# DTW-D: Time Series Semi-Supervised Learning from a Single Example

Yanping Chen Bing Hu Eamonn Keogh Gustavo E.A.P.A Batista[1]
University of California, Riverside [1]Universidade de São Paulo-USP
{ychen053, bhu002, eamonn}@cs.ucr.edu gbatista@icmc.usp.br

## ABSTRACT

Classification of time series data is an important problem with applications in virtually every scientific endeavor. The large research community working on time series classification has typically used the UCR Archive to test their algorithms. In this work we argue that the availability of this resource has isolated much of the research community from the following reality, *labeled* time series data is often *very* difficult to obtain.
The obvious solution to this problem is the application of semi-supervised learning; however, as we shall show, direct applications of off-the-shelf semi-supervised learning algorithms do not typically work well for time series. In this work we explain *why* semi-supervised learning algorithms typically fail for time series problems, and we introduce a simple but very effective fix. We demonstrate our ideas on diverse real word problems.

## Categories and Subject Descriptors

H.3.3 **[Information Systems]:** Information Search and Retrieval – *Information filtering, Selection process*

## Keywords

Time Series, Semi-Supervised Learning, Classification

## 1. INTRODUCTION

There has been an enormous interest in time series classification in the last two decades [2][6][10]. Two related conclusions have begun to emerge as a consensus in the community. First, while there is a plethora of classification algorithms in the literature, the nearest neighbor algorithm seems particularly suited to the unique structure of time series, and virtually all competitive attempts at time series classification use it [33]. Second, while there is also a surfeit of possible distance measures for time series, Dynamic Time Warping (DTW), a technique from the dawn of computing, is exceptionally difficult to beat [6]. In particular, a recent paper tested the most cited distance measures on 47 different datasets, and no method consistently outperforms DTW. Thus recent papers that claim improvements over DTW must resort to very powerful statistical tests to demonstrate *tiny* improvements in accuracy.

In the last decade, virtually all of the community has used the UCR Archive to test their algorithms [11]. We believe that the availability of this (admittedly very useful) resource has isolated much of the research community from the following reality, *labeled* time series data is often *very* difficult to obtain. For example, in many situations, from medicine [20] to astronomy [22], obtaining labeled data requires the time and attention of a busy domain expert.

The obvious solution to this problem may appear to be the application of semi-supervised learning; however, direct applications of off-the-shelf semi-supervised learning algorithms do not typically work well for time series. In this work we make two related contributions. We explain *why* semi-supervised learning algorithms typically fail for time series problems, and we introduce a simple but very effective fix. While we defer a detailed explanation of both until Section 4, we offer a simple three sentence preview here:

Under certain assumptions, unlabeled members of a circumscribed positive class may be closer to some unlabeled members of a diverse negative class than to the labeled positive data. This is true *even* under DTW. Nevertheless, unlabeled positive data tend to benefit more from using DTW than unlabeled negative examples. The amount of *benefit* from using DTW over Euclidean Distance (ED) is a meta-feature that can be exploited.

We illustrate this in Figure 1 where we show the hierarchical clustering of five objects under various measures. Two of the five objects are randomly chosen examples (red/bold) from class 3 of the Trace dataset [11]. The other three objects are simply random-walk time series (blue/light).

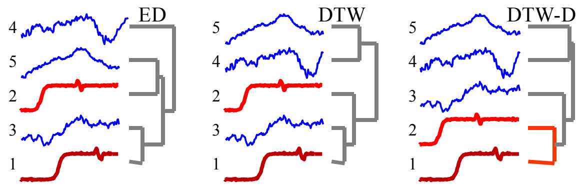

**Figure 1: A complete linkage hierarchical clustering of two items from the Trace dataset with three random walks. From left to right, Euclidean distance (ED), Dynamic Time Warping (DTW) and Dynamic Time Warping-Delta (DTW-D), our proposed technique.**

As we can see, Euclidean Distance does poorly here. This is not surprising, since the Trace dataset is known to have classes that contain exemplars that are time-warped versions of a prototypical shape. Indeed, we see that DTW *does* manage to do better, reducing the distance between Trace-1 and Trace-2. However, this reduction is not enough; random-walk-3 is still closer to Trace-1 than Trace-2 is.

Our key observation is that moving from ED to DTW seems to help the true class data more than the random unstructured data. We can encode this difference/delta that DTW makes with DTW-D, the ratio of DTW over ED. And as we see in Figure 1.*right*, this does produce the correct clustering, at least in this example.

Imagine that we had been doing semi-supervised learning in this dataset using just Trace-1 as our sole positive example. For both ED and DTW, the very first item we added to the positive set would have been a false positive, and it would be very difficult for any algorithm to recover from this. In contrast, DTW-D would have correctly suggested Trace-2 as the next item to add, and assuming only that we had good stopping criterion, we would have done very well.

