On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case

Mohammad Shokoohi-Yekta
University of California, Riverside
mshok002@cs.ucr.edu

Jun Wang
University of Texas, Dallas
wangjun@utdallas.edu

Eamonn Keogh
University of California, Riverside
eamonn@cs.ucr.edu

ABSTRACT

In the last decade, Dynamic Time Warping (DTW) has emerged as the distance measure of choice for virtually all time series data mining applications. This is the result of significant progress in improving DTW's efficiency, and multiple empirical studies showing that DTW-based classifiers at least equal the accuracy of all their rivals across dozens of datasets. Thus far, most of the research has considered only the one-dimensional case, with practitioners generalizing to the multi-dimensional case in one of two ways. In general, it appears the community believes either that the two ways are equivalent, or that the choice is irrelevant. In this work, we show that this is not the case. The two most commonly used multi-dimensional DTW methods can produce different classifications, and neither one dominates over the other. This seems to suggest that one should learn the best method for a particular application. However, we will show that this is not necessary; a simple, principled rule can be used on a case-by-case basis to predict which of the two methods we should give credence to. Our method allows us to ensure that classification results are at least as accurate as the better of the two rival methods, and in many cases, our method is strictly more accurate. We demonstrate our ideas with the most extensive set of multi-dimensional time series classification experiments ever attempted.

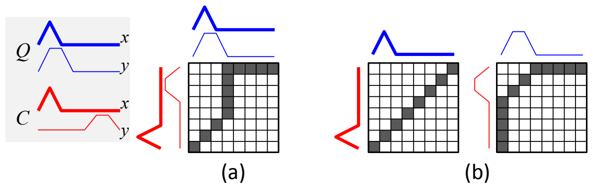
Keywords—Dynamic Time Warping; Classification

1. INTRODUCTION

There is now widespread acceptance that Dynamic Time Warping (DTW) is exceptionally difficult to beat as a time series distance measure, across a host of domain applications [2][10]. Moreover, the most often-cited reason for *not* using DTW, its relatively high time complexity, has recently become a non-issue. In particular, amortized over a subsequence search or subsequence monitoring task, DTW is slower than Euclidean Distance by less than a factor of two. As a practical matter, carefully optimized DTW is *much* faster than all but the most carefully optimized implementations of Euclidian Distance [1].

Most research on time series classification in the last two decades has focused on the single-dimensional case, with the assumption that the generalization to the multi-dimensional case is trivial. There are two obvious ways DTW can be generalized to the multi-dimensional case; Figure 1 gives a visual intuition, which we formalize later in this work. For clarity we refer to the two methods as DTW_D and DTW_I.

The vast majority of researchers seem to think that it makes no difference which method is used, as evidenced by the fact that they usually do not explicitly *tell* the reader which method is used.



(a) $DTW_D(Q,C) = DTW(\{Q_x,Q_y\},\{C_x,C_y\}) = \mathbf{3.2}$

(b) $DTW_I(Q,C) = DTW(Q_x,C_x) + DTW(Q_y,C_y) = \mathbf{2.4}$

Figure 1: *top.left*) Two multi-dimensional time series. *a*) The DTW_D distance between them is 3.2. *b*) The DTW_I distance between them is 2.4. All elements of this visual key are formally defined below.

With some introspection we can see that there are actually several possibilities:

- There is no difference between DTW_D and DTW_I; they produce the same values for all time series.

However, we can immediately dismiss this possibility; as shown in Figure 1, the two methods generally produce different distance values, and thus *could* produce different class labels if classifying an object using the Nearest Neighbor (NN) algorithm.

The next possibility seems to be the one *implicitly* assumed by the community:

- DTW_D and DTW_I *can* produce different distance values, but this makes no difference in the classification accuracy.

As we shall show, this is not the case. The choice of DTW_D vs. DTW_I *can* make a significant difference in the classification accuracy.

Given that DTW_D and DTW_I can have different classification accuracies, one might then imagine that the following is the case:

- While DTW_D and DTW_I can produce different classification accuracies, it happens that one of the two is always superior on all problems. If we could prove or experimentally demonstrate this, we could "retire" the weaker measure.

This idea is tempting, and has some precedents in similar situations in the literature. However, as we shall show, it is not the case. Datasets exist where DTW$_D$ significantly outperforms DTW$_I$ and vice-versa.

This would appear to be the end of the discussion. For a given problem we can use cross-validation to determine which method to use, then simply hard-code it into our classifier. However, there are two reasons why this is *not* the last word. First, we do not have the luxury of cross-validation when we have very small training sets, a situation that is very common when *cold-starting* a gesture recognition system, or when labeled data is expensive. Secondly, we are not done moving down the hierarchy of possibilities. In particular:

- For any given domain, it may be that on an individual *class-by-class*, or even *exemplar-by-exemplar* basis, DTW$_D$ and DTW$_I$ can produce different results, and that we could predict which of the two methods to trust at classification time.

This possibility is less intuitive than the others. It is not clear that the utility of the measures should vary within a single domain, and if it did, correctly predicting which measure was most likely to have been correct on a case-by-case basis seems like an untenable undertaking.

In this work, we show for the first time that this last possibility is correct. The utility of DTW$_D$ and DTW$_I$ varies on an instance-by-instance basis, and our technique, DTW$_A$ (DTW$_{Adaptive}$) can predict at run time, with high accuracy, which of them is more likely to be correct. Our work has two implications for the time series community: we free researchers/implementers from having to decide which technique to use for their problem; and because *error*(DTW$_A$) will be *minimum*[*error*(DTW$_D$), *error*(DTW$_I$)], they can use our method safe in the knowledge that they did not choose the suboptimal method.

However, this greatly understates the case, as the correct inequality implied by our work is the more unintuitive *error*(DTW$_A$) $\leq$ *minimum*(*error*(DTW$_D$), *error*(DTW$_I$)). That is to say, on some datasets our method can be significantly more accurate than *either* of the rival methods.

2. DEFINITIONS AND BACKGROUND

For our task at hand, each object in the dataset is a *time series*, which may be of different length.

