Constraint-Based Learning of Distance Functions for Object Trajectories

Wei Yu[1] and Michael Gertz[2]

[1] Department of Computer Science, University of California, Davis, U.S.A.
weyu@ucdavis.edu

[2] Institute of Computer Science, University of Heidelberg, Germany
gertz@informatik.uni-heidelberg.de

Abstract. With the drastic increase of object trajectory data, the analysis and exploration of trajectories has become a major research focus with many applications. In particular, several approaches have been proposed in the context of similarity-based trajectory retrieval. While these approaches try to be comprehensive by considering the different properties of object trajectories at different degrees, the distance functions are always pre-defined and therefore do not support different views on what users consider (dis)similar trajectories in a particular domain.

In this paper, we introduce a novel approach to learning distance functions in support of similarity-based retrieval of multi-dimensional object trajectories. Our approach is more generic than existing approaches in that distance functions are determined based on constraints, which specify what object trajectory pairs the user considers similar or dissimilar. Thus, using a single approach, different distance functions can be determined for different users views. We present two learning techniques, *transformed Euclidean distance* and *transformed Dynamic Time Warping*. Both techniques determine a linear transformation of the attributes of multi-dimensional trajectories, based on the constraints specified by the user. We demonstrate the flexibility and efficiency of our approach with applications to clustering and classification on real and synthetic object trajectory datasets from different application domains.

1 Introduction

Driven by major advancements in sensor technology, GPS-enabled mobile devices, and object tracking, large amounts of data describing moving object trajectories are currently generated and managed in various application domains (see, e.g., [17,19] for some excellent surveys). In order to effectively analyze and explore such data for patterns of interest and unexpected phenomena, several data mining techniques for object trajectories have been developed. In most of the applications, trajectory data are typically multi-dimensional. In particular, trajectory data can be treated as multi-dimensional time series and thus respective data analysis techniques can be applied.

A fundamental ingredient of most of the trajectory analysis tasks are distance measures that allow to effectively determine the similarity of trajectories. Respective tasks include trajectory clustering, classification, and k-nearest neighbor search. Several approaches and distance measures have been proposed, tailored to multi-dimensional object trajectories, e.g., [3,20,21]. Common to all the existing approaches is that distances measures (or functions) are explicitly given as part of the framework. Although these measures try to be comprehensive by considering the various dimensions of trajectory data, they typically assume that the dimensions determining similarity are the original dimensions that describe the trajectories. However, there are several application domains where different users have different views on what trajectories are similar and which ones are not. For example, if variables x and y describe 2-d trajectories, for the first user, dimension x might be important in determining similarity while for the second user, y is more important. And a third user considers a linear combination of x and y as a critical factor to describe trajectory similarity. If x is a basic component of the distance function, the similarity measure is of little help to the second and third user. In this sense, existing approaches relying on unsupervised, "hard-coded" distance measures do not provide much flexibility in terms of supporting different user views and domain specific knowledge about trajectories.

In this paper, we address this problem by introducing a novel approach in which, given a set of multi-dimensional object trajectories, distance measures are *learned from constraints*. These constraints are specified by the user and simply consist of a few pairs of trajectories the user considers similar or dissimilar. We propose learning techniques of two distance measures, *transformed Euclidean Distance* and *transformed Dynamic Time Warping*, which are based on posing an optimization problem. Both techniques detect critical attributes that determine similarity between the multi-dimensional trajectories based on the constraints specified by the user. A resulting linear transformation matrix is then simply embedded in a distance function that satisfies the user constraints best. In our approach we do not rely on the traditional techniques that utilize a comprehensive set of labeled training data, as they typically appear in (supervised) classification problems. The types of constraints considered in this paper are much simpler. Consequently, our approach cannot only be applied to traditional classification problems but also to clustering and similarity search for object trajectories. In order to also accommodate similarity views that consider object movement patters at different rates, an important consideration is that the distance measure has the ability to allow time warping along the time axis. We achieve this through our novel transformed Dynamic Time Warping technique, which is different from techniques that simply learn a Mahalanobis distance for multi-dimensional data points in a supervised fashion (e.g., [23]).

Thus, the main contribution of our work is to provide a single framework that allows to derive distance measures that satisfy different user needs and (subjective) views on some given trajectory data. This approach enables the efficient computation of distance measures applicable to various mining tasks for object trajectories, as we will demonstrate in our comprehensive experiments where we use different datasets and consider typical data mining tasks such as trajectory clustering and classification.

The rest of the paper is organized as follows. After a discussion of related work in the following section, in Sec. 3, we outline the concepts underlying trajectories and distance measures. In Sec. 4, we discuss in detail our approach to learning distance functions in the context of user constraints. After a demonstration of the effectiveness of the proposed approach using different datasets in Sec. 5, we conclude the paper in Sec. 6.

2 Related Work

Our approach is mostly related to the areas of similarity search for moving object trajectories and distance learning.

Recently, there has been very active research on mining trajectories, which typically can be viewed as (multi-dimensional) time series, and trajectory similarity search. For this, several alternative distance measures have been proposed. In [12], Lee et al. use the Euclidean Distance (ED) for multi-dimensional time series. Euclidean Distance, however, is known to be sensitive to local distortions in the time axis. To address this problem, Vlachos et al. extend Dynamic Time Warping (DTW) and Longest Common Subsequence (LCSS) for multi-dimensional trajectories [20]. Chen et al. in [3] study Edit Distance on Real sequence (EDR), which is based on string edit distance. Chen et al. in [2] propose ERP, which tries to combine the advantages of DTW and EDR. In [24], Wu et al. study a One-way distance (OWD) function to compute the spatial similarity of trajectories. Lee et al. also developed approaches for trajectory clustering [11] based on sub-trajectories. Vlachos et al. present a DTW-based distance measure that is invariant to rotation in [21].

All these approaches are designed for multi-dimensional trajectories. However, all these proposed distance measures assume that the dimensions used to describe trajectory similarity are the original dimensions that describe the trajectories. Our work is different from these approaches, because in our approach, distance measures are learned from user-specified constraints and are thus able to satisfy user views on trajectory similarity. Users are allowed to give examples to indicate which trajectories they consider similar and/or dissimilar. Our algorithm then detects meaningful underlying dimensions that allow to explain the similarity and/or dissimilarity, and yields a distance measure that satisfies the user constraints best.