## 2. RELATED WORK

In the past two decades, there has been an enormous interest in time series classification. Most research efforts leverage off the assumption that there are large amounts of labeled training data [24][29]. In reality, the high cost of labeling the data may render such an assumption invalid. For example, it requires the time and expertise of a cardiologist to annotate individual heartbeats in an ECG data trace [3], but a *single* polysomnography (sleep study) test may produce up to 40,000 such heartbeats. Given that the *acquisition* of unlabeled data is trivial, there is abundance of unlabeled data readily available. For instance, PhysioBank contains over 36,000 recordings of digitized polysomnographic and other physiologic signals, only a tiny fraction of which are labeled [7]; likewise there are tens of millions of books, images, maps and historical manuscripts available on the internet [9], many of which could be fruitfully mined in the time series space [26][32], if only we had more *labeled* data (cf. Section 6.2).

Semi-supervised learning (SSL) is a learning paradigm useful in application domains in which labeled data are limited, but unlabeled data are plentiful [23][8][4]. The literature offers a plethora of SSL methods, among which, *self-training* is perhaps the most commonly-used [17][27][5][34]. *Self-training* is a general framework with very few underlying assumptions. In self-training, a classifier is first trained with a small number of labeled data. It is then used to classify the unlabeled data, and adds the most confidently classified object into the labeled set. The classifier re-trains itself using the new labeled set and the procedure repeated until adding new objects to the labeled set does not increase the accuracy of the classifier or some other stopping criteria is met. This general review of SSL is necessarily brief; we refer the readers to [34] and the references therein for more details.

Recently, some SSL techniques explicitly designed for time series have been proposed. To our best knowledge, thus far there are only three approaches [16][25][19]. The first paper [16] proposed to iteratively expand the labeled set by adding the closest object that is classified as positive to the labeled set. The classifier considers all unlabeled data as negative, and uses the Euclidean Distance. As we shall show, and as has been noted elsewhere [6][33], the inferiority of Euclidean Distance to DTW is mitigated by large training set sizes. Conversely Euclidean Distance is much more brittle a measure than DTW for tiny datasets, which is of course exactly the situation we face here. Thus [25] proposed to build a SSL classifier using DTW distance. Although moving from ED to DTW helps to improve the accuracy of the classifier, the algorithm is still not accurate enough in most real applications (cf. Section 6.1).

More recently, the authors of [19] introduced a SSL technique that interleaves exemplar selection with feature selection, using the work of [16] as a starting point. The method of [19] improves the SSL algorithm, but still uses standard distance measures (Euclidean Distance). As such it is orthogonal to our contribution, which demonstrates that a subtle change in the *distance measure* dwarfs all possible changes in the *algorithms*.

In retrospect, only *three* research efforts on SSL for time series is a surprisingly small number, given that both SSL and time series classification are very popular research topics. In this work we venture to claim that we understand *why* progress in this area has been so slow. In brief, our claim is that there is little utility in tweaking the architecture of the SSL algorithms for time series, as they are all condemned to perform poorly if they use DTW or Euclidean Distance. The contribution of this work is to show a simple but effective fix that *will* allow the existing SSL methods to work very well for time series. It is important to recognize that we are not claiming a contribution to SSL algorithms per se. Rather we will show that changing the distance function used in SSL algorithms can produce a remarkable improvement for time series.

## 3. DEFINITIONS AND NOTATIONS

We begin by introducing all necessary notation and definitions. Although the algorithm presented in this work is applicable to all SSL methods, for ease of exposition, we present just the notations for Positive Unlabeled learning (PU learning), which is a collection of SSL methods that trains a classifier based on the positive (labeled) dataset and the unlabeled dataset only.

**Definition 1:** *P* is the set of training data, including all positively labeled objects.

*P* initially contains only a small number of labeled objects from the positive class, perhaps as few as one. As learning proceeds, the size of *P* increases as some of the previously unlabeled objects in *U* are labeled as positive and moved to *P*. Thus, *P* eventually contains both the original labeled objects, as well as the objects chosen by the classifier from the unlabeled dataset.

**Definition 2:** *U* is the set of unlabeled data.

Objects in dataset *U* can be from the positive class or the negative class. It is generally expected that the vast majority of *U* is from the negative class [34]. The goal of SSL is to map all the objects in *U* to the correct class so that the classifier is accurately trained with the classified objects. We denote individual time series objects from these two sets with subscripts, thus the $i^{th}$ time series object in *P* is denoted $P_i$.

Rather than making a onetime explicit decision as to which objects from *U* should be added to *P*, most algorithms simply iteratively add the next most likely candidate [27][18][34]. This means that we must also specify a *stopping criterion* for the algorithm to predict that it has added all the unlabeled positive objects [34]. The problem of finding a good stopping criteria is an open problem, with tentative solutions based on MDL, Bayesian information criterion, bootstrapping [28][34], etc. As this issue is somewhat orthogonal to our contribution, we simply gloss over it here. However, note that as we shall show in the empirical section, the difference our algorithm makes completely dwarfs any considerations of the optimal stopping criteria. That is to say, even if we did a post-hoc discovery of the optimal stopping criteria for the state-for-art rival, our method would have much higher accuracy for a huge range of "sub-optimal" stopping values.