***Definition* 1**: A *Time Series* $T = t_1, t_2, ..., t_n$ is an ordered set of real values. The total number of real values is equal to the length of the time series. A dataset $\boldsymbol{D} = \{T_1, T_2, ..., T_M\}$ is a collection of M such time series.

We are interested in multi-dimensional time series:

***Definition* 2**: *Multi-Dimensional Time Series* (MDT) consist of M individual time series ($M \geq 2$) where each time series has n observations:

$Q_1 = q_{1,1}, q_{2,1}, q_{3,1},... q_{n,1}$

$Q_2 = q_{1,2}, q_{2,2}, q_{3,2},... q_{n,2}$

...

$Q_M = q_{1,M}, q_{2,M}, q_{3,M},... q_{n,M}$

Unlike the Euclidian distance's strict *one-to-one* alignment, DTW allows a *one-to-many* alignment, as illustrated in Figure 1. To align sequences using DTW, an n-by-n matrix is constructed with the (i^{th}, j^{th}) element being the squared Euclidean distance $d(q_i, c_j)$ between the points q_i and c_j. A warping path P is a contiguous set of matrix elements defining a mapping between Q and C. The t^{th} element of P is defined as $p_t = (i,j)_t$, so we have:

$$P = p_1, p_2, ..., p_t, ..., p_T \quad n \leq T \leq 2n - 1$$

The warping path that defines the alignment between the two time series is usually subject to several constraints: the warping path must start and finish in diagonally opposite corner cells of the matrix, the steps in the warping path are restricted to adjacent cells, and the points in the warping path must be monotonically spaced in time. In addition, virtually all practitioners using DTW also constrain the warping path in a global sense by limiting how far it may stay from the diagonal [2][10]. A typical constraint is the Sakoe-Chiba Band which states that the warping path cannot deviate more than R cells from the diagonal [2][10][15]. This constraint prevents pathological warpings (for example, a single heartbeat mapping to ten heartbeats) and is at the heart of the LB$_{Keogh}$ lowerbounding technique used in virtually all speedup techniques for DTW [1][2].

While there are exponentially many warping paths that satisfy the above conditions, we are only interested in the path that minimizes the warping cost:

$$(1) \qquad DTW(Q,C) = \min\{\sqrt{\sum_{t=1}^{T} p_t}\}$$

This path can be found using dynamic programming to evaluate the following recurrence, which defines the cumulative distance $D(i,j)$ as the distance $d(i,j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements [8][13]:

$$(2) \qquad D(i,j) = d(q_i, c_j) + \min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}$$

in which: $d(q_i, c_j) = (q_i - c_j)^2$

While this recursive function is elegant and can be tersely implemented, in practice the community uses an *iterative algorithm*, which is faster and amiable to various early abandoning optimizations [1][2]. Moreover, the iterative algorithm implementation only constructs and considers a single column of the matrix at a time, and thus has a space complexity for just O(n).

The Euclidean distance between two sequences is a special case of DTW, where the t^{th} element of P is constrained such that $p_t = (i,j)_t$, $i = j = t$. This review of DTW is necessarily brief; we refer the interested reader to [1][2][8][13] for more details.

2.1 Generalizing to the Multi-Dimensional Case

The DTW distance as formalized in Eq. 2 is applicable to only single-dimensional time series, leaving open the question of how to extend it to the multi-dimensional time series (MDT) case. Consider both Q and C as two M-dimensional time series; we show two possible approaches for doing this, DTW_I and DTW_D:

***Definition* 3**: DTW_I is the cumulative distances of all dimensions independently measured under DTW. If $DTW(Q_m, C_m)$ is defined as the DTW distance of the m^{th} dimension of Q and the m^{th} dimension of C, we can write DTW_I as:

$$DTW_I(Q,C) = \sum_{m=1}^{M} DTW(Q_m, C_m) \tag{3}$$

In Eq. 3, each dimension is considered to be independent and DTW is allowed the freedom to warp each dimension independently of the others. The case when M is two was shown in Figure 1.*b*.

We can also compute the multi-dimensional DTW in a manner that forces all dimensions to warp identically. In other words, the independence of dimensions is no longer allowed and we assume mutual dependence between all dimensions. We define DTW_D as:

***Definition* 4**: DTW_D is calculated in a similar way to DTW for single-dimensional time series (Eq. 2), except that we redefine $d(q_i, c_j)$ as the cumulative squared Euclidean distances of M data points instead of the *single* data point used in the more familiar one-dimensional case. Formally, if $q_{i,m}$ is the i^{th} data point in the m^{th} dimension of Q and $c_{j,m}$ is the j^{th} data point in the m^{th} dimension of C, we replace $d(q_i, c_j)$ in (Eq. 2) with:

$$d(q_i, c_j) = \sum_{m=1}^{M} (q_{i,m} - c_{j,m})^2 \tag{4}$$

To make our distance measure invariant to scale and offset, we need to *z-normalize* each dimension of the time series before computing their DTW distance. As demonstrated in [6], even tiny differences in scale and offset rapidly swamp any similarity in shape. Note that this allows us to meaningfully compute either variant of the multi-dimensional DTW, even if the individual dimensions are not commensurate or are in different units, such as *accelerations* and *rotations*.

Using DTW_D and DTW_I distance measures to classify a time series exemplar, T, four different cases may occur:

1. T gets correctly classified by both DTW_I and DTW_D.
2. T gets misclassified by both DTW_I and DTW_D.
3. T gets classified correctly by DTW_I but misclassified by DTW_D.
4. T gets classified correctly by DTW_D but misclassified by DTW_I.

We are only interested in cases 3 and 4. We call such exemplars *iSuccess* and *dSuccess*, respectively:

***Definition* 5**: *iSuccess* is the set of time series exemplars that are classified correctly under DTW_I but misclassified under DTW_D.

***Definition* 6**: *dSuccess* is the set of time series exemplars that are classified correctly under DTW_D but misclassified under DTW_I.