Our work is also different from traditional unsupervised feature extraction techniques such as principal component analysis (PCA), factor analysis (FA), and Isomap. Also these techniques do not guarantee to capture the trajectory properties that are of interest to the user. They are learned only using the intrinsic properties of the trajectory data. For example, PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance.

There also has been considerable work on supervised distance learning, such as learning a distance function for classification problems (e.g., [5,25]). However, a problem with these methods is that they need a labeled training dataset for which then the classification accuracy can be optimized. We do not assume such a labeled training dataset but only a few simple user constraints that indicate which trajectories are (dis)similar. Xing et al. in [23] assume similar user constraints and propose a global Mahalanobis distance for data points that respect such constraints. However, we believe that their method does not solve our problem at hand, for the following reasons. First, the objective of our approach is to learn a good distance function for multi-dimensional object trajectories. Their approach, including many other distance learning techniques (see [25]), try to learn a distance for multi-dimensional data points, where only a snapshot of attribute values is meaningful. Second, many trajectory mining tasks aim to discover objects that have similar movement patterns, even at different rates. Hence, the distance measure is expected to have the ability to allow time warping in the time axis. This important consideration, however, is not present in learning a distance measure for multi-dimensional data points.

3 Background: Trajectories and Distances

In this paper, we consider object trajectories where n measurements are recorded at discrete points in time. For a trajectory of the pattern $P = [p_1, p_2, \ldots, p_m]$, each component p_i is an n-dimensional vector of measurements (attributes) $(p_{1,i}, p_{2,i}, \ldots, p_{n,i}) \in \mathbb{R}^n$ recorded at time i. We refer to the number of time instants in P as *size* of the trajectory.

For one-dimensional time series (trajectory data), the Euclidean distance (ED) and Dynamic Time Warping (DTW) distance are commonly used as similarity measures (e.g., [1,7,15,16]). Both distance measures can easily be extended to multi-dimensional trajectories as suggested, for example, by Vlachos et al. [20]. Because Euclidean distance is only defined for trajectories of the same size, an interpolation needs to be applied to the input trajectories. For this, different techniques can be used (see, e.g., [9,15]).

Definition 1. *Given two n-dimensional trajectories P and Q of size k (after interpolation). The Euclidean Distance between P and Q, denoted $ED(P, Q)$, is defined as*

$$ED(P, Q) := \sqrt{\sum_{i=1}^{k} (p_i - q_i)^T (p_i - q_i)}$$

For the DTW distance, input trajectories do not necessarily have to have the same size. In the following, for a trajectory $P = [p_1, p_2, \ldots, p_m]$, we denote with $P_{1\ldots i}$ the sub-trajectory of P containing the i first elements of P, i.e., $P_{1\ldots i} = [p_1, p_2, \ldots, p_i]$.

Definition 2. *Given two n-dimensional trajectories P and Q of size m and l, respectively. The DTW distance between P and Q, denoted as $DTW(P, Q)$, is determined by evaluating the recurrence equation*

$$DTW(P_{1\ldots i}, Q_{1\ldots j}) := \sqrt{(p_i - q_j)^T (p_i - q_j) + \min \begin{cases} DTW^2(P_{1,\ldots,i-1}, Q_{1,\ldots,j-1}) \\ DTW^2(P_{1,\ldots,i-1}, Q_{1,\ldots,j}) \\ DTW^2(P_{1\ldots,i}, Q_{1,\ldots,j-1}) \end{cases}}$$

Note that some papers (e.g., [20,22]) omit the square root function in the definition of DTW. It is only used for some optimizations and does not change the essence of the function [22]. DTW finds the optimal alignment between two time series so that their distance is minimized. An alignment can be obtained by using dynamic programming to solve the recurrence equation. In order to speed up the computation and to prevent pathological warping, band constraints such as the Sakoe-Chiba band [18] can be applied. For more details on DTW, we refer the reader to [1,15].

4 Learning Distance Functions

In this section, we introduce two flexible distance functions for multi-dimensional object trajectories, called *transformed Euclidean distance* and *transformed DTW*. We present the learning algorithms underlying these two distance functions in Sections 4.1 and 4.2, respectively.

Throughout the paper, we use the notion of *(user) constraints*. For this, assume a given set of trajectories $S = \{s_i\}_{i=1}^{u}$, $s_i \in R^{n \times m}$. Constraints are specified by the user in the form of two sets: the ML (*Must-Link*) set and the CL (*Cannot-Link*) set. If the user specifies a pair (s_i, s_j) to be in ML, then he (subjectively) considers the trajectories s_i and s_j to be similar. Analogously, he specifies a pair (s_i, s_j) to be in CL, if he considers s_i and s_j to be dissimilar.

Given ML and CL constraints, the objective is now to learn a distance measure for a given set of trajectories such that the pairs of trajectories in ML end up to be similar to each other (based on the computed distance) and the pairs in CL are dissimilar to each other. For this, we first propose the following distance functions, transformed Euclidean distance and transformed DTW, to be learned. After this, we provide the intuition behind these two functions.

Definition 3. *Given two n-dimensional trajectories P and Q of length k after interpolation and a real symmetric positive semi-definite matrix $A \in R^{n \times n}$. The* Transformed Euclidean distance *between P and Q, denoted $ED_A(P, Q)$, is defined as*

$$ED_A(P, Q) := \sqrt{\sum_{i=1}^{k} (p_i - q_i)^T A (p_i - q_i)}.$$

Definition 4. *Given two n-dimensional trajectories P and Q of length m and l, respectively, and a real symmetric positive semi-definite matrix $A \in R^{n \times n}$. The* Transformed Dynamic Time Warping distance *between P and Q, denoted $DTW_A(P, Q)$, is obtained by evaluating the following recurrence equation*

$$DTW_A(P_{1\ldots i}, Q_{1\ldots j}) := \sqrt{(p_i - q_j)^T A (p_i - q_j) + \min \begin{cases} DTW_A^2(P_{1,\ldots,i-1}, Q_{1,\ldots,j-1}) \\ DTW_A^2(P_{1,\ldots,i-1}, Q_{1,\ldots,j}) \\ DTW_A^2(P_{1\ldots i}, Q_{1,\ldots,j-1})) \end{cases}}$$