For brevity, we do not explicitly define *time series*, *Euclidean distance* or *Dynamic Time Warping*, which in any case are rather well known. Instead we use the notation from [6], a heavily cited survey paper on these topics. We do note however the following useful fact, that the ED is an upper bound to the DTW. That is to say, for all $x$, $y$, we have $\mathrm{DTW}(x,y) \leq \mathrm{ED}(x,y)$.

## 4. DTW-D

To explain our observations and our key insight, we consider a concrete example. Let us imagine that we have target class of objects that are defined by having three periods of a sine wave. The instances may be corrupted by warping, different dampening rates, noise, minor changes to the starting phases etc, but as shown in Figure 2 they are unambiguously recognizable to the human eye.

In this example the negative class consists of just a constant line with the same mean as the positive class[1], corrupted by some noise.

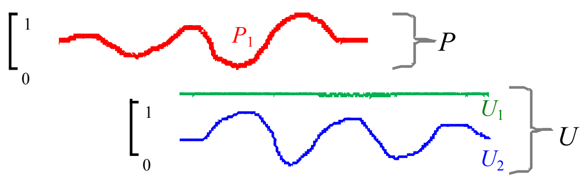

**Figure 2: *top*) A labeled dataset *P* that consists of a single object $P_1$. *bottom*) The unlabeled dataset consist of a single true negative $U_1$ and a single true positive $U_2$.**

Suppose we ask any SSL algorithm to choose one object from *U* to add to *P* using the Euclidean distance. As we can see in Figure 3, $U_1$ is much closer to $P_1$ than $U_2$ is, thus our SSL algorithm would do poorly here.

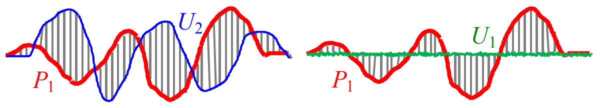

**Figure 3 : The Euclidean distance between two time series is proportional to the length of the gray hatch lines. It is easy to see that $ED(P_1,U_2) > ED(P_1,U_1)$ .**

In retrospect this is not surprising. The brittleness of Euclidean distance to even small amounts of warping is well known, and explains the ubiquity of DTW in most research efforts [13][6][26]. By finding the optimal "peak-to-peak/valley-to-valley" alignment between two time series *before* calculating the distances, DTW is largely invariant to warping, as shown in Figure 4.

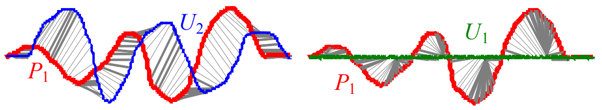

**Figure 4: The DTW distance between two time series is proportional to the y-axis length of the gray lines.**

Unfortunately, while DTW helps significantly, as shown in Table 1, it is not enough: $U_1$ is *still* closer to $P_1$ than $U_2$ is, and the SSL algorithm would *still* pick the wrong object to move from *U* to *P*. Why did DTW not solve our problem? While DTW *is* invariant to warping, there are other differences between $P_1$ and $U_2$, including the fact that the first and last peaks have different heights. DTW cannot mitigate this.

**Table 1: The Distance Matrices for the Three Objects in [*P*,*U*], under both the ED and DTW Distances**

| | ED | | | DTW | | |
|---|---|---|---|---|---|---|
| | $P_1$ | $U_1$ | $U_2$ | $P_1$ | $U_1$ | $U_2$ |
| $P_1$ | 0 | 6.2 | 11 | 0 | 5.8 | 6.1 |
| $U_1$ | | 0 | 6.8 | | 0 | 6.5 |
| $U_2$ | | | 0 | | | 0 |

Moreover, the problem is compounded by the fact that $U_1$ is a "simple" shape. As pointed out in a recent paper, simple shapes tend to be "close to everything" [2]. Figure 3 gives a hint as to why this is true. The flat shape of $U_1$ means that no part of it is more than 0.5 away from any part of $P_1$. In contrast where $P_1$ and $U_2$ are out of phase, and the Euclidean distance is forced to match a peak to a valley, the distance can be as much as 0.8. This is why smooth, flat or least very slowly changing time series tend to be (subjectively) surprisingly close to other objects [2]. This is a grave disappointment – this seems to be a dataset for which DTW is ideally suited, yet DTW fails here.

However, an examination of distance matrices shown in Table 1 *does* reveal an interesting fact. Moving from ED to DTW barely changed the distance between $P_1$ and $U_1$, but it did greatly affect the distance between $P_1$ and $U_2$. We can codify this with the following observation:

**Observation 1:** If a class is characterized by the existence of intra-class warping (possibly among other distortions [2]), then we should expect that moving from ED to DTW reduces distances more for intra-class comparisons than interclass comparisons.

To see this more clearly, we can consider the ratio of distance under DTW and ED as shown in Table 2.

