticControl	2	98.83%	**99.78%** ✓	0.9668	**0.9933**✓	0.9843	**0.9913**✓
CBF	1	96.89%	**98.56%** ✓	0.9498	**0.9779**✓	0.9283	**0.9661**✓
FaceAll	1	98.83%	**99.26%** ✓	0.9063	**0.9396**✓	0.9307	**0.9391**✓
OSULeaf	2	86.16%	**87.05%** ✓	0.3689	**0.5315**✓*	0.3294	**0.4923**✓*
SwedishLeaf	8	98.27%	**99.11%** ✓	0.8552	**0.9384**✓	0.8145	**0.9400**✓*
50Words	1	98.78%	**98.95%** ✓	0.3236	**0.4662**✓*	0.2785	**0.4162**✓*
Trace	2	91.50%	**96.75%** ✓*	0.7921	**0.9336**✓*	0.7484	**0.9301**✓*
TwoPatterns	2	99.78%	**99.96%** ✓	0.9954	**0.9991**✓	0.9927	**0.9986**✓
Wafer	5	96.25%	**99.81%** ✓*	0.7913	**0.9910**✓*	0.7060	**0.9937**✓*
FaceFour	1	91.19%	**96.88%** ✓*	0.7902	**0.9394**✓*	0.7356	**0.9235**✓*
Lightning2	1	73.77%	**83.61%** ✓*	0.6190	**0.8000**✓*	0.5159	**0.7463**✓*
Lightning7	1	89.63%	**93.54%** ✓*	0.4522	**0.7234**✓*	0.3970	**0.6920**✓*
ECG	1	87.00%	**93.00%** ✓*	0.7869	**0.8955**✓*	0.7101	**0.8571**✓*
Adiac	3	98.07%	**98.29%** ✓	0.4416	**0.6246**✓*	0.3765	**0.5758**✓*
Fish	3	94.86%	**97.39%** ✓	0.7550	**0.9065**✓*	0.6857	**0.8888**✓*
Beef	1	**82.67%** ✓	81.33%	0.1667	**0.3418**✓*	0.1667	**0.3105**✓*
OliveOil	1	91.11%	**94.44%** ✓*	0.5708	**0.7454**✓*	0.5429	**0.7019**✓*

Table 3.3: The results of adding ghost points to the OSB distance scores on the imbalanced UCR time series data sets. Bolded, checked results indicate best scores. An asterisk for accuracy indicates at least 3 percentage points difference; for F_β-measure it indicates at least 10 percentage points difference.

Data Set	Num GP Added per Minority Example	Overall Accuracy		F_1-Measure Minority Class		F_2-Measure Minority Class	
		SVM	SVM-GP	SVM	SVM-GP	SVM	SVM-GP
SyntheticControl	2	97.44%	**99.28%** ✓	0.9292	**0.9788**✓	0.9683	**0.9814**✓
CBF	1	95.83%	**97.72%** ✓	0.9337	**0.9644**✓	0.9171	**0.9445**✓
FaceAll	1	96.08%	**97.56%** ✓	0.7306	**0.8444**✓*	0.7916	**0.8372**✓
OSULeaf	2	85.12%	**86.98%** ✓	0.3450	**0.4775**✓*	0.3085	**0.4317**✓*
SwedishLeaf	8	97.94%	**98.71%** ✓	0.8294	**0.9071**✓	0.7913	**0.9105**✓*
50Words	1	98.76%	**98.97%** ✓	0.3109	**0.4719**✓*	0.2717	**0.4173**✓*
Trace	2	90.25%	**95.50%** ✓*	0.7690	**0.9090**✓*	0.7171	**0.8995**✓*
TwoPatterns	2	98.45%	**99.08%** ✓	0.9680	**0.9812**✓	0.9528	**0.9704**✓
Wafer	5	96.82%	**99.69%** ✓	0.8299	**0.9858**✓*	0.7595	**0.9880**✓*
FaceFour	1	83.52%	**92.33%** ✓*	0.5152	**0.8351**✓*	0.4638	**0.7900**✓*
Lightning2	1	77.05%	**83.61%** ✓*	0.6818	**0.7917**✓*	0.5859	**0.7197**✓*
Lightning7	1	90.02%	**90.80%** ✓	0.4411	**0.5877**✓*	0.4213	**0.5807**✓*
ECG	1	82.00%	**84.00%** ✓	0.7097	**0.7419**✓	0.6471	**0.6765**✓
Adiac	3	97.74%	**98.05%** ✓	0.4188	**0.6264**✓*	0.3887	**0.6221**✓*
Fish	3	93.55%	**95.76%** ✓	0.7077	**0.8450**✓*	0.6502	**0.8243**✓*
Beef	1	**82.00%** ✓	81.33%	0.1667	**0.3084**✓*	0.1667	**0.3002**✓*
OliveOil	1	85.56%	**91.11%** ✓*	0.4000	**0.7264**✓*	0.3714	**0.7254**✓*

Table 3.4: The results of adding ghost points to the DTW distance scores on the imbalanced UCR time series data sets. Bolded, checked results indicate best scores. An asterisk for accuracy indicates at least 3 percentage points difference; for F_β-measure it indicates at least 10 percentage points difference.