In order to give the intuition behind the two distance measures and explain what A actually is, we introduce the concept of *transformed trajectories.*

Definition 5. *Given a trajectory* $P = [p_1, p_2, \ldots, p_m], p_i \in R^n$ *and a linear transformation matrix* $W \in R^{n \times n}$*. The* transformed trajectory*, denoted* P_W*, is defined as* $P_W := [Wp_1, Wp_2, \ldots, Wp_m]$*.*

The transformation of P using matrix W thus can be considered as applying a linear transformation matrix W to each point $p_i (1 \leq i \leq m)$ in P, replacing p_i with Wp_i. If and only if A is real symmetric, positive semi-definite, one can find a real matrix $W = A^{1/2}$, i.e. $A = W^T W$. Then one can also find a correspondence between ED$_A$, DTW$_A$, and the transformed trajectories, as discussed below. Given above definitions, it is straightforward to see that

(1) $\text{ED}_A(P, Q) = \text{ED}(P_{A^{1/2}}, Q_{A^{1/2}})$ and
(2) $\text{DTW}_A(P, Q) = \text{DTW}(P_{A^{1/2}}, Q_{A^{1/2}})$

Hence, learning $\text{ED}_A(P, Q)$ or $\text{DTW}_A(P, Q)$ is equivalent to finding a transformation matrix $W(W = A^{1/2})$ for all trajectories such that all pairs of trajectories in ML end up to be similar to each other, and the pairs of trajectories in CL are dissimilar to each other.

4.1 Learning a Transformed Euclidean Distance from Constraints

In the following, we present the algorithm for learning the transformed Euclidean distance ED$_A$ for a set of trajectories from some user-specified ML and CL constraints. We pose the distance learning approach as an optimization problem in a way similar to the approach proposed by Xing el al. in [23]. Their approach focuses on learning a distance metric for multi-dimensional data points, while our method tries to learn distance functions for multi-dimensional object trajectories. Intuitively, the desired transformed Euclidean distance measure should bring each pair of trajectories (s_i, s_j) in ML as close as possible, and each pair of trajectories (s_i, s_j) in CL as far apart as possible. Therefore, for the optimization, the sum of squared distances between all pairs (s_i, s_j) in ML, denoted $f(A)$, is to be minimized, with the constraint that the sum of distances between all pairs (s_i, s_j) in CL, denoted $g(A)$, is greater than the constant 1 to ensure that $A \neq 0$. In the constraint formulation (2) below, we do not use $ED_A^2(s_i, s_j)$ because this then always leads to the matrix A having rank 1. As indicated earlier, another constraint is that the matrix A is positive semi-definite, denoted $A \geq 0$. The optimization problem then is stated as follows:

$$\min_A \; f(A) = \sum_{(s_i, s_j) \in ML} ED_A^2(s_i, s_j) \tag{1}$$

$$\text{s.t.} \;\; g(A) = \sum_{(s_i,s_j)\in CL} ED_A(s_i,s_j) \geq 1 \text{ and } A \geq 0 \tag{2}$$

Note that the constant 1 on the right hand side of the first constraint in (2) can be replaced by any other positive constant c because this only leads to A being replaced by c^2A. According to the definition of ED$_A$, the objective function $f(A)$ and both constraints are convex functions. Therefore, one can efficiently find the global optimum. If A is a diagonal matrix, then the transformation W (i.e., $A^{1/2}$) is a scaling matrix. Each diagonal entry $W(i,i)$ can be considered as the "weight" assigned to the dimension $d_i, 1 \leq i \leq n$. Finding the optimum is then equivalent to minimizing

$$\sum_{(s_i,s_j)\in ML} ED_A^2(s_i,s_j) - \log\left(\sum_{(s_i,s_j)\in CL} ED_A(s_i,s_j)\right)$$

To solve this optimization problem, we choose the well-known Newton-Raphson method as in [23]. If A is a full $n \times n$ square matrix, this method takes $O(n^6)$ to invert the Hessian matrix. For efficiency, we solve the problem using gradient ascent and iterative projections in $O(n^2)$, as proposed by Xing et al. in [23].

4.2 Learning Transformed Dynamic Time Warping from Constraints

We now discuss the learning algorithm for DTW$_A$, which can address the problem of distortions in the time axis. First, we pose the optimization problem. Then, we discuss the case where A is diagonal and finally present the solution of a full matrix A.

Optimization Problem. Based on an idea similar to deriving the optimization problem in Sec. 4.1, we now simply replace the distance function ED$_A$ with DTW$_A$ and obtain our new optimization problem, denoted OP1, as follows:

$$\min_A \;\; f(A) = \sum_{(s_i,s_j)\in ML} DTW_A^2(s_i,s_j) \tag{3}$$

$$\text{s.t.} \;\; g(A) = \sum_{(s_i,s_j)\in CL} DTW_A(s_i,s_j) \geq 1 \text{ and } A \geq 0 \tag{4}$$

However, according to the definition of DTW$_A$, the objective function $f(A)$ and the constraints are not convex, and it thus cannot be solved using the methods presented earlier. We now formulate the problem in a different way, denoted OP2, as follows:

$$\min_A \;\; h(A) = \frac{f(A)}{g^2(A)} = \frac{\sum_{(s_i,s_j)\in ML} DTW_A^2(s_i,s_j)}{(\sum_{(s_i,s_j)\in CL} DTW_A(s_i,s_j))^2} \tag{5}$$

$$\text{s.t.} \;\; trace(A) = 1 \text{ and } A \geq 0 \tag{6}$$

OP2 is in fact equivalent to OP1. Due to space constraints, we only give proof sketch here: Suppose A_{OP1} is a solution of OP1, we can prove that A_{OP1} multiplied by a certain constant is a solution of OP2, and vise versa. According to the definition of DTW_A, a constant factor to A only results in all pairwise DTW_A distances between trajectories multiplied by the square root of this constant. Thus it does not change the clustering, classification, or similarity search result.

In OP2, the objective function $h(A)$ is derived from $f(A)$ and $g(A)$. $h(A)$ is to be minimized under the constraints that the trace of A is 1 and A is positive semi-definite ($A \geq 0$). Our motivation for this specific choice is that it significantly simplifies the heuristic search. The solution A is required to satisfy all the constraints and minimize the objective function. If we treat the set of matrices satisfying the constraints as the search space, then the task is to find a matrix within this search space that minimizes the objective function. In the problem formulation OP2, the search space is the set of all positive semi-definite matrices with trace 1, which is equivalent to the set of symmetric matrices whose eigenvalues add up to 1 and each eigenvalue is non-negative. As we will show in the next sections, this search space is constructed very easily.