**Table 2: The Ratio of the ED and DTW Distances Shown in Table 1. This ratio is called DTW-D**

| | DTW/ED | | | DTW-D = DTW/ED | | |
|---|---|---|---|---|---|---|
| | $P_1$ | $U_1$ | $U_2$ | $P_1$ | $U_1$ | $U_2$ |
| $P_1$ | 0 | 5.8/6.2 | 6.1/11 | 0 | 0.93 | 0.55 |
| $U_1$ | | 0 | 6.5/6.8 | | 0 | 0.95 |
| $U_2$ | | | 0 | | | 0 |

Note that if we consider the DTW-D *ratios*, we finally have $P_1$ and $U_2$ appear closer than $P_1$ and $U_1$. Figure 5 visualizes all three distance matrices with a complete linkage clustering.

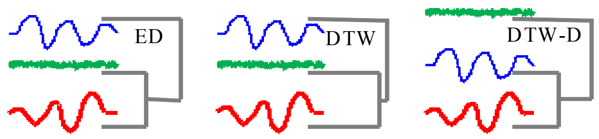

**Figure 5: A visualization of the three distance matrices shown in Table 1 and Table 2 under complete linkage hierarchical clustering.**

Note that there is one minor special case we need to consider. If the ED is zero, then DTW-D would give a *divide-by-zero* error. We simply define this special case as having a value of zero. If the ED distance (and therefore also the DTW distance) between two objects is zero, it would be perverse to call them anything but the same class. This is a moot point, as we never expect observe perfect duplicates for real-values objects.

We have now concretely seen the problem with using ED/DTW for SSL, and our suggested fix, on a toy problem. However, it is natural to ask when this phenomenon actually occurs in the real-world, and would be amenable to our DTW-D solution. In the next section, we explicitly discuss our assumptions about when our ideas can help.

## 4.1 Two Key Assumptions

We do not claim our ideas will help for all time series problems. In particular, we are making two explicit assumptions which we will enumerate and discuss below. We will later show that these assumptions are very often true in real world domains.

**Assumption 1:** The positive class (the target concept) contains time warped versions of some platonic ideal (some prototypical shape), possibly with other types of noise/distortions.

Note that this assumption was true of our toy example in Figure 1. While all members of the Trace dataset have some noise, as

[1] In Figure 2 the objects are shown to the same scale. They are *not* normalized for visual clarity. However, our analysis can be demonstrated for z-normalization, min/max normalization etc.

shown in Figure 6, the most obvious variability between instances is in the timing of the onset of the "ramp-up" and the "oscillation" patterns. Dynamic time warping is able to compensate for and remove this variability.

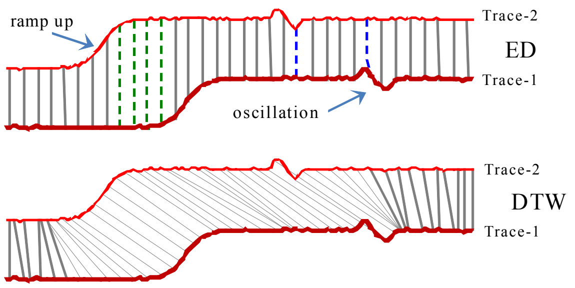

**Figure 6: The two examples from "Trace" shown in Figure 1 compared under ED and DTW. In this plot, the distances are proportional to the variance of the y-axis lengths of the hatch lines. Thus the longer and shorter hatch lines in ED contribute to its large distance, whereas the y-axis lengths for DTW are almost the same, producing a small distance.**

This ability of DTW to compensate for the inherent warping in this class can produce a dramatic difference in classification accuracy. In the UCR Archive dataset the four-class Trace data is provided with a 100/100 train test split. The ED error rate on this dataset is 0.24, whereas DTW has an error-rate of 0.0. Since the exact same splits and classification algorithm (1NN) were used, and zero parameters are tuned for either approach, *all* of this difference can be attributed to the superiority of DTW over ED.

**Assumption 2:** The negative class may be very diverse, and occasionally by chance produces objects close to a member of the positive class, even under DTW.

Empirically, negative classes do tend to be diverse [15][30]. For example, there are only a limited number of ways an audio snippet can sound like a mosquito, but there are infinite ways a sound can be a non-mosquito (c.f. Section 6.1). Once again, this assumption was illustrated by our toy example in Figure 1. The random walk class is naturally very diverse, and it can (and did) produce an instance that is closer to Trace-1 than the other member of the positive class (Trace-2).

It is our central claim that if the two assumptions are true for a given problem, our novel scoring function DTW-D will be better than either ED or DTW. As these are the central assumptions, we will next consider *when* we might expect them to be true.

## 4.2 Observations on our Key Assumptions

In the following sections we consider the implications of our assumptions for the task at hand, and empirically investigate whether these assumptions are warranted.