Distance Measure	Characteristics: Num GP Added per Minority Example	Number of Minority Examples	Number of Majority Examples	Overall Accuracy: SVM	Overall Accuracy: SVM-GP	F_1-Measure Minority Class: SVM	F_1-Measure Minority Class: SVM-GP	F_2-Measure Minority Class: SVM	F_2-Measure Minority Class: SVM-GP
OSB	4	10	1390	99.43%	**99.74%** ✓	0.7097	**0.8974** ✓*	0.6616	**0.8681** ✓*
DTW	5	10	1390	99.11%	**99.20%** ✓	0.6027	**0.7673** ✓*	0.5805	**0.7940** ✓*

Table 3.5: The results of adding ghost points to the OSB and DTW distance scores on the MPEG-7 data set. Bolded, checked results indicate best scores. An asterisk for accuracy indicates at least 3 percentage points difference; for F_β-measure it indicates at least 10 percentage points difference.

of the seventeen data sets. In fact, four of the data sets, Trace, FaceFour, Lightning-2, and ECG, have increases in overall accuracy of over 5 percentage points. On all seventeen of the data sets data sets, the F_1-measure and the F_2-measure improve with ghost points by as much as 29.5 percentage points; twelve data sets see an increase of at least 10 percentage points in the F_1-measure and thirteen data sets in the F_2-measure. For the Lightning-7 data set, adding ghost points increases the accuracy by 3.9 percentage points, the F_1-measure by 27.1 percentage points, and the F_2-measure by 29.5 percentage points. The overall accuracy of Lightning-2 has the largest increase when ghost points are added, an increase of 9.8 percentage points, while the F_1-measure and F_2-measure increase by 18.1 and 23 percentage points, respectively. The Beef data set, which is the only data set in Table 3.3 that decreases in overall accuracy when ghost points are added (by 1.3 percentage points), still gains in the F_1-measure, which increases by 17.5 percentage points, and the F_2-measure, which increases by 14.4 percentage points.

When using the DTW score matrix of the UCR data sets (Table 3.4), adding ghost points increases the overall accuracy again on sixteen of the seventeen data sets; on the Beef data set, the accuracy decreased by 0.7 percentage points. For four of the data sets (Trace, FaceFour, Lightning-2, and OliveOil), ghost points increase the accuracy by over 5 percentage points. The F_1-measure and the F_2-measure increase for all seventeen data sets when ghost

points are added. Twelve data sets see an increase of at least 10 percentage points in both the F_1-measure and the F_2-measure. Two data sets (FaceFour and OliveOil) see an increase of over 30 percentage points in both their F_1-measure and the F_2-measure. The data set with the largest gain in its F_1-measure F_2-measure is OliveOil; it gains 32.6 and 35.4 percentage points respectively. The accuracy rate for OliveOil also increases by 5.6 percentage points with ghost points. The data set FaceFour, which has the greatest accuracy gain (8.8 percentage points) also has an increase in its F_1-measure of 32 percentage points and in its F_2-measure of 32.6 percentage points. The only data set, Beef, where ghost points actually decrease the overall accuracy (by 0.7 percentage points), still has an impressive gain in the F_1-measure and the F_2-measure; 14.2 and 13.4 respectively.

For the MPEG-7 data set (Table 3.5), the results are similar to those discussed above. With both OSB and DTW, all measures increase. With OSB on the MPEG-7 data set and ghost points, the overall accuracy increases 0.3 percentage points, the F_1-measure by 18.8 percentage points, and the F_2-measure by 20.6 percentage points. The increases of the results for DTW on the MPEG-7 data set are similar; overall accuracy increases by 0.1 percentage points, the F_1-measure by 16.5 percentage points, and the F_2-measure by 21.3 percentage points.

As discussed in Section 3.2, computing the distance of a ghost point to

Data Set	Num Gp Added per Minority Example	Number of Type 1 Distance Computations	Number of Type 2 Distance Computations	Number of Type 3 Distance Computations	Total Number of Distance Computations per Minority Class	Percentage of Type 1	Percentage of Type 2	Percentage of Type 2
SyntheticControl	1	29,389	10	1,826	31,225	94%	0%	6%
CBF	1	7,796	4	607	8,407	93%	0%	7%
FaceAll	1	67,809	48	22,923	90,780	75%	0%	25%
OSULeaf	2	26,738	42	5,126	31,905	84%	0%	16%
SwedishLeaf	7	182,556	261	107,177	289,993	63%	0%	37%
50Words	1	5,014	7	3,360	8,381	60%	0%	40%
Trace	2	1,871	34	9,352	11,256	17%	0%	83%
TwoPatterns	2	1,500,368	270	1,124,450	2,625,087	57%	0%	43%
Wafer	6	1,840,370	2,577	2,495,572	4,338,519	42%	0%	58%
FaceFour	1	551	4	134	689	80%	1%	19%
Lightning2	1	1,828	16	766	2,610	70%	1%	29%
Lightning7	1	741	9	734	1,484	50%	1%	49%
ECG	1	4,229	7	2,429	6,665	63%	0%	36%
Adiac	3	15,615	126	9,467	25,208	62%	1%	38%
Fish	3	13,706	65	15,282	29,053	47%	0%	53%
Beef	1	56	17	301	375	15%	5%	80%
OliveOil	1	235	2	118	355	66%	1%	33%
MPEG-7	4	51,195	23	5,561	56,780	90%	0%	10%

Table 3.6: Using OSB to induce the distance matrices on the eighteen data sets tested, we count the types of distance computations made from the ghost points to every other point in the distance space (see Section 3.2). Note that due to averaging over multiple minority classes and rounding, some numbers do not add up.