Having reviewed the necessary formal definitions, we are now in a position to introduce our observations about the relative merits of DTW_D and DTW_I.

3. OBSERVATIONS

We begin with some informal notation. We say a dataset is "in *D*" if we expect DTW_D to achieve higher accuracy, and "in *I*" if we anticipate DTW_I will be more accurate. In the introduction we claimed that there are datasets in which we expect DTW_D to outperform DTW_I and vice versa. A natural question to ask is under what conditions we can expect each of these methods to be superior. As we shall see, one of the fundamental contributions of this work is to make this question moot, by producing an algorithm that is always *at least as good* as the better choice. Nevertheless, it is instructive to ask and answer this question.

Assume that the data in question corresponds to an *event*. An event could be an arrhythmic heartbeat, the writing of the letter 'Z,' a golf swing, a bird call, a self-driving car parallel parking itself, etc. Further assume that we have multi-dimensional time series recordings of such events. It is possible that each dimension is simply recording two views of the same physical phenomena. For example, consider the MFCC coefficients of the bird call shown in Figure 2.

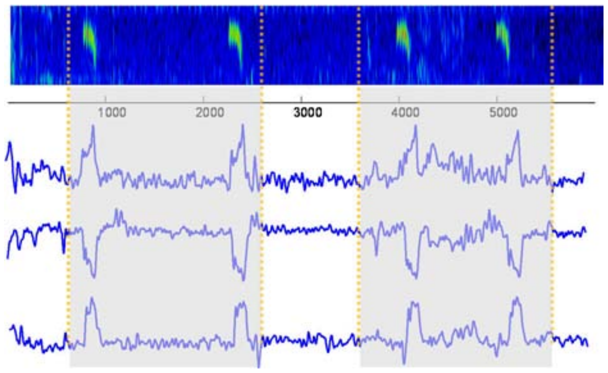


Figure 2: Three coefficients from the MFCC space of a Southern Chestnut-Tailed Antbird (*Myrmeciza hemimelaena*). This bird's call is typically transcribed as "*klee-klee*" [3]; thus, the above shows two calls, the second being significantly briefer.

It is clear that while the coefficients are (somewhat) independent in the Y-axis values they can take on, they are *not* independent in the time axis. In the second bird call, all the relevant peaks and valleys move by *exactly* the same amount in the time axis. Because of this structure, we expect that this dataset is in *D*. In contrast, consider the *event* shown in Figure 3 of a cricket umpire signaling `TV-Replay`, in which the umpire traces a rectangle representing a television screen.

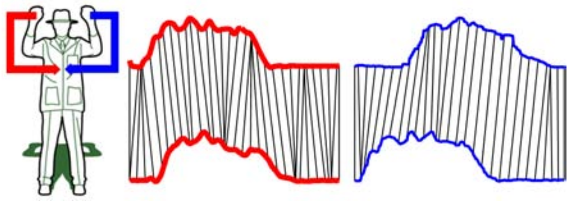


Figure 3: (*left*) A cricket umpire signaling "`TV-Replay`." (*right*) The Dynamic Time Warping distance between two time series of the X-axis from the right and left hand.

Here we have measured a multi-d time series that consists of the X-axis acceleration of sensors worn on each wrist. Note that in contrast to the bird call example, the two time series are very unlikely be *perfectly* dependent in how they evolve in the time axis. Try as he might, the umpire could not move both hands in a *perfectly* symmetric fashion. There are at least two possible and non-exclusive sources of difference:

- **Lag**: Here we imagine that the umpire favors his dominant hand, and the other hand follows at a more or less constant speed of a fraction of a second behind.
- **Loose Coupling**: Here the *event* does cause two or more things to happen, both of which are recording as a time series, but there is more freedom in the performance of the event. For example, if the *event* is the umpire wishing to signal a `Leg-Bye`, he will tap his raised knee with this hand. However, while he typically uses his dominant hand to do this, he may touch *either* knee. Moreover, his "free" hand may rest by his side, or he may raise it, and even waive it slightly, to drawn attention to the fact he is making a signal. This variability in performance means that two wrist-worn sensors are only loosely coupled for this event.

In the next section, we will explicitly test the effects of these factors with some experiments. To do this we consider a dataset that we are sure is in *D*, and then we synthetically add increasing amounts of lag and loose coupling to see if this would move the dataset into *I*.

We consider a *handwriting* dataset, which we are sure is in *D*. Because we are considering the X and Y accelerations of the point of the writing tip of a pen, the two dimensions are physically coupled. Here an *event* is the production of one of the twenty-six lower-case letters. We estimate the error rate of classification by randomly dividing the 5,000 objects into a stratified 1,000/4,000 train test split thirty times and reporting the average error rate.

It is critical to note that in this dataset, if we were considering only the one-dimensional case, our synthetic modifications of the data would make essentially no difference to the distances DTW returns, or the overall error rate. As shown in Figure 4, even a huge change in the lag makes almost no difference to the single-dimensional DTW case.

Thus, all effects shown in the next two sections are due not to the effects of modifying the data objects per se, but to the effect this has on DTW_D and DTW_I.

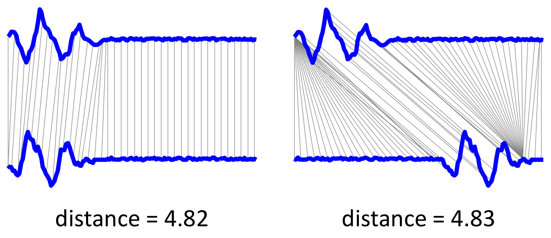


Figure 4: For the one-dimensional case, adding lag to one of the time series can change the *warping* drastically, without changing the *distance* by a significant amount.

3.1 The Effect of Lag

We induce lag in one of two ways. First, for each object we add *Random Lag* of *K* by shifting *just* the Y-axis by an amount randomly chosen from the range [0, *K*]. Second, we add a *Fixed Lag* variant by shifting *just* the Y-axis by increasing values of *K*. For clarity, in this variant all objects will have the same lag of exactly *K*. In Figure 5 we show the effect of varying the amount of *Random Lag* from zero to forty.