### *4.2.1 Assumption 1 is mitigated by large amounts of labeled data*

If we have a large enough set of *labeled* examples, we expect that simple DTW or even ED will work very well. Our noted weakness of semi-supervised learning happens when the nearest instance to a labeled positive exemplar is a negative instance. With more labeled positive instances this becomes less and less likely to happen. To see this, we performed an experiment that generalizes the toy example in Figure 1. We created an unlabeled dataset *U* that contains just one exemplar from Class 3 of Trace, and 200 random walks. We then consider the question of what is the probability that the first object added to the labeled dataset *P* is that sole true positive in *U*, for various sizes of *P* from 1 to 10 (i.e. *P* has 1 to 10 true members from Trace). To smooth out our estimate, we averaged over 1,000 runs. As we can see in Figure 7, this probability does indeed increase as |*P*| gets larger.

This plot suggests that if we had a large enough *P*, then DTW-D would offer only a small advantage over ED, and a barely perceptible improvement over DTW. However when *P* is small, DTW-D is dramatically better than both ED/DTW, supporting our assumption.

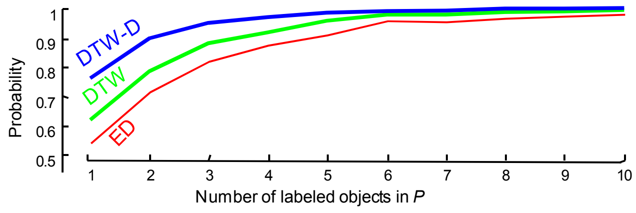

**Figure 7: The probability that the first object added to *P* is a true positive as the number of labeled objects increases, for three distance measures.**

### *4.2.2 Assumption 2 is compounded by a large negative dataset*

In a sense this observation is a direct corollary of the above. If the negative class is random and/or diverse, then the larger the negative class is, the more likely it is that it will produce an instance that just happens to be close to a labeled positive item.

To see this, we again perform an experiment that generalizes our toy example. We created a dataset *P* that contains just one exemplar from Class 3 of Trace. Once again *U* contains a single true positive, but this time we vary the number of random walks from 100 to 1,000. As before we measure the probability that the first object added to *P* is the true positive, averaged over 1,000 runs. Figure 8 shows the results.

In most semi-supervised settings, we expect |*U*| to be many orders of magnitude larger than the |*P*|, thus this assumption is almost always true in real settings.

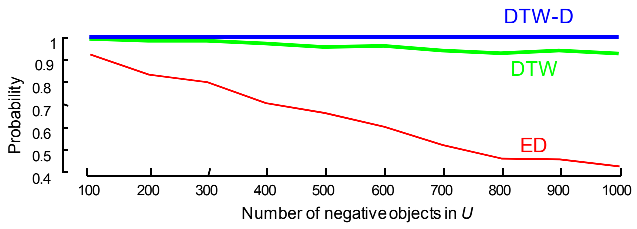

**Figure 8: The probability the first object added is a true positive as the number of true negatives in *U* increases.**

Again this figure strongly supports our assumption. When we have relatively few true negatives, all methods work well. However, as the number of true negatives in *U* increases, ED rapidly deteriorates. In contrast, DTW deteriorates more slowly. Remarkably, however, DTW-D is completely unaffected by a surfeit of true negatives, maintaining perfect accuracy.

### *4.2.3 Assumption 2 is compounded by low complexity negative data*

Our final observation requires us to define what is meant by the "complexity" of a time series. Our remarks here are inspired by [2], which makes a similar observation, but in a very different context. As noted in [2], while a "complex" time series is hard to define, it is something we can intuitively understand. For our purposes, let us say that a complex time series is one that is not well approximated by few DFT coefficients or by a low degree polynomial.

The problem caused by low complexity data is that it is "close to everything", and as such, the chances that at least one instance from the negative class is closer to an exemplar from *P* than a true positive is much greater if some or all the negative data has low complexity.

To see this we can repeat the experiment shown in Figure 1 after replacing the random walk series by randomly generated third-degree polynomials.

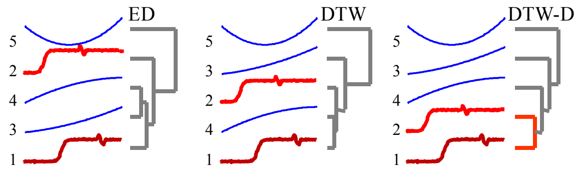

**Figure 9: The experiment shown in Figure 1 after replacing the three random walks with three random third-degree polynomials.**

The results here are visually jarring. It is important to emphasize that this is not the result of an error, contriving of the data, or crippling the ED/DTW in any way. It is simply the case that low complexity time series have a tendency to have a small distance to all other objects.

Apart from [2], other works have indirectly noted this phenomenon. For example, [12] notes that if we average all subsequences in a long time series, we will get a *constant line*, which is surely the *least* complex time series under any definition. Implicitly, this means that a constant line is the time series with minimal *expected distance* to any randomly chosen time series.

Thus, if the negative class is complex, we should expect the DTW or even ED will work well for semi-supervised learning. To see this, we can repeat the experiment shown in Figure 1/Figure 9 after replacing negative class with pure random vectors. Note that while we may consider random vectors as "noisy", it is incidental to the point that they are *complex*.