Data Set	Num Gp Added per Minority Example	Number of Type 1 Distance Computations	Number of Type 2 Distance Computations	Number of Type 3 Distance Computations	Total Number of Distance Computations per Minority Class	Percentage of Type 1	Percentage of Type 2	Percentage of Type 2
SyntheticControl	1	30,982	0	243	31,225	99%	0%	1%
CBF	1	8,337	0	70	8,407	99%	0%	1%
FaceAll	6	549,721	13	18,946	568,680	97%	0%	3%
OSULeaf	2	31,609	1	295	31,905	99%	0%	1%
SwedishLeaf	7	267,539	43	22,411	289,993	92%	0%	8%
50Words	1	7,784	0	597	8,381	93%	0%	7%
Trace	2	7,504	15	3,738	11,256	67%	0%	33%
TwoPatterns	2	2,544,129	13	80,946	2,625,087	97%	0%	3%
Wafer	5	3,566,072	274	25,564	3,591,910	99%	0%	1%
FaceFour	1	688	0	2	689	100%	0%	0%
Lightning2	1	2,591	0	19	2,610	99%	0%	1%
Lightning7	2	3,055	0	29	3,085	99%	0%	1%
ECG	2	14,003	0	288	14,291	98%	0%	2%
Adiac	3	22,581	101	2,526	25,208	90%	0%	10%
Fish	2	15,353	3	3,381	18,738	82%	0%	18%
Beef	1	209	0	166	375	56%	0%	44%
OliveOil	1	355	0	0	355	100%	0%	0%
MPEG-7	4	71,101	22	103	71,225	100%	0%	0%

Table 3.7: Using DTW to induce the distance matrices on the eighteen data sets tested, we count the types of distance computations made from the ghost points to every other point in the distance space (see Section 3.2). Note that due to averaging over multiple minority classes and rounding, some numbers do not add up.

the other points in the distance matrix can take one of three forms (see Cases 3.1, 3.2, and 3.3). Tables 3.6 and 3.7 show the number of the different types of computations made for the eighteen data sets using OSB and DTW respectively. It is interesting to note that most of the distance spaces induced by DTW contain very few Type 2 and Type 3 computations. Fifteen of the data sets had less than 8% of non-Type 1 computations, and one of these (OliveOil) had 0%. This indicates that the distance space induced by DTW on these fifteen data sets is very close to a metric space. The remaining three data sets (Trace, Fish, and Beef) had non-Type1 computations of 33%, 18%, and 44% respectively, and thus have distance spaces relatively close to a metric space. On the other hand, the distance spaces induced by OSB are much more variable, where the number of non-Type 1 computations ranges from 6% to 85%. For example, under DTW, the OliveOil data set has 100% Type 1 computations, while only 66% under OSB; the data set Wafer has 99% Type 1 computations under DTW, but only 42% under OSB. Thus OSB is more likely to induce non-metric distance spaces. Though again we state, and our experimental results show, that the application of the proposed method can densify distance spaces that are non-metric, and such spaces are common in many cognitively motivated tasks.

It is clear to see that ghost points increase the overall accuracy for most data sets, and also the F_1-measure and F_2-measure, at times very significantly.

When a data set is imbalanced, and the cost of false negatives is high (but can't be easily quantified), then adding ghost points may significantly reduce the number of false negatives while at the same time increase overall accuracy.

CHAPTER 4

Locally Constrained Diffusion Process on Imbalanced Data Sets

4.1 Introduction

Ghost points can be added to data sets using either supervised or unsupervised learning. In order to insert a single ghost point for a given data point x_i, we need to find another, partner point from the data set $X = \{x_1, \ldots, x_n\}$ to form a pair of points that determine the ghost point. When we know the class labels, ghost points are inserted supervised, i.e., only between data points of the same class. Normally, we would then find the nearest neighbor to x_i

according to some distance measure and insert the ghost point between them. We expect each data point and its nearest neighbors to be able to describe the local structure of the data manifold, which is commonly assumed in manifold learning [41, 46]. However, sometimes the original distance may not be able to describe the data relation correctly. Therefore, we propose a novel strategy to find a partner c_i for a given data point x_i: we use a modified diffusion process to find the nearest neighbor, but insert the ghost point into the original distance space. As the modified diffusion process can improve the underlying structure of the data manifold, it can better describe the relation between data points.

Since the classical diffusion process can be influenced even by moderate noise and outliers, in order to reduce the effect of noisy data points we replace the original diffusion process with a locally constrained diffusion process (LCDP). As we will demonstrate in Section 4.3, it is significantly more robust to noise than the original diffusion process.

4.2 Diffusion Process

Given a set of data points $X = \{x_1, \ldots, x_n\}$, we consider a fully connected graph $G = (X, E)$. The vertices of G are the data points and each edge E is labeled with the strength of the connection $E(i, j) = k(x_i, x_j)$, where k is a kernel function that is symmetric and positivity preserving. In this paper,

given two shapes x_i and x_j, $k(x_i, x_j)$ is defined by applying a Gaussian to the shape distance $\rho(x_i, x_j)$.

From the symmetric graph defined by (X, E), one can construct a reversible Markov chain on X. This is a classic technique in many fields. The degree of each node is defined as

$$D(x_i) = \sum_{j=1}^{n} k(x_i, x_j)$$

and the transition probability is defined as

$$P(x_i, x_j) = \frac{k(x_i, x_j)}{D(x_i)}.$$

It is obvious that the transition matrix P inherits the positivity-preserving property, but it is no longer symmetric. However, we have gained a conservation property:

$$\sum_{j=1}^{n} P(x_i, x_j) = 1$$

From a data analysis point of view, the reason for studying this diffusion process is that the matrix P contains geometric information about the data set X. Indeed, the transitions that it defines directly reflect the local geometry defined by the immediate neighbors of each node in the graph of the data. In other words, $P(x_i, x_j)$ represents the probability of transition in one time step from node x_i to node x_j and it is proportional to the edge-weight $k(x_i, x_j)$. For $t \geq 0$, the probability of transition from x_i to x_j in t time steps is given by $P^t(x_i, x_j)$, which is the tth power P^t of P. One of the main ideas of the diffusion

framework is that the chain running forward in time, or equivalently, taking larger powers of P, allows us to integrate the local geometry and therefore reveals relevant geometric structures of X at different scales, where t plays the role of a scale parameter. In [17], the data points can be embedded into Euclidean space by diffusion maps (DM), which can then reorganize the data points according to their geometric relation as revealed by the diffusion process.