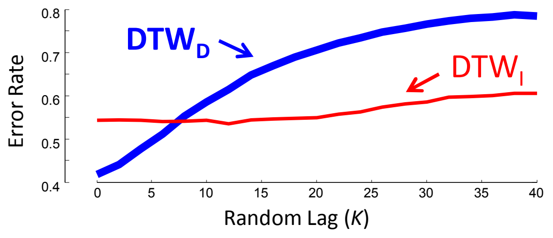


Figure 5: The effect of adding *Random Lag* on the classification accuracy of DTW_D and DTW_I .

In Figure 6 we show the effect of varying the amount of *Fixed Lag*, again from zero to forty.

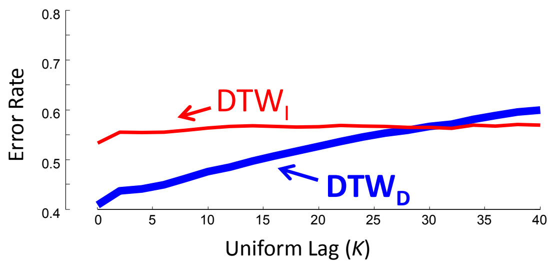


Figure 6: The effect of adding *Fixed Lag* on the classification accuracy of DTW_D and DTW_I .

If we consider just the values at either end of the range, this provides us with the first experimental evidence that datasets exist that are strongly in *D*, along with datasets (albeit *contrived* here) strongly in *I*. More generally, the trend lines in the figure confirm our claim that *Lag* is one of the elements responsible for datasets falling in *D* or *I*. Note that the effects are not small; the wrong choice of DTW_D or DTW_I here can almost double the error rate.

3.2 The Effect of Loose Coupling

We create synthetic loose coupling by adding increasing amounts of random warping to just the Y-axis of each exemplar. For brevity and to enhance the flow of the presentation, we relegate the explanation (and the actual code) of how we do this to the supporting website [19] . However, we note that the modified data is plausible and realistic data. For example, if we use it to regenerate the original letters, they are readable and believable as a person's handwriting. Figure 7 shows the effect of increasingly loose coupling.

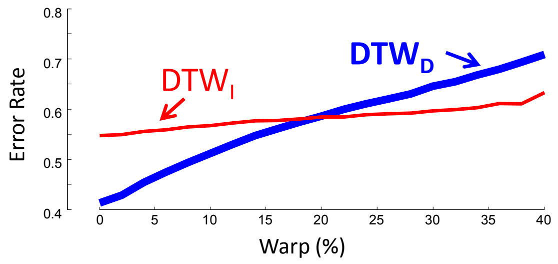


Figure 7: The effect of adding *Fixed Warping* on the classification accuracy of DTW_D and DTW_I .

Once again, these results provide us with evidence that some datasets are in *D* and some are in *I*, and that loose coupling can be a reason for this division.

3.3 Implication of Observations

At first blush we might interpret the above results as implying that all datasets lie on a spectrum between being strongly in *D* and strongly in *I*. If true, then the only task left to us is to discover where on the spectrum a dataset falls, so that we can use the correct technique.

However, this idea has two difficulties. For some datasets we may not have enough training data to learn whether we are in *D* or in *I* with high confidence. The second issue is simply that the assumption that all *datasets* lie on such a spectrum misses a crucial point. It is possible that the suitability for DTW_D or DTW_I occurs at a *class-by-class* level, or even an *exemplar-by-exemplar* level, not at a *dataset-by-dataset* level.

It is easy to imagine such examples. Suppose we have accelerometers on both wrists of a tennis player, and our classification task is to label data into the following shot types `{serve|forehand|lob|other}`. For many exemplars we might expect DTW_I to work best, since the hands are generally loosely coupled in tennis. However, for some classes, such as the `backhand`, most players use a two-handed grip, temporarily coupling the two accelerometers. This would give us a *class-by-class*-level difference in the suitability of the warping technique. Moreover, some players, notably French professional Jo-Wilfried Tsonga, switch between one-handed and two-handed back-hand shots during the game. This would give us an *exemplar-by-exemplar*-level difference in the suitability of the warping technique.

4. PROPOSED FRAMEWORK

In essence, our task reduces to a meta-classification problem. Given an instance to classify, we must first decide whether it is "an object best classified by DTW_I" or "an object best classified by DTW_D." More formally:

Problem Statement: Given we are using NN-DTW to classify an exemplar *Q*, and that we have discovered the nearest neighbor to *Q* under both DTW_I and DTW_D, if the classes of the two nearest neighbors differ, predict the distance function most likely to be correct.

Our problem statement is superficially similar to a "gating network" in a Mixture of Experts (ME), a technique frequently used in neural networks and some other classifiers [18]. Using the divide and conquer principle, several "experts" that are either regression functions or classifiers are created, such that they specialize in a particular region of the input space. The gating network defines those regions where the individual expert opinions are trustworthy, and uses a probabilistic model to combine the expert's opinions.

There are many variants of ME (see [18] and references therein). However, in contrast to most versions, we have a much narrower task. We have exactly two experts, and we are "weighting" their opinions only in a strictly binary sense.

We propose the following solution to our problem. Offline, on the training data, we will compute a *threshold*. At query time, we will compute a score *S*, and choose which method to trust based on the value of *S* relative to the *threshold*. In order to best explain our framework, we first explain *how* our classification model works given that a *threshold* has been learned from the *trainData.* Later, in Section 4.2*,* we consider the task of *learning* the *threshold*. If we have lack of labeled data, we outline two possible approaches. We can either learn the *threshold* from a different dataset from the same or similar domain or we can simply hardcode the *threshold* to one which gives us much of the benefit of our observation. We have explored both ideas in [19].