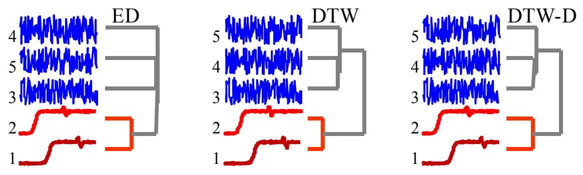

**Figure 10: The experiment shown in Figure 1/Figure 9 after replacing the negative class with random vectors.**

Figure 10 demonstrates that when the unlabeled data are complex, even ED has little trouble in grouping the positive class.

Beyond the visual evidence shown in Figure 9 and Figure 10, we can test our observation with another simple experiment. Once more we perform an experiment that generalizes our toy example. We created a labeled dataset *P* that contains just one exemplar from Class 3 of Trace. This time *U* contains one true positive and 200 random time series that are approximated by *k* non-zero DFT coefficients, with *k* ranging from 5 to 20. Figure 11 shows some examples of time series that are approximated by 5 and 20 non-zero DFT coefficients respectively.

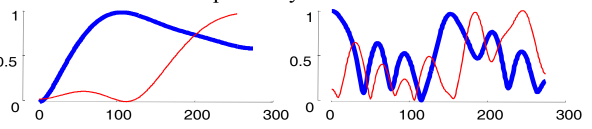

**Figure 11: *left*) Two time series examples that are created using 5 non-zero DFT coefficients. *right*) Two time series examples that are created using 20 non-zero DFT coefficients. Clearly the latter are more complex.**

As before we measure the probability that the first object added to *P* is a true positive, averaged over 1,000 runs. Figure 12 shows the results.

Once again this experiment strongly supports our hypothesis. Low complexity items in the negative class make SSL more difficult for all distance measures, but using DTW-D does greatly mitigate the problem.

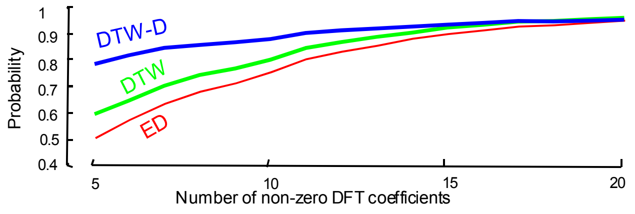

**Figure 12: The probability the first object added is true negative as the negative objects increase in complexity.**

### *4.2.4 Implications of observations/assumptions*

The observations and experiments in the previous sections tell us when we should expect the DTW-D method to help. We should *not* expect DTW-D to help with the classic time series classification problems as exemplified by the UCR archive [11]. These datasets typically *do* have a relatively large set of label positive data (at least dozens), and typically do *not* have one class with much lower complexity and/or much higher diversity than the other classes.

In contrast, it is easy to see that all our assumptions mesh perfectly with most SSL assumptions [28][27][23]:

- We *do* have very small (as few as one) positive examples (cf. Section 4.2.1).
- We *do* have relatively large amounts of negative data. For example, when trying to learn the `vacuum-cleaning` concept. (cf. Section 6.3), we must expect that most people spend less than 0.01% of their time vacuuming. Likewise, if we monitor audio streams, most sounds we hear are *not* insects (cf. Section 6.1) etc.
- We *do* have at least some low complexity negative instances (cf. Section 4.2.3). This is true because the negative class is usually highly variable. For human activity monitoring, there are likely moments when a person is sitting still, producing very low complexity data (cf. Section 6.3)

As we shall show in Section 6, SSL problems in very diverse domains fit into our assumptions.

## 5. ALGORITHM DETAILS

While we believe that our ideas can be applied to essentially any time series SSL learning framework, simply by replacing the ED or DTW distance calculations with DTW-D. However, for concreteness, in this section we will explicitly define the exact SSL algorithm used in our experiments. Note that we took pains to choose a simple SSL algorithm here, because as we noted before, we are not claiming a contribution to SSL per se. Rather, our contribution is a simple but effective fix to a problem that will otherwise plague any attempt at SSL for time series. Our focus in this work is to demonstrate the effectiveness of our ideas, even with a simple SSL algorithm.

Note that we are assuming that whatever "flavor" of SSL is, the underlying classification algorithm will use Nearest Neighbor (NN) [6]. This is because, in spite of two decades of experimentation with neural networks, decision trees, Bayesian methods [29] etc., there is strong empirical evidence that nearest neighbor algorithms are the best approach for time series (see [33], and the references therein and thereof).

## 5.1 DTW-D Algorithm

As hinted in Section 4, our proposed distance measure DTW-D is accomplished with a simple equation:

$$\text{DTW-D}(x,y) = \frac{\text{DTW}(x,y)}{\text{ED}(x,y) + \epsilon} \tag{1}$$

Where $\epsilon$ is an extremely small positive quantity used to avoid divide-by-zero error. We reiterate that $\epsilon$ is *not* parameter of our system, it is device to enable a terser definition.