The case of a diagonal matrix *A*. In the following, we consider the case of learning a diagonal matrix A, which corresponds to a scaling matrix W whose diagonal entries $W(i, i)$ are considered the "weight" assigned to dimension i.

In this approach, we use a hill-climbing search algorithm to find a good solution for A. The components of the search algorithm are the initial state, heuristic function, successor function, and goal test. We first give the intuition behind this approach before explaining how we realize these four components.

We denote the diagonal matrix A from learning ED_A in Sec. 4.1 as A_0, that is, A_0 is the global optimum when ED_A is used. Now we start with A_0 and increase or decrease its diagonal entries to find a matrix A that works better than A_0 when DTW_A is used. In order to mitigate the problem of a local optimum in hill-climbing search, we also conduct the search by multiple restarts with a random diagonal matrix A. We then compare the solutions for A from both approaches and finally choose the one with the minimum objective function.

(1) Initial State: We perform experiments on two different initial states.

– Starting with A_0, the diagonal matrix from learning the transformed Euclidean distance ED_A.
– Starting with A_{rand}, a diagonal matrix with random diagonal entries between 0 and 1.

Note that the sum of diagonal entries of A_0 and A_{rand} is normalized to 1, required by constraint (6). This can simply be done by dividing each entry by the sum of all diagonal entries.

(2) Heuristic Function: The heuristic function h is used to evaluate the quality of an operation. We use as heuristic function the objective function (5) formulated in the optimization problem OP2, i.e.,

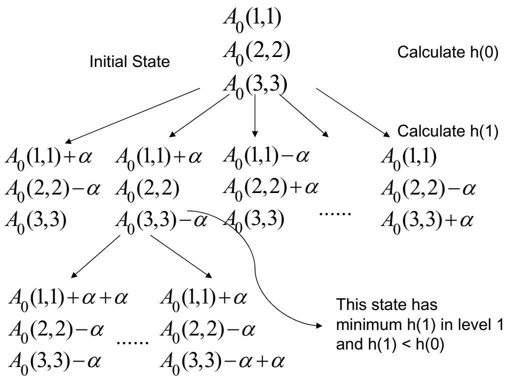

Initial State
$A_0(1,1)$ $A_0(2,2)$ $A_0(3,3)$
Calculate h(0)
Calculate h(1)
$A_0(1,1)+\alpha$ $A_0(2,2)-\alpha$ $A_0(3,3)$
$A_0(1,1)+\alpha$ $A_0(2,2)$ $A_0(3,3)-\alpha$
$A_0(1,1)-\alpha$ $A_0(2,2)+\alpha$ $A_0(3,3)$
......
$A_0(1,1)$ $A_0(2,2)-\alpha$ $A_0(3,3)+\alpha$
$A_0(1,1)+\alpha+\alpha$ $A_0(2,2)-\alpha$ $A_0(3,3)-\alpha$
......
$A_0(1,1)+\alpha$ $A_0(2,2)-\alpha$ $A_0(3,3)-\alpha+\alpha$
This state has minimum h(1) in level 1 and h(1) < h(0)

Fig. 1. Illustration of hill-climbing algorithm, $n = 3$. α is the step-size parameter

$$h = \frac{\sum\limits_{(s_i,s_j)\in ML} DTW_A^2(s_i,s_j)}{(\sum\limits_{(s_i,s_j)\in CL} DTW_A(s_i,s_j))^2} \tag{7}$$

(3) Successor function: Once the initial state has been determined, we want to improve that solution. A successor function generates a successor state of the current state. Figure 1 illustrates this progress. We define a step-size parameter α, $0 \le \alpha \le 1$. Assume there are n dimensions. In the current state, the weights of the corresponding dimensions are $A(1,1), A(2,2), \ldots, A(n,n)$. To generate the successor states, we increment the weight $A(i,i), 1 \le i \le n$ by α and decrement another weight $A(j,j), j \neq i, 1 \le j \le n$ by α. In this case, the sum of the weights is always 1. We apply this step to each (i,j) pair and pick the optimal successor state to survive and reproduce. According to our definition of the heuristic function, the optimal successor state is the state whose h is minimal among the states at the same level. If it is better than the current state, let it reproduce. If not, return the current state.

(4) Terminal test: It defines the condition when to stop the search. We stop the search if no improvement can be made, and the current state is returned.

The case of a full matrix *A*. We now detail the learning approach for a full matrix A. The idea again is to use a hill-climbing algorithm. Similar to the learning of a diagonal matrix A, we start with the full matrix A from learning of ED$_A$, denoted as A_e, and try to find a matrix A that works better than A_e with DTW$_A$.

First, we find the number of parameters that determine a real symmetric $n \times n$ full matrix A. With eigendecomposition, $A = ULU^T$. Since A is real symmetric, we can always find an orthonormal matrix U with $det(U) = 1$. Hence, both U and U^T are rotation matrices in an n-dimensional space. Similarly, with eigendecomposition, we obtain $A_e = U_e L_e U_e^T$, where U_e is a rotation matrix. We can relate U to U_e as $U = BU_e$, where B is also a rotation matrix. Hence, A

can be expressed as $A = BU_eLU_e^TB^T$. $A = A_e$ when $B = I_n, L = L_e$. Since U_e is known, A is determined by only two matrices B and L. L is parameterized by n eigenvalues of A. As a rotation matrix in n-dimensional space, B is determined by $\frac{n(n-1)}{2}$ angles. This is obvious in a 2-dimensional space where a simple rotation matrix is parameterized by one angle θ as $\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$. Generally, a rotation matrix in an n-dimensional space can be considered as a composition of the rotations occurring in each plane formed by any two coordinate axes, and there are $\frac{n(n-1)}{2}$ such planes. In an n-dimensional space, the rotation matrix that rotates the axis X_i in the direction of X_j by angle θ is $R_{i,j}(\theta) = \left\{ r_{ab} \mid \begin{array}{l} r_{i,i} = cos\theta \\ r_{j,j} = cos\theta \\ r_{i,j} = -sin\theta \\ r_{j,i} = sin\theta \\ r_{a,a} = 1, a \neq i, a \neq b \\ r_{a,b} = 0, \text{elsewhere} \end{array} \right\}$.