4.1 Adaptive Classifier for MDT

Table 1 outlines our classification algorithm. In line 1 the algorithm finds the nearest neighbor distance in the training set for *Q* under DTW_D, *minD*. In line 2 we find the nearest neighbor distance under DTW_I , *minI*. In line 3 the procedure divides *minD* by *minI*, which is our scoring function, *S*. In lines 4 to 8 the algorithm compares our scoring function, *S*, to the previously learned *threshold*. If *S* is greater than the *threshold*, we believe that *Q* is most likely in *I* and thus return DTW_I as the distance measure for classification, whereas if *S* is less than or equal to the *threshold*, we predict that *Q* is most likely in *D* and the function returns DTW_D.

Table 1: Adaptive classification algorithm

```
Procedure adaptive_Classifier (Q, trainData, threshold)
Input: A time series query, Q, the labeled data, trainData, a threshold;
Output: An adaptive distance measure to classify Q, $DTW_A$;
1   minD ← Nearest_Neighbor_Distance_D (Q, trainData);
2   minI ← Nearest_Neighbor_Distance_I (Q, trainData);
3   S ← minD / minI;
4   if S > threshold
5       $DTW_A$ ← $DTW_I$ ;
6   else
7       $DTW_A$ ← $DTW_D$ ;
8   end if
9   Return $DTW_A$
```

We formally define our scoring function as:

$$S = \frac{\textit{Nearest Neighbor Distance under } DTW_D}{\textit{Nearest Neighbor Distance under } DTW_I + \epsilon} \qquad (5)$$

The *epsilon* in the denominator is to prevent division by zero, a theoretical possibility but never observed. While S can change in the range of $[\varepsilon, \infty]$, in practice we find its value to always be in the range of [0.5, 2].

Having explained our simple classification algorithm, all that remains is to explain how we set the threshold (and *why*). In fact, hardcoding the threshold to a value of exactly *one* works very well and allows the algorithm in Table 1 to beat the rival methods. However, we can further improve the accuracy by tuning the threshold, so in the next section we will explain how that is done.

4.2 Learning the Adjusted Threshold

In order to learn the *threshold* we use in Table 1 , we first need to identify the *iSuccess* (*def.* 5) and *dSuccess* (*def.* 6) exemplars in the training set using cross validation (Table 3). As shown in Table 2, we consider four cases based on whether *iSuccess* and *dSuccess* are empty sets or not.

Table 2: Learning the adjusted threshold

```
Procedure Learn_Threshold (trainData)
Input: Labeled data, trainData;
Output: Adjusted threshold, threshold;
1   [S_iSuccess, S_dSuccess] ← find_Scores (trainData);
2   if (S_iSuccess == $\phi$ && S_dSuccess == $\phi$ )
3       threshold ← 1;
4   else if (S_iSuccess == $\phi$ && S_dSuccess != $\phi$ )
5       threshold ← max(S_dSuccess) ;
6   else if (S_iSuccess != $\phi$ && S_dSuccess == $\phi$ )
7       threshold ← min(S_iSuccess) ;
8   else if (S_iSuccess != $\phi$ && S_dSuccess != $\phi$ )
9       threshold ← Decision_Tree (S_iSuccess, S_dSuccess);
10  end if
```

In line 1 we run the subroutine in Table 3 to find all the *S* scores for *iSuccess* and *dSuccess*, and then we consider four cases on the two sets. Line 2 is the case in which both sets are empty, so the problem (at least the training data) is independent of *D* or *I*, and picking either DTW_I or DTW_D will make no difference in classifying the data. Therefore, we assign the value one, an arbitrary number, to the *threshold* in line 3. We note that this case is possible, but we never observed it.

In line 4 we test to see if *S_iSuccess* is empty and *S_dSuccess* is non-empty. If so, the dataset is almost certainly in *D*, and we need to set the *threshold* such that the *S* score for all *dSuccess* exemplars will be less than the *threshold*. Therefore, in line 5 the *threshold* gets assigned to the maximum value of *S_dSuccess.*

The opposite case, in which *S_dSuccess* is empty and *S_iSuccess* is non-empty (line 6), offers evidence that the dataset is in *I,* and we need to set the *threshold* such that the *S* score for all *iSuccess* exemplars will be greater than the *threshold*. We ensure this (in line 7) by assigning the *threshold* to the minimum value of *S_iSuccess*.

In practice, the three cases above are rare, and in lines 8 to 10 we find the *threshold* for the most common case, in which both *S_iSuccess* and *S_dSuccess* are non-empty sets. The best *threshold* is a value that maximizes the total number of *S_iSuccess* with values greater than the *threshold* and *S_dSuccess* with values less than the *threshold*. Finding such a *threshold* is essentially the decision tree problem of maximizing the *information gain* by finding the optimal split point. For more details on information gain, we refer the interested reader to [17][12].

Recall the function *find_Scores* (*trainData*) called in Table 2. The function uses cross validation to find the two sets *iSuccess* and *dSuccess,* then calculates the *S* scores (Eq. 5) for all their exemplars. The algorithm is described in Table 3. In line 1 we apply cross validation to the entire *trainData*. In line 2 we calculate the nearest neighbor distance under DTW_D for each exemplar, and in line 3 we do the same under DTW_I. In lines 4 to 7, if the exemplar in the *trainData* is classified correctly under DTW_D and misclassified under DTW_I, the *S* score (Eq. 5) is calculated and gets added to *S_dSuccess*. In line 8 to 11, if the exemplar is misclassified under DTW_D and classified correctly under DTW_I, we calculate the *S* score and add it to *S_iSuccess*.