As shown in Table 3, the computation of DTW-D can be achieved on two series $x$ and $y$ using one line of matlab code:

**Table 3: Our Proposed Distance Measure**

```
function      distance = DTW-D(x, y)
     distance = DTW(x,y) / (ED(x,y) + eps);
```

## 5.2 Training the Classifier

The classifier used is a one-class classifier [30]. The training dataset contains *only* the objects from the positive class. The goal of the classifier is to accurately extract all the positive class objects from the unlabeled dataset [15].

The data used to train the classifier includes a labeled dataset *P* and an unlabeled dataset *U*. In the beginning, there is as few as one labeled object in *P*. As shown in Table 4, the classifier trains itself through the following steps:

Step 1: The classifier is trained on the initial labeled dataset, which contains as few as only one object from the positive class. Note that the labeled dataset is augmented gradually during the training process.

Step 2: For each object in the unlabeled dataset *U*, we compute its distance to the labeled dataset using DTW-D (Line 10 to Line 13). An object's distance to the labeled dataset is the distance of the object to its nearest neighbor in the labeled dataset.

Step 3: Among all the unlabeled objects, the one we can most confidently classify as positive is the object that is closest to the labeled dataset (Line 3). The object is added into the labeled dataset (Line 4) and removed from the unlabeled dataset (Line 5). With the labeled dataset being adjusted, we return to Step 1 to re-train the classifier. The procedure repeated until some stopping criterion is met.

The intuition behind the algorithm is straightforward. The labeled dataset defines the concept space for the positive objects. The object closest to the labeled dataset is deemed to have the highest probability to belong to the positive class.

**Table 4: Time Series Semi-supervised Learning Algorithm**

```
Function P = Train_Classifier (P, U, distance, N)
Input:    P, the initial training dataset with a single training object
          U, the unlabeled dataset
          distance, the distance function
          N, the number of objects to be moved from U to P
Output:   a trained classifier (for a NN classifier, it is the learned P)

1    function P = Train_Classifier ( P, U, distance, N)
2       for iterations = 1 : N
3            NNObject = findUnlabeledNN( P, U, distance);
4            P = [P, NNObject];      // update P
5            U = U - NNObject ;      // update U
6       end
7       return P;
8    end
9    function NNObject = findUnlabeledNN ( P, U, distance)
10      Dist = zeros(1, |U|);
11      for i = 1 : |U|
12         Dist (i) = min_{j = 1,...|P|} distance(U_i,P_j);
13      end
14      [~, NN_index] = min(Dist);   // closest to P
15      NNObject = U_{NN_index} ;
16      return NNObject;
17   end
```

At the first blush, our algorithm to train the classifier seems quite similar to the algorithm used in [16]. However, a more careful introspection would reveal that they are fundamentally different. The classifier used in [16] is a binary classifier, with all the unlabeled objects regarded as training examples from the negative class, whereas our classifier is a one-class classifier with no training examples from the negative class. The advantage of our classifier is that it makes much more realistic assumptions about how SSL work in practice. As noted elsewhere, the negative class is typically not a single well-defined concept as [16] and others assume, rather it tends to be an *extremely* heterogeneous class. To give an example in a domain we consider, there are very limited ways humans can perform `vacuum cleaning`, but there are an infinite number of possible human activities that are not `vacuum cleaning`.

As noted above, in this work we gloss over the orthogonal problem of finding a good stopping criterion. In our experimental section, the training process stops when the unlabeled dataset *U* is exhausted of true positives by DTW-D.

## 5.3 Evaluating the Classifier

To evaluate the accuracy of all classifiers, we test the classifier using data that is "hidden" during the training stage. The test (holdout) dataset contains some positive class objects and many other objects. The goal of the classifier is to accurately extract all the positive class objects from the test dataset. We use the classic notion of *precision* and *recall* [31] to measure the performance of the classifier. If an instance in the test dataset is top *K* closest to the labeled dataset, the instance is classified as positive, otherwise it is negative. *K* is the number of positive objects in the test dataset. Thus we can count the number of true positives out of *K* classifications.

Note that with this evaluation method, the value of *recall* equals to the value of *precision* (because the number of false negatives is the same as the number of false positives). For brevity, we report only the *precision* here. The computation of *precision* is shown in Equation (2), where $N_{\text{positive}}$ denotes the number of true positives among the top *K* closest instances.

$$precision = \frac{N_{positive}}{K} \tag{2}$$

## 6. EXPERIMENTAL EVALUATION

We begin by noting that *all* experiments (including *all* the figures above) are completely reproducible. All experimental code and data (and additional experiments omitted for brevity) are archived in perpetuity at [36].

For all experiments, we divide the data into two mutually exclusive datasets: the learning dataset and the holdout dataset.

- **Learning dataset:** The *learning dataset* is used in the SSL process to train the classifier. It is divided into the labeled dataset *P* and the unlabeled dataset *U*. The labeled dataset includes a *single* positive example, which is a randomly selected true positive object from the learning dataset. The rest of objects in the learning dataset are regarded as unlabeled objects and are included in *U*.
- **Holdout dataset:** The *holdout dataset* is used to test the accuracy of the learned classifier. Objects in the holdout dataset are hidden from the SSL process.