As a rotation matrix in n-dimensional space, B can be obtained by building the product of all $R_{i,j}(\theta)$ [6]. As one can see, each matrix $R_{i,j}$ has one parameter θ. Thus B is parameterized by $\frac{n(n-1)}{2}$ angles. We therefore conclude that learning A is equivalent to assigning values to $\frac{n(n-1)}{2}$ angles and n eigenvalues.

We now determine the search space. According to constraint (6), $L_{i,i}$ (i.e., the eigenvalues of A) should satisfy $\sum_{i=1}^{n} L_{i,i} = 1$ and $L_{i,i} \geq 0, 1 \leq i \leq n$. Each angle is between 0 and 2π. Let $\theta_1, \ldots, \theta_{\frac{n(n-1)}{2}}$ denote the angles that parameterize B. The components of the search algorithm are then as follows:

(1) Initial State: Similar to the learning of a diagonal A, there are two different initial states, corresponding to A_e and a random matrix A, respectively.

(1) $\theta_i = 0$; $L = L_e$.

(2) Random θ_i and L with random entries.

(2) Heuristic Function:

$$h = \frac{\sum_{(s_i,s_j)\in ML} DTW_A^2(s_i,s_j)}{(\sum_{(s_i,s_j)\in CL} DTW_A(s_i,s_j))^2} \tag{8}$$

(3) Successor Function: The successor function generates successor states by changing the values of the parameters in the current state. Suppose α_θ, α_L are the step-size parameters for θ_i and L_i, respectively. In the current state, suppose the angles are $\theta_1, \ldots, \theta_{\frac{n(n-1)}{2}}$ and the eigenvalues are $L_1, \ldots, L_n$. There are two available successor functions:

(a) Increasing θ_i by α_θ if $\theta_i + \alpha_\theta \leq 2\pi$.
(b) Increasing L_i by α_L and decrementing another eigenvalue L_j by α_L, to keep the sum of L_i as 1.

(a) and (b) alternate in the process. The algorithm evaluates the successor states according to the heuristic function presented above and chooses the best state to continue the search.

(4) Goal Test: The search terminates if no successor state is better than the current state.

5 Experiments and Evaluation

In this section, we present the experiments and evaluations we conducted to demonstrate the flexibility and effectiveness of our distance learning approach. In Sec. 5.1, we discuss the adaptivity of our approach to user-specified constraints. In Sec. 5.2, we then show how our learning approach is used to improve clustering and classification performance for object trajectories. In our experiments, we use synthetic and real-world datasets that describe object trajectories.

For this, we compare the performance of the following six distance measures:

- (1) Euclidean distance ED and (2) Dynamic Time Warping DTW,
- Transformed Euclidean distance ED_A with (3) diagonal and (4) full matrix A, and
- Transformed Dynamic Time Warping DTW_A with (5) diagonal and (6) full matrix A.

We chose the measures ED and DTW, because Ding et al. have shown that EDR, LCSS, ERP etc. are not more accurate than the classic DTW in general [4]. Thus, if our learned distance measures beat ED and DTW, this suggests that they also beat the other distance measures. Therefore, we do not discuss measures other than ED and DTW in our experiments. Furthermore, we constrain the warping band of DTW and DTW_A to up to 20% of the trajectory size as in [20]. We use $\alpha = \alpha_L = 0.1$, and $\alpha_\theta = \pi/8$ in this work.

5.1 Adaptivity to Constraints

In the following, we show that our learning approach produces distance measures that are adaptive to the constraints specified by users. Assume two users with different views on similarity of a given trajectory dataset and each user specifies a set of constraints reflecting his view. By first learning the distance measures from the constraints and then performing clustering using the learned distance measures, we show that our "single" learning approach can help both users find the results satisfying their views on trajectory similarity.

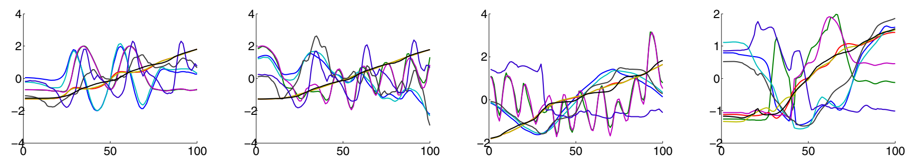
We use a real world trajectory dataset to verify the adaptivity of our approach to constraints. In addition to the clustering accuracy of the six different distance measures on this dataset, we also present the visual comparison between original and transformed trajectories to illustrate the flexibility of our approach.

For this experiment, we generated a trajectory dataset, called myMotion [26], that represents human motion. The myMotion dataset consists of human gait data captured by a VICON system. All the data were created using a female actor who had markers attached. The female actor performed 4 groups of movements, each group consisting of 9 examples. The camera recording rate was 60 frames per second. For each marker, its 3-d positions x, y and z in each frame were recorded. The 4 groups of motion data represent the following movements:

G1: Walking in a straight line and swinging one arm.
G2: Walking in a straight line and keeping arms stable.
G3: Walking in a Z-shape line and keeping arms stable.
G4: Walking in a Z-shape line and swinging one arm.

We use the data readings from 3 markers attached to the actor's body, one arm, and one leg. Each marker has 3 features, corresponding to the x, y, and z positions. Thus, each example has 9 features. The myMotion dataset contains a total of 36 9-d time series data instances. All instances have the same length of 100, obtained through interpolation. Each dimension is Z-normalized, as a standard preprocessing step in motion matching.

Figure 2 shows four samples of the 9-d trajectories. Each of the four plots represents one sample from G1 to G4. In the figures, each dimension is plotted as a time series in a unique color. The x axis represents the time t, and the y axis corresponds to the coordinates.



(a) Sample from $G1$. (b) Sample from $G2$. (c) Sample from $G3$. (d) Sample from $G4$.