Ideally, diffusion coordinates generated by diffusion maps should reveal the intrinsic geometric structure of the underlying data manifold. However, as we illustrate by the following example, the diffusion process is still sensitive to noise. Our example illustrates that the diffusion process may fail to capture the correct topology if the actual topology of the data manifold is changed because of noise or outliers. Since noise and outliers can influence the distribution of data points, low density areas may become high density areas or vice versa, which will make the transition probability of the diffusion process incorrect. In Fig. 4.1, the samples are taken from a spiral as a function of arc length l with added Gaussian noise and a noise 'bridge' between inner and outer samples. Since the underlying manifold has a 1D structure, we would expect the diffusion process to be able to recover it when we use the coordinates of the second most important eigenvector, as described in [41, 46].

In Figs. 4.1a and (c), we plot the coordinates of the second most important eigenvector as a function of arc length (measured as point index). As can be

clearly observed in Fig. 4.1a, the function from arc length to the second diffusion coordinate is not one-to-one, which means that the intrinsic 1D structure of the spiral has not been recovered by the standard diffusion process. Correspondingly, in Fig. 4.1b, the order of points according to their second diffusion coordinate is color coded. Points with similar color have similar second diffusion coordinates. The fact that the 1D structure is not recovered is shown by the yellow colored points that are present in the bottom left as well as in the top right parts of the spiral. As shown in Figs. 4.1c and (d), the proposed locally constrained diffusion process (Sec. 4.3) is able to recover the 1D structure of the spiral. The graph in (c) does jitter a bit since we approximate the arc length coordinates of the spiral with the point index.

4.3 Locally Constrained Diffusion Process

In the classical diffusion process setting, all paths between nodes x_i and x_j are considered when computing the probability of a walk from x_i to x_j. If there are several noisy nodes, the paths passing through these nodes will affect this probability as we demonstrated in Fig. 4.1.

A solution to this problem is introduced in [45], where a random walk is restricted to the K nearest neighbors of the data points by replacing the original graph G with a K nearest neighbor (KNN) graph G_K that has the edge weights defined as follows: $E_K(i,j) = k(x_i, x_j)$ if x_j belongs to the KNNs

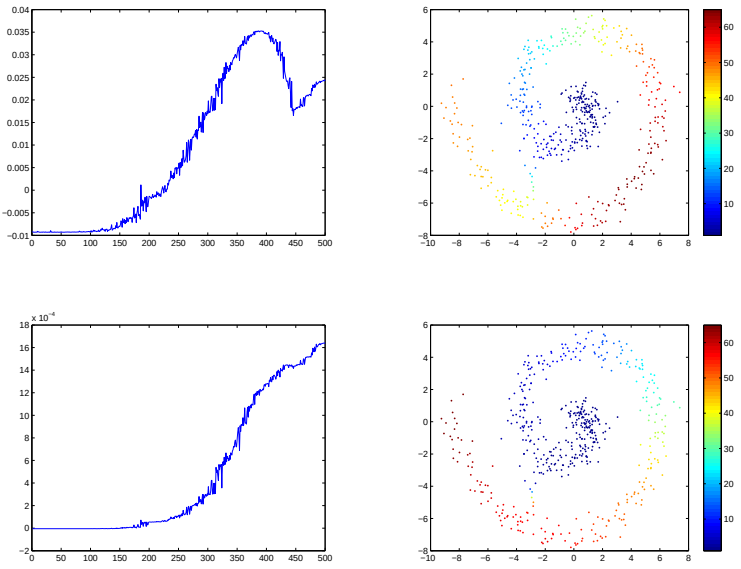


Fig. 4.1: An example comparing the standard diffusion process (DM) to our method (LCDP). (a) is the plot of second most important eigenvector as a function of arc length. (b) shows the points color coded according to their second diffusion coordinate using DM. (c) and (d) show the same plots as (a) and (b) but using LCDP.

of x_i and $E_K(i,j) = 0$ otherwise. Then, the one-step transition probabilities $P_K(x_i, x_j)$ from x_i to x_j are defined

$$P_K(x_i, x_j) = \frac{E_K(i,j)}{\sum_j E_K(i,j)}.$$

Through replacing the P in Section 4.2 by P_K, the effect of noise is reduced, but the process is still not robust enough to noise. The reason is that the relation between the $KNN(x_i)$ and $KNN(x_j)$ is too hard and too narrow. It counts a data point x_k only if x_k is a KNN of both x_i and x_j. This causes problems if both points x_i and x_j belong to the same dense cluster, in which case they may have no common KNNs although they are very similar. In other words, although x_i and x_j are very similar to each other and there are many short paths connecting them in graph G, they may have no common neighbor in G_K.