Table 3: An algorithm to find *iSuccess* and *dSuccess* and compute *S* scores for all their exemplars

```
Procedure find_Scores (trainData)
Input: Labeled data, trainData;
Output: S scores for iSuccess and dSuccess, S_iSuccess and S_dSuccess;
1   for n ← 1 to size(trainData)
2       minD ← Nearest_Neighbor_Distance_D (trainData(n),trainData);
3       minI ← Nearest_Neighbor_Distance_I (trainData(n), trainData);
4       if (trainData(n).label == Nearest_Neighbor_D ().label &&
5               trainData(n).label != Nearest_Neighbor_I ().label )
6           S_dSuccess.add (minD / minI);
7       end if
8       if (trainData(n).label != Nearest_Neighbor_D ().label &&
9               trainData(n).label == Nearest_Neighbor_I ().label )
10          S_iSuccess.add (minD / minI);
11      end if
12  end for
```

4.3 The Intuition Behind Our Scoring Function, S

In this section we explain the intuition behind our scoring function (Eq. 5) introduced in Section 4.1. We will show how the ratio of the nearest neighbor distance under DTW_D (*minD*) and DTW_I (*minI*) is capable of discriminating multi-dimensional time series in *I* from exemplars in *D*. For any time series in *I,* each dimension is at least somewhat independent; therefore, exemplars that are from the same class and are warped/shifted independently in each dimension exist in the training set. Figure 8.*top* shows a time series in *I,* Q_I and its nearest neighbor in the training set, C_I .

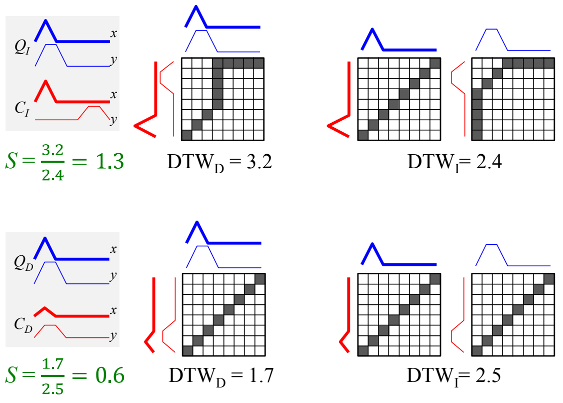


Figure 8: ***top*) Computing the *S* score for a two-dimensional time series in *I*. *bottom*) *S* score calculation of a two-dimensional time series in *D*.**

Note that in our proposed algorithm in Section 4.1, the nearest neighbors under DTW_I and DTW_D are not necessarily the same; however, for simplicity of presentation we consider C_I as the nearest neighbor under both DTW_I and DTW_D. Since Q_I is in *I* and its dimensions are warped independently, the DTW_I distance will be less than or equal to the DTW_D simply because DTW_I is allowed the freedom to find the nearest distance independently in each dimension. In Figure 8.*top*, DTW_I calculates the distance in two different paths, whereas DTW_D has only one path to pick which is a combination of the two paths in DTW_I and eventually produces a larger distance. For any instance in *I*, *minD* is larger and *minI* gets smaller; therefore, *minD*/*minI* tends to be larger.

In Figure 8.*bottom* we show an instance, Q_D, which is in *D*. In this case the nearest neighbor in the training set, C_D, will be an exemplar in which both dimensions are dependently warped. In such a case, the warping path for both dimensions in DTW_I are the same as and similar to the path in DTW_D. In contrast to the previous case, DTW_I does not take advantage of different warping paths in order to achieve a lower distance score compared to DTW_D. However, we show for the same warping path, the distance under DTW_I is larger than the DTW_D distance. Since DTW_D and DTW_I both take the same path, we can compare their cumulative distances in a meaningful way.

If $q_{i,m}$ is the i^{th} data point in the m^{th} dimension of Q_D and $c_{j,m}$ is the j^{th} data point in the m^{th} dimension of C_D, for the two-dimensional case in Figure 8.*bottom* we can prove the following:

$$DTW_D(Q_D,C_D) = \sqrt{\sum_{i,j=1}^{n}[(q_{i,1}-c_{j,1})^2+(q_{i,2}-c_{j,2})^2]} \prec$$
$$\sqrt{\sum_{i,j=1}^{n}(q_{i,1}-c_{j,1})^2} + \sqrt{\sum_{i,j=1}^{n}(q_{i,2}-c_{j,2})^2} = DTW_I(Q_D,C_D)$$

Accordingly, for a time series in *D*, *minD* is smaller and *minI* gets larger; therefore, *minD*/*minI* tends to be smaller. We considered a two-dimensional time series here and assumed that for a query in *I*, the path in DTW_I and DTW_D are exactly the same; however, we can simply generalize the above illustration to dimensions greater than two, and for queries in *I*, different but similar paths for both DTW_I and DTW_D.

We have shown that our scoring function, *S* (Eq. 5), tends to produce larger values for queries in *I* and smaller values for queries in *D*, as illustrated above; thus, our scoring function is capable of discriminating time series in *I* from exemplars in *D*. We will demonstrate the effectiveness of our scoring function with extensive experiments in the next section.

5. EXPERIMENTS

We have designed all our experiments to ensure that they are *very* easy to reproduce. A supporting webpage [19] contains *all* the code, datasets, and raw data spreadsheets used in this work. Moreover, although this work is completely self-contained, the webpage contains additional experiments for the interested reader.

In addition to comparing to DTW_D and DTW_I, we also compare to the classification using each *individual* dimension, which we refer to using the notation DTW(*1st*), DTW(*2nd*), etc. Note that all experiments use exactly the same base algorithm, one nearest neighbor, and exactly the same train/test splits. Thus, any differences in results can be attributed solely to the distance measure used. It is known that the warping window width can slightly affect the classification accuracy. As this issue is orthogonal to our work, we simply set the warping window constraint for DTW to be 20% for all experiments [2].

5.1 Recognition of Cricket Umpire Signals

Cricket requires an umpire to signal different events in the game to a distant scorer/bookkeeper. The signals are communicated with motions of the hands. For example, `No-Ball` is signaled by touching each shoulder with the opposite hand, and `TV-Replay`, a request for an off-field review of the video of a play, is signaled by miming the outline of a TV screen (*cf.* Figure 3).

The dataset introduced in [7] consists of four umpires performing twelve signals, each with ten repetitions. The data, recorded at a frequency of 184Hz, was collected by placing accelerometers on the wrists of the umpires. Each accelerometer has three synchronous measures for three axes (*X*, *Y* and *Z*). Thus, we have a six-dimensional MDT from the two accelerometers and we can combine any number of them to create a multi-dimensional classification problem. Figure 9 shows the data for two example signals, `Six` and `Leg-Bye`. To signal `Six`, the umpire raises both hands above his head. `Leg-Bye` is signaled with a hand touching the umpire's raised knee three times.