Fig. 2. Four **original** samples from the myMotion dataset

Suppose there are two users with different views of similarity and they both want to cluster the 36 trajectories into 2 clusters. For the first user, the "true clusters" are distinguished by the actor's arm movements. Therefore, he thinks that the data from $G1$ and $G4$ belong to one cluster and G2 and G3 belong to the other. On the other hand, the "true clusters" for the second user are distinguished by the actor's walking routine, i.e., Z-shape or straight line. He considers the trajectories from G1 and G2 being one cluster and the trajectories from G3 and G4 being the other cluster. Both users now specify their preferences in the form of ML and CL constraints, indicating their respective similarity views. In this experiment, we "simulate" the specification of user constraints by random sampling. That is, trajectories in a user's cluster are assumed to have the same label. Constraints ML are random samples from all pairs of trajectories with the same label from the user's point of view, and constraints CL include random samples of all pairs of trajectories having different labels.

For each user and his constraints CL and ML, the learning approach generates the distance measures from his constraints. Then the trajectories are clustered according to the learned distance measures. In the clustering experiment, we use the group average hierarchical clustering algorithm to cluster the data. In the corresponding dendrogram, we look at the first (top) branch, which produces two subtrees, each representing one cluster. The cluster accuracy is given as

$$\text{Accuracy} = \frac{\text{number of correctly labeled data}}{\text{number of all data}}.$$

We compute the clustering accuracy based on ED, DTW and the four distance measures ED_A and DTW_A for a diagonal and full matrix A, respectively. Here, 4% of all pairs of trajectories having the same label (resp. different labels) are randomly sampled as ML (resp. CL). We will talk more about the relation between the size of constraints and the clustering accuracy in Sec. 5.2. Table 1 lists the accuracy values for both users for the learned matrix A. As one can see, although the two users give two different sets of constraints, our approach achieves a very high clustering accuracy in both cases.

Table 1. Clustering accuracy for the myMotion dataset given two different sets of user constraints. The numbers in bold show the best clustering accuracy for each user. DTW_A performs best for both users.

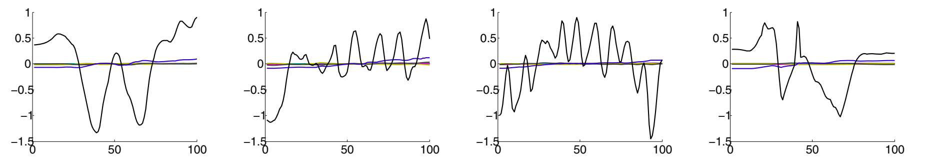
	ED_A, diag. A	ED_A, full A	DTW_A, diag. A	DTW_A, full A	DTW	ED
User 1	83.3%	94.4%	97.22%	**100%**	72.2%	83.3%
User 2	61.1%	55.56%	**97.22%**	94.4%	72.2%	61.1%

Recall that learning the matrix A is equivalent to finding a transformation W ($W = A^{1/2}$) of the trajectories so that the pairs in ML are more similar to each other and the pairs in CL are different from each other. To visually illustrate this aspect, we plot the transformed trajectories learned from the first and second user constraints, respectively. The transformation corresponds to the distance measure leading to the best clustering accuracy for each case. In other words, the transformation in Fig. 3 is based on the full matrix A with DTW_A learned from the first user's constraints, and the transformation in Fig. 4 corresponds to the diagonal matrix A with DTW_A learned from the second user's constraints.

As one can see in Fig. 3, after the transformation, the trajectory samples from G1 (Fig. 3(a)) and G4 (Fig. 3(d)) are very similar to each other. Also the samples from G2 (Fig. 3(b)) and G3 (Fig. 3(c)) look similar now. On the other hand, in Fig. 4, the samples from G1 (Fig. 4(a)) and G2 (Fig. 4(b)) are similar and so are the samples from G3 (Fig. 4(c)) and G4 (Fig. 4(d)).

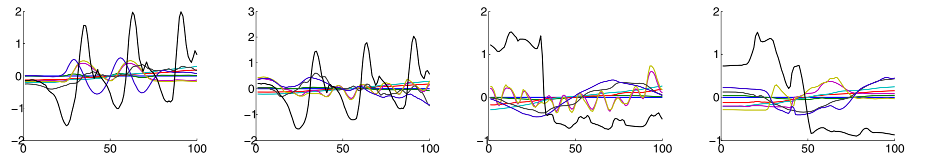
5.2 Improvement of Accuracy for Clustering and Classification

The other important application of our approach is to improve the accuracy in the context of clustering and classification. To demonstrate this feature, we performed experiments on datasets from different application domains.



(a) Sample from $G1$. (b) Sample from $G2$. (c) Sample from $G3$. (d) Sample from $G4$.

Fig. 3. Four **transformed** samples. The transformation corresponds to a full matrix A when DTW$_A$ is learned from the first user's constraints. Samples from G1 and G4 show similarity, and the G2 sample is very similar to the G3 sample.



(a) Sample from $G1$. (b) Sample from $G2$. (c) Sample from $G3$. (d) Sample from $G4$.

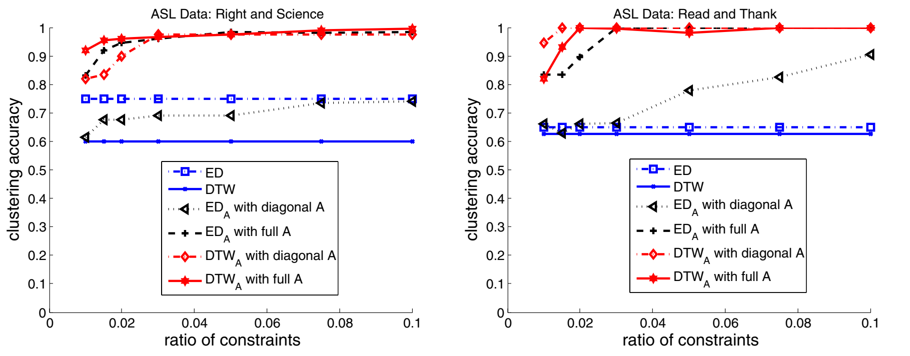
Fig. 4. Four **transformed** samples. The transformation corresponding to a diagonal matrix A for DTW$_A$ is learned from the second user's constraints. Samples from G1 and G2 show similarity, and the G3 sample is very similar to the G4 sample.