In order to solve this problem, we consider the paths between KNNs of x_i and KNNs of x_j, which can be viewed as a soft measure of their KNNs' compatibility. The probability of transition from node x_i to x_j is high if all the the paths between points in $KNN(x_i)$ and in $KNN(x_j)$ are short. We define

$$P_{KK}^{t+1}(x_i, x_j) = \sum_{k \in KNN(x_i), l \in KNN(x_j)} P(x_i, x_k) P_{KK}^{t}(x_k, x_l) P(x_l, x_j) \tag{4.1}$$

Eq. 4.1 can be viewed as a symmetric version of the approach in [45], and can

be expressed as matrix multiplication

$$P_{KK}^{t+1} = P_K \; P_{KK}^{t} \; (P_K)^T.$$

The embedding results of our proposed approach on the noisy spiral data are shown in Figs. 4.1c and (d). These figures demonstrate that the proposed locally constrained diffusion process (LCDP) is able to recover the intrinsic geometric structure of the spiral.

4.4 Methodology

We add ghost points to the minority class of the training and testing sets and perform classification in the following method:

1. The training set
   (a) Given a training set consisting of m time series examples with sequence length s, create the $m \times m$ distance matrix by calculating the OSB or DTW distance between each pair of examples.
   (b) The distance matrix is converted to an affinity matrix using the approach in [5]
   (c) Calculate LCDP on the affinity matrix as discussed in Section 4.3
   (d) For each minority class example x, find its Knn in the LCDP matrix. Then using the original distance matrix, add one ghost point between x and each of the Knn found using LCDP. This gives us a total of p new points.
   (e) Calculate the distance from the p ghost points to every other point in the training set using Eq. 3.1. We now have an $(m+p) \times (m+p)$ matrix.
   (f) Convert both the original and augmented OSB or DTW score matrix to affinity matrices using the approach in [5].

(g) Use these affinity matrices as the *user-defined* or *precomputed* kernel for the SVM to get two models: one that includes ghost points and one that does not.

(h) Run SVM to train.

2. The testing set

(a) Given a testing set consisting of n time series examples with sequence length s and the training set consisting of m time series examples with sequence length s, create the $n \times m$ OSB or DTW distance score matrix.

(b) Calculate the distance from each test data point to each of the p ghost points using Eq. 3.1; we now have an $n \times (m + p)$ distance matrix.

(c) Convert both the original and augmented OSB or DTW score matrix to an affinity matrix as in step 1f above.

(d) Use these affinity matrices as the *user-defined* or *precomputed* kernel for the SVM as in step 1g above.

(e) Run SVM to test.

In our experiments, when we convert a distance matrix to an affinity matrix, there are two critical parameters to set, a and K, that modify the σ for the Gaussian Kernel function. As stated in [62], the scaling parameter σ is some measure of when two points are considered similar. It is common for σ to be chosen manually, but sometimes a single value of σ does not work well for an entire data set. Therefore, we use the method in [5] to calculate the local scaling parameter σ_{ij} for each pair of data points x_i and x_j. The affinity between a pair of points can be written as:

$$k(x_i, x_j) = exp(\frac{-d(x_i, x_j)^2}{\sigma_{ij}}) \tag{4.2}$$

where $\sigma_{ij} = a \cdot mean\{K\text{nn } d(x_i), K\text{nn } d(x_j)\}$, $mean\{K\text{nn } d(x_i), K\text{nn } d(x_j)\}$ is the the mean distance of the K_d nearest neighbors of points x_i, x_j, and a is a scaling parameter. For SVM, there is the additional cost parameter C. For all experiments we used $a = 0.5$, $K = 5$, and $C = 0.5$. The final parameter to set is the number of ghost points to add per minority example, as the final results can be sensitive to the number of ghost points added. If the data set is highly imbalanced, a good heuristic is to balance the the classes, but this does not always give the best results. We then run SVM on the four matrices (after converting them to kernels): OSB score matrix without ghost points; OSB score matrix with ghost points; DTW score matrix without ghost points; and DTW score matrix with ghost points.

4.5 Experimental Results on UCR Time Series

Of the twenty UCR time series data sets, there are 17 that contain what we define as a minority class, i.e. a data set that has at least one class that is at most 35% of the size of the total data set. These data sets are listed in Table 4.1. The number of classes for these data sets range from two to fifty. Three of the data sets, *Wafer*, *Lightning-2*, and *ECG*, each have only two classes, one of which meets our definition of a minority class. For each

of the other fourteen data sets, any class whose size is at most 35% of the entire data set is taken as a minority class, the examples of all other classes of the data set are combined into a single class, and then we run the algorithm described in Section 4.4. If a data set has more than one of these simulated minority classes, we average the results over all the minority classes for that data set. The only exception to this method is with the *50 Words* data set. It has one class with only one training example, and since we need at least two examples to be able to insert a ghost point between them, this one class is excluded as a minority class. Tables 4.1 and 4.2 list the characteristics for each data set, including the number of original classes, the number of classes that are used as a minority class as they meet the requirements we defined, and the minimum and maximum number of minority class examples in the training data for each data set.

We compare the results of SVM on OSB with and without ghost points on the seventeen data sets in Table 4.1 and the results of SVM on DTW with and without ghost points on the seventeen data sets in Table 4.2. Because we are interested in the performance on minority classes, specifically minimizing the number of false negatives (see Section 3.4.1), we measure the overall accuracy (Eq. 3.7), the F_1-measure (Eq. 3.8 with $\beta = 1$) which weights precision and recall equally, and the F_2-measure (Eq. 3.8 with $\beta = 2$) which weights recall twice as heavily as precision.