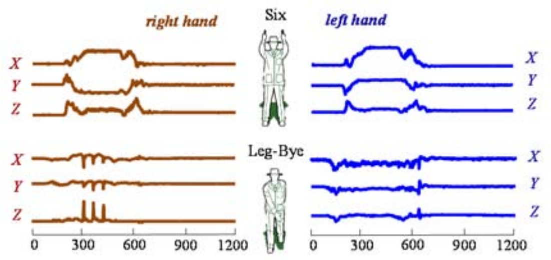


Figure 9: *X, Y* and *Z* acceleration data from the *right hand* (*left*), a representation of the umpire's body position (*center*), and the *X, Y* and *Z* acceleration data from the *left hand,* for the two umpire signals `Six` and `Leg-Bye`.

We used 40% of the data for training and use the rest as testing data. The classification results using various combinations of dimensions are shown in Table 4.

Table 4: Classification error rates on the cricket data

Data	DTW($_{1st}$)	DTW($_{2nd}$)	DTW$_I$	DTW$_D$	DTW$_A$
$X_{right}\_X_{left}$	0.26	0.27	0.13	0.20	**0.11**
$Y_{right}\_X_{left}$	0.17	0.27	0.07	0.04	**0.03**
$X_{right}\_Y_{left}$	0.26	0.14	0.07	0.10	**0.06**
$Y_{right}\_Y_{left}$	0.17	0.14	0.04	0.03	**0.03**
$Z_{right}\_Z_{left}$	0.18	0.18	0.07	0.06	**0.04**
$Z_{right}\_X_{left}$	0.18	0.27	0.07	0.04	**0.04**

Note that *all* combinations support our original claims that neither DTW$_D$ nor DTW$_I$ dominates the other, and that on all datasets, DTW$_A$ is *at least as* accurate as the better of DTW$_D$ and DTW$_I$, and often *more* accurate.

Above we considered only *pairs* of dimensions, however, the results generalize for any-sized subsets of dimensions. Obviously, adding more dimensions does not guarantee improved accuracy. In [4] the authors outline a strategy for choosing *which* dimensions to add to an MDT. Note, however, that this issue is completely orthogonal to our contributions; Table 4 suggests that whatever set of dimensions you choose, you are better off with DTW$_A$ than any other method.

5.2 Accelerometer-Based Gesture Recognition

There is increasing interest in using gesture commands for interacting with and controlling external devices. The results in [5] suggest that different people often have different interpretations of even simple gestures, and thus it is necessary to *learn* personalized classifiers on an individual basis.

A widely-used benchmark dataset introduced in [9] consists of the performances of the gestures shown in Figure 10 by eight participants. To produce realistic variability in performance, the data was collected on multiple days over three weeks. On each day the participant held a Nintendo Wii remote and repeated each of the eight gestures ten times.

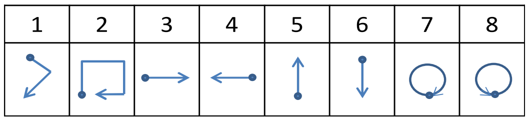


Figure 10: Gesture vocabulary adopted from [5]. The dot denotes *the start* and the arrow *the end* of the gesture.

The dataset consists of 4,480 gestures in total, 560 for each participant. The accelerometer has three axes (*X*, *Y* and *Z*); thus, we have a three-dimensional MDT form and we can combine them to create a two or three multi-dimensional classification problem. We combined every pair of dimensions to create all possible two-dimensional time series, and combined all three for the three-dimensional case. The classification results are shown in Table 5.

Table 5: Error rates for gesture recognition

Data	DTW($_{1st}$)	DTW($_{2nd}$)	DTW($_{3rd}$)	DTW$_I$	DTW$_D$	DTW$_A$
X_Y	0.34	0.48	-	0.23	0.18	**0.18**
X_Z	0.34	0.41	-	0.13	0.13	**0.12**
Y_Z	0.48	0.41	-	0.26	0.24	**0.23**
X_Y_Z	0.34	0.48	0.41	0.11	0.09	**0.08**

As before, the results support our claim that DTW$_A$ is *at least as* accurate as the better of DTW$_D$ or DTW$_I$.

5.3 Word Recognition from Vocal Data (EMA)

An ElectroMagnetic Articulograph (EMA) is an apparatus used to measure the movement of the tongue and lips during speech. We consider the EMA dataset in [16] which contains data collected from multiple English native speakers producing 25 words. Twelve sensors were used, each providing *X*, *Y* and *Z* accelerations. Of the total of 36 available dimensions, for brevity and simplicity we show only some random combinations of dimensions extracted from the sensors on the tongue tip (*T1*), the upper lip (*UL*) and lower lip (*LL*). The classification results are shown in Table 6.

Table 6: Classification error rates on the continuous articulary movement dataset

Data	DTW($_{1st}$)	DTW($_{2nd}$)	DTW($_{3rd}$)	DTW$_I$	DTW$_D$	DTW$_A$
$T1_Z\_UL_X$	0.34	0.59	-	0.25	0.31	**0.25**
$T1_Y\_UL_Y$	0.38	0.55	-	0.22	0.15	**0.13**
$T1_Z\_LL_Z$	0.34	0.47	-	0.17	0.10	**0.10**
$T1_Z\_LL_Y$	0.34	0.48	-	0.20	0.12	**0.11**
$T1_Y\_LL_Y$	0.38	0.48	-	0.21	0.14	**0.14**
$T1_Y\_T1_Z\_LL_X$	0.38	0.34	0.67	0.12	0.08	**0.08**
$T1_Z\_LL_X\_LL_Y$	0.34	0.67	0.48	0.14	0.12	**0.10**

Yet again the results support our claim that DTW$_A$ is *at least as* accurate as *the better* of DTW$_D$ or DTW$_I$.