Application to Clustering. First, we show that our method can be applied to improve the accuracy of clustering. Assume a trajectory dataset $S = \{s_i\}_{i=1}^{u}$ and constraints ML and CL specified by a user. As mentioned before, $(s_i, s_j) \in ML$ means that the user considers s_i and s_j to belong to the same cluster, and for $(s_i, s_j) \in CL$, s_i and s_j belong to different clusters. We applied our approach to two labeled multi-dimensional time series datasets, ASL and Trace.

The Australian Sign Language (ASL) dataset [13] consists of the hand trajectories of a native ASL speaker when he expressed signs. We use the cleaned dataset from the UCR data archive [14]. The dataset has 10 classes, and each class has 20 examples. The 20 examples in each class represent the same word in ASL, each example having 8 features. All trajectories are interpolated to the length of 30. In this clustering experiment, we use two pair of classes, which represent the signs "read" and "thank", "right" and "science", respectively.

The Transient Classification Benchmark (Trace) dataset [14] is synthetic and has 16 classes. We use the class pairs (2 and 3), and (6 and 7) in the clustering experiments. We use 20 instances from each class, where each instance has 4 features. All data are interpolated to the same length of 50.

For each pair of classes, we combine the data into one set and perform the group-average hierarchical clustering algorithm to distinguish them. The computation of the clustering accuracy is the same as the one used for myMotion dataset in Sec. 5.1. With each dataset, the ML and CL constraints are generated as follows. ML is generated by selecting a random subset of all pairs of data having the same class label. Analogously, CL includes a random subset of all pairs of data with different class labels.



(a) Words 'Right' and 'Science' from the ASL dataset (b) Words 'Read' and 'Thank' from the ASL dataset

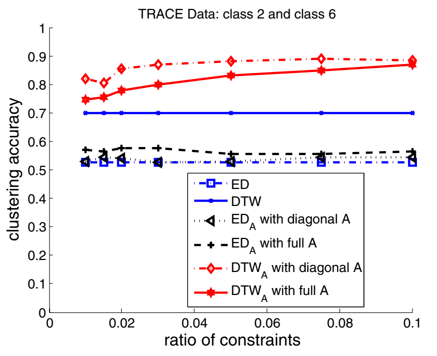
Fig. 5. Clustering accuracy vs size of constraints; the x axis is the fraction of all pairs of data sharing the same (different) class label(s) that are sampled to be included in ML (CL); the y axis shows the clustering accuracy

In order to discover the relationship between clustering accuracy and size (i.e., number) of constraints, we generated sets of constraints of different sizes and determined the clustering accuracy for each set. We repeated the experiment five times, because of the randomness of the selection of constraints. The average accuracy of the five trials is shown in Fig. 5, which shows the plot of clustering accuracy vs. size of constraints for the ASL dataset. The results for the Trace dataset are shown in Fig. 6.

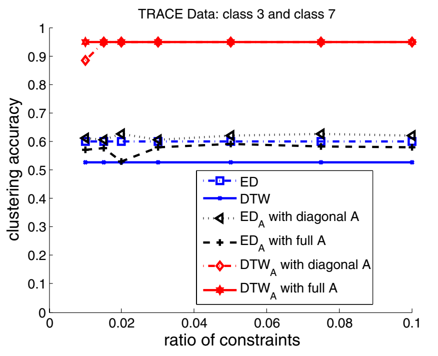
As one can see in Fig. 5 and Fig. 6, the clustering accuracy is obviously affected by the size of the constraints. We tried constraint sizes from 1% to 10% of all pairs of data with the same label. Clearly the accuracy of ED and DTW does not change with the size of the constraints as they do not consider constraints. Both ED$_A$ with diagonal and full matrix A and DTW$_A$ with diagonal and full matrix A have a better performance when the number of constraints increases, except in Fig. 6(b), where DTW$_A$ with full matrix A achieves high accuracy even when the constraint size is very small. In Fig. 5, when the number of constraints is small, DTW$_A$ performs better than ED$_A$. If we specify a larger size of constraints, ED$_A$ achieves a performance comparable to the one of DTW$_A$. However, for the Trace dataset, whose results are shown in Fig. 6, DTW$_A$ always performs better than ED$_A$. The reason for this is that there is more distortion in the time axis in the Trace dataset than in the ASL dataset.

Application to Classification. The learning approach presented in this paper can also be used to improve the accuracy of classification. In this experiment, we use three labeled multi-dimensional time series datasets.

50*CommonWords data.* The 50CommonWords dataset from the UCR data repository [14] contains 50 distinct words, each of which has various handwriting instances. Each instance is represented by 4 features describing the handwriting of a word. The instances of the same word are considered data having the same



(a) Classes 2 and 6 in the Trace dataset



(b) Classes 3 and 7 in the Trace dataset

Fig. 6. Clustering accuracy vs size of constraints; the x axis is the fraction of all pairs of data having the same (different) class label(s) that are sampled to be included in ML (CL); the y axis shows the clustering accuracy

label. For simplicity, we picked the instances of the commonly used words "of", "be", and "at". "of" has 54 instances, "be" has 38 instances, and "at" has 22 instances. All instances were interpolated to the length of 50.

ASL dataset. Here we used the classes "girl", "come" and "name", which are different from the data in the clustering experiment.

Trace dataset. We used the classes 9, 11, and 13 here to use data different from the clustering experiment.

We evaluate the application of our distance measure learning approach to classification by using an objective evaluation framework proposed by Ding et al. [4]. The general idea is to use a cross validation method and a 1-nearest neighbor (1NN) classifier for the labeled data to evaluate the accuracy of the learned distance measure. Assume a labeled dataset partitioned into a training dataset and a test dataset. For each data item in the test dataset, we predict its label to be the same as the label of its nearest neighbor in the training dataset. If the predicted label is the same as the actual label, it is considered a hit, otherwise it is considered a miss.

In order to conduct the k cross validation, the labeled dataset is randomly partitioned into k sets. The k cross validation has k runs. In each run, one of the k sets is chosen as the training dataset, and the other $k-1$ sets are used as the testing dataset. The ML and CL constraints are generated from the training dataset in the same way as in the clustering experiments described above. For the generated constraints, the distance learning algorithms are invoked to generate the distance measures (1) ED_A with a diagonal matrix A and a full matrix A, and (2) DTW_A parameterized with a diagonal matrix A and a full matrix A. For each of the resulting four distance measures, we conduct the 1NN classification to evaluate its accuracy. We also apply the 1NN classification to the pre-defined distance measures ED and DTW for comparison. We used the

Table 2. The classification accuracy of six distance measures for the three datasets. The numbers in bold indicate the best accuracy for each dataset. In general, for all datasets, the best accuracy is obtained by using the learned distance measures.