As the results show in Table 4.1, for the OSB score matrix adding ghost points improves the overall accuracy on fifteen of the seventeen data sets. On the two data sets (*Beef* and *TwoPatterns*) where the overall accuracy decreases after adding ghost points, the drop is by less than 1 percentage point for both. Of the data sets where overall accuracy increases, seven have increases of over three percentage points. For example, adding ghost points increases *Lightening-2*'s accuracy by 11.5 percentage points, and *ECG*'s accuracy increases by 6.0 percentage points. For sixteen of the seventeen data sets, the F_1-measure increases with the addition of ghost points and for twelve of these data sets, the increase is more than 10 percentage points. For three data sets (*Lightening-2*, *Lightening-7*, and *Adiac*, ghost points increase the F_1-measure by over 20 percentage points (increases of 21.7, 27.4, and 20.5 percentage points, respectively), and *Beef*'s F_1-measure increases by 33.7 percentage points. Fifteen of the seventeen data sets' F_2-measure increases, with again twelve of them having increases of more than 10 percentage points. Two data sets (*Wafer* and *Adiac*) have increases over 20 percentage points, and three data sets (*Lightning-2*, *Lightning-7*, and *Beef*) have increases over 30 percentage points. The only data set where adding ghost points decreases all three measures is *TwoPatterns*, but that is because we do not train any of the three parameters discussed in Section 4.4. With training, the results can be improved so that adding ghost points performs better than not adding ghost

points.

When using the DTW score matrix (Table 4.2), adding ghost points increases the overall classification accuracy for fifteen of the seventeen data sets, decreases it for one data set, and leaves one unchanged. Four of the accuracy increases are over 4 percentage points. One data set, *Face Four*, has an increase in overall classification accuracy of 8.8 percentage points when ghost points are added. For sixteen of the seventeen data sets, adding ghost points increases the F_1-measure and the F_2-measure. These measures decrease for only the *ECG* data set. Ten data sets' F_1-measure increases by more than 10 percentage points with ghost points. Four of them have increases over 20 percentage points and two have increases over 30 (*FaceFour* and *OliveOil*. Of the sixteen data sets with F_2-measure increases, eleven have increases over 10 percentage points, four with over 20, and two with over 30 (again *FaceFour* and *OliveOil*.

These results for both OSB and DTW distance measures demonstrate that adding ghost points to minority classes may not only improve the overall classification accuracy, but may significantly increase the accuracy of predicting the rare event and at the same time reduce the number of false negatives.

Data Set	Num GP Added per Minority Example	Overall Accuracy		F_1-Measure Minority Class		F_2-Measure Minority Class	
		SVM	SVM-GP	SVM	SVM-GP	SVM	SVM-GP
SyntheticControl	2	98.83%	**99.78%** ✓	0.9668	**0.9933** ✓	0.9843	**0.9913** ✓
CBF	1	96.89%	**98.67%** ✓	0.9498	**0.9794** ✓	0.9283	**0.9687** ✓
FaceAll	1	98.83%	**99.10%** ✓	0.9063	**0.9332** ✓	**0.9307** ✓	0.9255
OSULeaf	2	86.16%	**87.33%** ✓	0.3689	**0.5604** ✓*	0.3294	**0.5267** ✓*
SwedishLeaf	3	98.27%	**98.51%** ✓	0.8552	**0.8826** ✓	0.8145	**0.8496** ✓
50Words	1	98.78%	**98.87%** ✓	0.3236	**0.4674** ✓*	0.2785	**0.4124** ✓*
Trace	2	91.50%	**96.50%** ✓*	0.7921	**0.9266** ✓*	0.7484	**0.9250** ✓*
TwoPatterns	1	**99.78%** ✓	98.88%	**0.9954** ✓	0.9771	**0.9927** ✓	0.9640
Wafer	9	96.25%	**99.53%** ✓*	0.7913	**0.9781** ✓*	0.7060	**0.9750** ✓*
FaceFour	1	91.19%	**96.88%** ✓*	0.7902	**0.9394** ✓*	0.7356	**0.9235** ✓*
Lightning2	1	73.77%	**85.25%** ✓*	0.6190	**0.8364** ✓*	0.5159	**0.8273** ✓*
Lightning7	1	89.63%	**93.74%** ✓*	0.4522	**0.7259** ✓*	0.3970	**0.7026** ✓*
ECG	1	87.00%	**93.00%** ✓*	0.7869	**0.8955** ✓*	0.7101	**0.8571** ✓*
Adiac	3	98.07%	**98.30%** ✓	0.4416	**0.6465** ✓*	0.3765	**0.6092** ✓*
Fish	3	94.86%	**96.16%** ✓	0.7550	**0.8572** ✓*	0.6857	**0.8370** ✓*
Beef	2	**82.67%** ✓	82.00%	0.1667	**0.5034** ✓*	0.1667	**0.4760** ✓*
OliveOil	1	91.11%	**94.44%** ✓*	0.5708	**0.7454** ✓*	0.5429	**0.7019** ✓*

Table 4.1: The results of adding ghost points using LCDP to the OSB distance scores on the imbalanced UCR time series data sets. Bolded, checked results indicate best scores. An asterisk for accuracy indicates at least 3 percentage points difference; for F_β-measure it indicates at least 10 percentage points difference.