5.4 Revisting the Semi-Synthetic Data

We conclude by revisiting the handwriting data set used in Section 3. Recall that the data is *real*, but manipulated in ways such that it changed from being in *D* to being in *I*. In Figure 11 we revisit these problems, this time using DTW_A. Once again these experiments offer strong support for our claims about DTW_A dominating DTW_I and DTW_D.

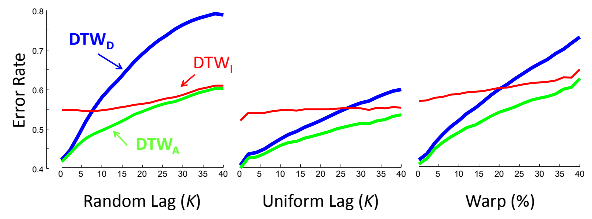


Figure 11: The experiments shown in Figure 5, Figure 6 and Figure 7, revisited using the DTW_A technique.

6. CONCLUSIONS AND RELATED WORK

We have deferred a discussion of related work until now, when the reader can appreciate the nature of our contributions. While there are hundreds of research efforts that use DTW in a multi-dimensional setting [3][5][7][9][11][16], we are not aware of any work that discusses the relative merits of DTW_I and DTW_D, or even explicitly notes that they are alternatives. The ubiquity of multi-dimensional time series, especially given the recent explosion of interest in wearable devices, has produced significant research in speeding up DTW [14], choosing which subset of dimensions to use [4], choosing a setting for the warping window constraint [2], etc. However, all such work is orthogonal to (and compatible with) our contributions. We have created an annotated bibliography of which papers use DTW_I vs. DTW_D (hosted at [19] due to space limitations). As we noted above, there is significant research in "gating networks" and related techniques for choosing which classifier to use on a given region of the input space [18]. However, to the best of our knowledge, these ideas have never been applied to time series, and are only superficially related to the task at hand.

In this work we demonstrate for the first time that of the two obvious ways to do multi-dimensional NN-DTW classification, neither is always superior. We show that the differences are not trivial, as the wrong choice can *double* the error rate. We introduce a simple algorithm that can pick the method that is most likely to predict the correct class on a case-by-case basis. Our algorithm is simple to implement and its overhead is inconsequential in terms of both time and space.

For concreteness we have confined our remarks and empirical demonstrations to *classification* problems, but note distance measures are at the heart of many time series data mining tasks, including clustering, summarization, motif discovery, and many forms of anomaly detection.

ACKNOWLEDGMENTS

This research was supported by NSF IIS-1161997 and NIH R01 DC013547.

7. REFERENCES

[1] I. Assent, M. Wichterich, R. Krieger, H. Kremer, T. Seidl: *Anticipatory DTW for Efficient Similarity Search in Time Series Databases*. PVLDB 2(1): 826-837 (2009).

[2] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. 2008. *Querying and mining of time series data: experimental comparison of representations and distance measures*. PVLDB 1, 2, 1542-52.

[3] Field Guide to the Songbirds of South America The Passerines - Mildred Wyatt-Wold Series in Ornithology(2009) Robert S. Ridgely, Guy Tudor. University of Texas Press.

[4] B. Hu, Y. Chen, J. Zakaria, L. Ulanova, E. Keogh: *Classification of Multi-dimensional Streaming Time Series by Weighting Each Classifier's Track Record*. ICDM 2013: 281-290

[5] J. Kela, P. Korpipaa, J. Mantyjarvi, S. Kallio, G. Savino, L. Jozzo and D. Marca. *Acceleromter-based gesture control for a design environment*, Personal and Ubiquitous Computing, vol. 10, no. 5, pp. 285-299, 2006.

[6] E. Keogh and S. Kasetty, *On the need for Time Series Data Mining Benchmarks: a survey and empirical demonstration*, Proc. ACM KDD 2002, pp. 102-111.

[7] M. H. Ko, G. West, S. Venkatesh and M. Kumar. *Online context recognition in multisensory system using dynamic time warping*. In Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005.

[8] J. B. Kruskall, and M. Liberman, *The symmetric time warping algorithm: From continous to discrete. In Time Warps, String Edits and Macromolecules*. Addison-Wesley, 1983.

[9] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya and V. Vasudevan, *uWave: Accelerometer-based personalized gesture recognition and its applications*, IEEE intl. Conf. Pervasive Computing and Communication (PerCom), 2009.

[10] P. Papapetrou, V. Athitsos, M. Potamias, G. Kollios, and D. Gunopulos. *Embedding-based subsequence matching in time-series databases*. ACM TODS 36, 3, 17, 2011.

[11] F. Petitjean, J. Inglada and P. Gancarski. *Satellite Image Time Series Analysis under Time Warping*. IEEE Transactions on Geoscience and Remote Sensing, vol. 50-. 8, pp. 3081-95, 2012.

[12] J. R. Quinlan, (1986). *Induction of Decision Trees*. Machine Learning 1: 81-106, Kluwer Academic Publishers.

[13] L. Rabiner, and B. Juang, *Fundamentals of speech recognition*. Englewood Cliffs, N.J, Prentice Hall, 1993.

[14] T. Rakthanmanon et al., "*Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping*," SIGKDD 2012.

[15] Y. Sakurai, C. Faloutsos, and M. Yamamuro. 2007. *Stream monitoring under the time warping distance*. ICDE, 1046-55.

[16] J. Wang, A. et al. 2013). *Word recognition from continuous articulatory movement time-series data using symbolic representations*, ACL/ISCA Interspeech Workshop, Grenoble, France, 119-127.

[17] L. Ye and E. J. Keogh. *Time Series Shapelets:A New Primitive for Data Mining*. KDD 2009.

[18] S. E. Yuksel , J. N. Wilson and P. D. Gader. *Twenty years of mixture of experts*, IEEETrans. Neural Netw. Learn. Syst., vol. 23, no. 8, pp.1177 -1193 2012.

[19] Project webpage: https://sites.google.com/site/dtwAdaptive