	ED	DTW	ED_A, diag. A	ED_A, full A	DTW_A, diag. A	DTW_A, full A
ASL	78.75%	75%	**95.42%**	93.75%	91.67%	90.83%
Trace	39.58%	52.92%	41.67%	40.27%	**60.42%**	57.50%
50Com.	90.94%	95%	91.25%	90.94%	**95.94%**	95.31%

LB-Keogh lower bounding [15] to speed up the computation for DTW and DTW_A and the following formula to compute the accuracy for each run:

$$\text{Accuracy} = \frac{\text{number of correctly labeled testing data}}{\text{number of all testing data}}.$$

Because there are k runs, we compute the average of the accuracy of k runs, as recommended by Ding et al. in [4]. The accuracy measures how well the predicted labels match the actual labels of the testing data. In this experiment, we use $k = 5$, which is within the range recommended by [4] to minimize the bias and variation. Table 2 lists the classification accuracy for all the datasets. As one can see, for all datasets, the best accuracy is obtained by using the learned distance measures. DTW_A performs best on the Trace and 50CommonWords datasets and ED_A performs best for the ASL dataset.

Note that ED_A performs better than (resp. equal to) DTW_A on the ASL dataset in classification (resp. clustering). This is consistent with the fact that ED performs better than DTW on the ASL dataset in all experiments, indicating that there is not much distortion in the time axis in this dataset. In a practical use, if one can obtain some more information about the time distortion in the dataset, one then can decide which distance measure performs better.

6 Conclusions and Ongoing Work

Distance measures play an important role in many data mining and analysis tasks for multi-dimensional object trajectories. Instead of relying on some of the existing, hard-coded distance measures and to support different user views on what trajectories are (dis)similar in a particular domain, in this paper, we presented a comprehensive approach for learning distance measures from user constraints. A key idea is to pose the proposed learning approach as an optimization problem that effectively utilizes well-known techniques. Our evaluations demonstrate that the learned transformed Euclidean and transformed DTW not only provide a high degree of adaptivity to user constraints but also achieve a high degree of accuracy in clustering and classification tasks for object trajectories. In general, the proposed techniques provide much more flexibility to support different user views than existing approaches.

We are currently studying the performance of the learning approach using additional datasets from other domains, in order to better evaluate the choice between transformed Euclidean and transformed DTW distance. Another interesting aspect, along the line of [20], is to derive an index structure for indexing the trajectories once a distance measure has been learned.

Acknowledgments. This work is in part supported by the National Science Foundation under Award No. ATM-0619139.

References

1. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In: AAAI Workshop on Knowledge Discovery in Databases (1994)
2. Chen, L., Ng, R.T.: On the marriage of Lp-norms and edit distance. In: VLDB, pp. 792–803 (2004)
3. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: ACM SIGMOD 2005, pp. 491–502 (2005)
4. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. In: VLDB 2008, pp. 1542–1552 (2008)
5. Domeniconi, C., Gunopulos, D.: Adaptive Nearest Neighbor Classification using Support Vector Machines. In: Advances in Neural Information Processing Sytems, vol. 13, pp. 665–672. MIT Press, Cambridge (2002)
6. Duffin, K.L., Barrett, W.A.: Spiders: A New User Interface for Rotation and Visualization of N-dimensional Point Sets. In: IEEE Visualization 1994, pp. 205–211 (1994)
7. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: ACM SIGMOD 1994, pp. 419–429 (1994)
8. Frentzos, E., Gratsias, K., Theodoridis, Y.: Index-based Most Similar Trajectory Search. In: ICDE, pp. 816–825 (2007)
9. Grumbach, S., Rigaux, P., Segoufin, L.: Manipulating Interpolated Data is Easier than you Thought. In: VLDB 2000, pp. 156–165 (2000)
10. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
11. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD 2007, pp. 593–604 (2007)
12. Lee, S.L., Chun, S.J., Kim, D.H., Lee, J.H., Chung, C.W.: Similarity Search for Multidimensional Data Sequences. In: ICDE 2000, pp. 599–608 (2000)
13. Kadous, M.W.: Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. Ph.D. Thesis, School of Computer Science and Engineering, University of New South Wales (2002)
14. Keogh, E.J.: The UCR Time Series Data Mining Archive, University of California at Riverside, `http://www.cs.ucr.edu/~eamonn/TSDMA`
15. Keogh, E.J.: Exact Indexing of Dynamic Time Warping. In: VLDB 2002, pp. 406–417 (2002)
16. Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: ACM SIGMOD 1997, pp. 289–300 (1997)
17. Mokbel, M.F., Aref, W.G.: Location-aware Query Processing and Optimization: A Tutorial. In: Presented at the 8th Int. Conf. on Mobile Data Management (2007)
18. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech and Signal Processing 26(1), 43–49 (1978)

19. Tsotras, V.J.: Recent Advances on Querying and Managing Trajectories. In: Tutorial at the 10th International Symposium on Spatial and Temporal Databases (2007)
20. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.J.: Indexing multi-dimensional time-series with support for multiple distance measures. In: KDD 2003, pp. 216–225 (2003)
21. Vlachos, M., Gunopulos, D., Das, G.: Rotation invariant distance measures for trajectories. In: KDD 2004, pp. 707–712 (2004)
22. Fu, A.W.-C., Keogh, E.J., Lau, L.Y.H., Ratanamahatana, C.: Scaling and time warping in time series querying. In: VLDB 2005, pp. 649–660 (2005)
23. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems, vol. 15, pp. 505–512 (2003)
24. Yanagisawa, Y., Akahani, J., Satoh, T.: Shape-Based Similarity Query for Trajectory of Mobile Objects. In: 4th Int. Conf. on Mobile Data Management, pp. 63–77 (2003)
25. Yang, L.: Distance Metric Learning: A Comprehensive Survey. Dept. of Comp. Science and Eng. Michigan State University (2006)
26. `http://wwwcsif.cs.ucdavis.edu/~yuwei/learning.html`