Data Set	Num GP Added per Minority Example	Overall Accuracy		F_1-Measure Minority Class		F_2-Measure Minority Class	
		SVM	**SVM-GP**	**SVM**	**SVM-GP**	**SVM**	**SVM-GP**
SyntheticControl	2	97.44%	**99.28%** ✓	0.9292	**0.9788** ✓	0.9683	**0.9814** ✓
CBF	1	95.83%	**97.78%** ✓	0.9337	**0.9652** ✓	0.9171	**0.9478** ✓
FaceAll	1	96.08%	**97.10%** ✓	0.7306	**0.8064** ✓	0.7916	**0.8119** ✓
OSULeaf	2	85.12%	**86.29%** ✓	0.3450	**0.4968** ✓*	0.3085	**0.4581** ✓*
SwedishLeaf	3	97.94%	**98.63%** ✓	0.8294	**0.8964** ✓	0.7913	**0.8885** ✓
50Words	1	98.76%	**98.97%** ✓	0.3109	**0.4968** ✓*	0.2717	**0.4502** ✓*
Trace	2	90.25%	**95.00%** ✓*	0.7690	**0.8977** ✓*	0.7171	**0.8790** ✓*
TwoPatterns	1	98.45%	**99.03%** ✓	0.9680	**0.9802** ✓	0.9528	**0.9690** ✓
Wafer	9	96.82%	**99.71%** ✓	0.8299	**0.9866** ✓*	0.7595	**0.9910** ✓*
FaceFour	1	83.52%	**92.33%** ✓*	0.5152	**0.8386** ✓*	0.4638	**0.7991** ✓*
Lightning2	1	77.05%	**81.97%** ✓*	0.6818	**0.7755** ✓	0.5859	**0.7143** ✓*
Lightning7	1	90.02%	**91.78%** ✓	0.4411	**0.6439** ✓*	0.4213	**0.6337** ✓*
ECG	1	**82.00%** ✓*	79.00%	**0.7097** ✓	0.6667	**0.6471** ✓	0.6140
Adiac	3	97.74%	**98.09%** ✓	0.4188	**0.5549** ✓*	0.3887	**0.5197** ✓*
Fish	3	93.55%	**95.51%** ✓	0.7077	**0.8298** ✓*	0.6502	**0.7939** ✓*
Beef	2	**82.00%** ✓	**82.00%** ✓	0.1667	**0.3833** ✓*	0.1667	**0.3487** ✓*
OliveOil	1	85.56%	**90.00%** ✓*	0.4000	**0.7044** ✓*	0.3714	**0.6937** ✓*

Table 4.2: The results of adding ghost points using LCDP to the DTW distance scores on the imbalanced UCR time series data sets. Bolded, checked results indicate best scores. An asterisk for accuracy indicates at least 3 percentage points difference; for F_β-measure it indicates at least 10 percentage points difference.

CHAPTER 5

Conclusions

The body of work presented here addresses several significant problems in time series classification and imbalanced data sets.

Firstly, the proposed sequence matching method, OSB, directly optimizes the sum of distances of corresponding elements, allows penalized skipping of outlier elements, and defines a bijection on the remaining subsequences. A key feature of OSB is the fact that penalty for skipping outliers is part of the edge weights of the DAG built from two matched sequences. This results in skipping decisions being made with a dynamic threshold whose optimization is directly included in the dynamic programming optimization.

As demonstrated in the experimental results on standard time series data sets, the ability of skipping outlier elements leads to improved retrieval performance on many of the test data sets. However, for some data sets with-

out significant outliers it may lead to slightly reduced retrieval performance. When dealing with partial shape similarity in the presence of noise, the ability of skipping outlier elements is essential.

Secondly, we introduce an innovative method for over-sampling the minority class of imbalanced data sets. Unlike other feature based methods, our synthetic points, which we call ghost points, are added in distance space. In addition, ghost points can be added to distance spaces that are not metric, such as those induced by elastic sequence matching algorithms like Dynamic Time Warping and Optimal Subsequence Bijection. The experimental results on standard time series data sets from varied domains show that adding ghost points to the minority class can significantly improve the overall accuracy, and especially the F_1-measure and F_2-measure.

We also introduce a way to use ghost points to visualize distance data for imbalanced data sets. When plotting the distance space, adding ghost points to the minority class may change the underlying structure of the distance space such that the previously indistinct minority class now becomes observable.

Lastly, we introduce the addition of ghost points to densify minority classes in imbalanced data sets and then use the locally constrained diffusion process to reveal the intrinsic relation between data. Instead of using the direct distance between data, our approach can capture the topology of the data so that the distance measure between objects is found through the manifold enclosing

the data. It can also be viewed as a novel supervised learning method for relevance ranking. The supervised scenario is realistic, since we know the class labels of the database objects for training data.

REFERENCES

[1] John Aach and George M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17:495–508, 2001.

[2] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Proceedings of ECML'04*, pages 39–50, 2004.

[4] Arik Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *In Proceedings of the 24th International Conference on Machine Learning*, 2007.

[5] Xiang Bai, Xingwei Yang, Longin Jan Latecki, Wenyu Liu, and Zhuowen Tu. Learning context-sensitive shape similarity by graph transduc-

tion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:861–874, 2010.

[6] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, 2004.

[7] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[9] Philip Chan and Salvatore J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168. AAAI Press, 1998.

[10] N. V. Chawla, K. W. Bowyer, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[11] Nitesh V. Chawla, Ar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: improving prediction of the minority class in boosting.

In *Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*, pages 107–119, 2003.

[12] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 792–803. VLDB Endowment, 2004.

[13] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2005. ACM.

[14] Chiu, Keogh, and Lonardi. Probabilistic discovery of time series motifs. In *Proceedings ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Washington*, 2003.

[15] Chu, Keogh, Hart, and Pazzani. Iterative deepening dynamic time warping for time series. In *Proceedings SIAM International Conference on Data Mining*, 2002.

[16] David A. Cieslak and Nitesh V. Chawla. Start globally, optimize locally, predict globally: Improving performance on imbalanced data. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on*

Data Mining, pages 143–152, Washington, DC, USA, 2008. IEEE Computer Society.

[17] Ronald R. Coifman and Stephane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006.

[18] Das, Gunopulos, and Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.

[19] Alex A. Freitas and Jon Timmis. Revisiting the foundations of artificial immune systems: a problem-oriented perspective. In *Hart (Eds.) Artificial Immune Systems (Proc. ICARIS-2003), LNCS 2787*, pages 229–241. Springer, 2003.

[20] Costis Georgiou and Hamed Hatami. CSC2414- Metric embeddings. Lecture 1: A brief introduction to metric embeddings, examples and motivation. 2008.

[21] Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. volume 3644 of *Lecture Notes in Computer Science*, pages 878–887. Springer, 2005.

[22] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong jiang Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:328–340, 2005.

[23] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, 1975.

[24] Hoeppner. Discovery of temporal patterns. learning rules about the qualitative behavior of time series. In *Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, Freiburg*, pages 192–203, 2001.

[25] Karen Hovsepian, Peter Anselmo, and Subhasish Mazumdar. Supervised inductive learning with LotkaVolterra derived models. *Knowl. Inf. Syst.*, 2010.

[26] David W. Jacobs, Daphna Weinshall, and Yoram Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:583–600, 2000.

[27] Keogh, Lonardi, and Ratanamahatana. Towards parameter-free data mining. In *Proceedings ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Seattle*, 2004.

[28] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. UCR time series classification/clustering page. Website, 2006. `http://www.cs.ucr.edu/~eamonn/time_series_data/`.

[29] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215, 1998.

[30] Longin Jan Latecki, Rolf Lakämper, and Ulrich Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 424–429, 2000.

[31] Longin Jan Latecki, Qiang Wang, Suzan Köknar-Tezel, and Vasileios Megalooikonomou. Optimal subsequence bijection. *IEEE International Conference on Data Mining*, pages 565–570, 2007.

[32] Julian Laub and Klaus-Robert Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5:801–818, 2004.

[33] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell*, 29:286–299, 2007.

[34] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[35] Pierre-François Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318, 2009.

[36] Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[37] Vasileios Megalooikonomou, Qiang Wang, Guo Li, and Christos Faloutsos. A multiresolution symbolic representation of time series. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 668–679, Washington, DC, USA, 2005. IEEE Computer Society.

[38] L. Mena and J.A. Gonzalez. Machine learning for imbalanced datasets: Application in medical diagnostic. In *In Proceedings of the 19th International FLAIRS Conference*, 2006.

[39] Michael D. Morse and Jignesh M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 569–580, New York, NY, USA, 2007. ACM.

[40] Rafiei. On similarity-based queries for time series data. In *Proceedings of the Int. Conf. on Data Engineering, Sydney*, pages 410–417, 1999.

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

[42] Hiroaki Sakoe and Seibi Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International*

Congress on Acoustics, Budapest, volume 3, pages 65–69, Budapest, 1971. Akadémiai Kiadó.

[43] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[44] Salvador, Chan, and Brodie. Learning states and rules for time series anomaly detection. In *Proceedings of the 17th Intl. Florida Artificial Intelligence Research Society Conference, Florida*, pages 306–311, 2004.

[45] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.

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

[47] Edward R. Tufte. *The Visual Display of Quantitative Information, 2nd edition*. Graphics Press, Cheshire, CT, USA, second edition, 2001.

[48] C.J van Rijsbergen. In *Information Retrieval*. Butterworths, London, 1979.

[49] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[50] V. M. Velichko and N. G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2:223–234, 1970.

[51] Vlachos, Hadjieleftheriou, Gunopulos, and Keogh. Indexing multidimensional time-series with support for multiple distance measures. In *Proceedings of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Washington*, pages 216–225, 2003.

[52] Vlachos, Kollios, and Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings of 18th ICDE, San Jose, CA*, pages 673–684, 2002.

[53] Benjamin X. Wang and Nathalie Japkowicz. Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, March 2009.

[54] Marc Weber, Marc Alexa, and Wolfgang Müller. Visualizing time-series on spirals. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 7–14, 2001.

[55] Gary M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explor. Newsl.*, 6(1):7–19, 2004.

[56] Gary M. Weiss and Haym Hirsh. Learning to predict rare events in event sequences. In *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 359–363. AAAI Press, 1998.

[57] Gary M. Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[58] K. Woods, C. Doss, K. Bowyer, J. Solka, C. Priebe, and P. Kegelmeyer. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:1417–1436, 1993.

[59] Gang Wu and Edward Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *Workshop on Learning from Imbalanced Datasets in International Conference on Machine Learning (ICML)*, 2003.

[60] Xingwei Yang, Xiang Bai, Longin Jan Latecki, and Zhuowen Tu. Improving shape retrieval by learning graph transduction. In *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 788–801. Springer, 2008.

[61] Yi, Jagadish, and Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings Int. Conf. on Data Engineering (ICDE98)*, pages 201–208, 1998.

[62] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2004.

[63] Huimin Zhao. Instance weighting versus threshold adjusting for cost-sensitive classification. *Knowl. Inf. Syst.*, 15(3):321–334, 2008.

[64] Dengyong Zhou and Bernhard Schlkopf. Learning from labeled and unlabeled data using random walks. In *Pattern Recognition, Proceedings of the 26th DAGM Symposium*, pages 237–244. Springer, 2004.

[65] Xueyuan Zhou and Chunping Li. Text classification by markov random walks with reward. In *DMIN*, pages 275–278, 2005.