The performance of the trained classifier can be sensitive to the initial training (labeled) example. To mitigate this sensitivity, for each experiment, we repeat the training process by each time starting from a different training example. In particular, we allow each positive object in the learning dataset to be used as the initial training example once, and average the accuracy of the classifier over all runs.

To show the changes in the performance of the classifier as the labeled dataset *P* is gradually augmented, we show the average accuracy for each size of *P*. That is, we evaluate the classifier using the holdout dataset each time an unlabeled object is added to *P*. Thus all figures shown below show the *holdout* accuracy.

For each experiment, we compare the performance of three different classifiers, the classifier using ED, the classifier using DTW, and the classifier using DTW-D. All three classifiers are trained using the same SSL algorithm as shown in Table 4. The only difference among them is the distance function used. As we shall show, by simply changing the distance function from ED or DTW to DTW-D, we can improve the performance of SSL algorithms for time series by a significant amount.

In all our experiments, DTW-D learns from a *single* positive example. There is nothing about our technique that requires this. We simply wish to show we can learn under the most hostile assumptions.

We also compare our idea with rival time series SSL approaches [16][25]. In order to be scrupulously fair to the rival approaches, we allow them to "cheat" by starting with more training examples. As we shall show, even given this severe disadvantage, our algorithm still significantly outperforms the rival approaches.

## 6.1 Insect Wingbeat Sound Detection

In this experiment, we would like to detect insect wing-beat sounds from unstructured audio streams. The insect used in this experiment is *Culex quinquefasciatus* female. The wingbeat sounds are acquired using the sensors described in [1], and the data stream also contains diverse negative data, including speech/music etc.

For this experiment, we randomly select 1,000 insect sounds and 4,200 non-insect segments. The length of each sound snippet is 0.1 second. All the sound data are first converted into "time series" using DFT [1]. Based on entomological advice, we preserve only the coefficients corresponding to the frequency range between 200 and 2,000, because all other coefficients are unlikely to be the result of insect activity.

We divide the time series dataset into two parts: a learning dataset with 500 insect sounds and 2000 non-insect sounds, and a holdout dataset with 500 insect sounds and 2,200 radio sounds.

The SSL process is repeated 500 times, each time starting with a different training example. For each run, we trained three classifiers, the NN classifier using ED distance, the NN classifier using DTW and the NN classifier using DTW-D. The average performance of the three classifiers over 500 runs for each size of *P* is shown in Figure 13.

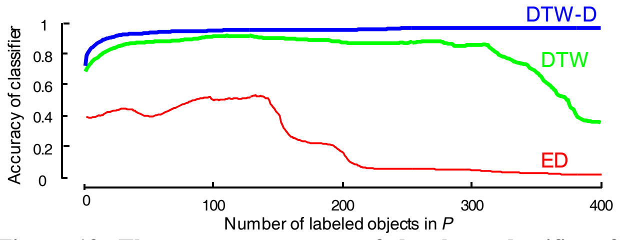

**Figure 13: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset.**

The results are impressive, given that the default accuracy is just 20%. With DTW-D, we can converge on greater than 95% accuracy. The DTW-D classifier consistently outperforms the other two classifiers across the entire range of values for *P*. Note that, after the size of *P* reaches 150, the accuracy of the ED classifier begins to decrease. As we might expect, the DTW classifier's invariance to warping allows it *both* to start from a higher baseline, *and* keep improving for a longer time. However it too is doomed to eventually add true negatives into *P* and begin a rapid decrease in performance.

### 6.1.1 Why is DTW-D better?

While DTW-D is clearly better than its rivals, Figure 13 does not tell us *why*. In particular we may ask if it is because: DTW-D generally selects better labeled objects during the SSL process, *or* because DTW-D selects better top *K* nearest neighbors from the holdout dataset in the classifier's evaluation process (recall that *K* is the number of true positives in the holdout dataset).

To answer this question, we conducted a combinatorial experiment in which we crippled DTW-D independently in each phase (training/evaluating).

For example, to see if DTW-D selects better labeled objects than DTW, we train two NN classifiers, one using DTW-D and one using DTW. We then evaluate both classifiers using the same holdout dataset. In the evaluation process, we use the same distance function (DTW) to find the top *K* nearest neighbors for both classifiers. In this way, we ensure that the *only* difference between the two classifiers is the use of two different distance functions in the training process.

The experiment to see if DTW-D is better at selecting the top *K* nearest neighbors during evaluation process is similar. This time, we train only one classifier, the NN classifier using DTW. In the evaluating process, we use two different distance functions, DTW and DTW-D, to find the top *K* nearest neighbors for this classifier. In this experiment, the labeled dataset learned from the training process is the same. The *only* difference is the use of different distance functions in the evaluation process.

Figure 14 shows the results of the combinatorial experiment for this dataset.

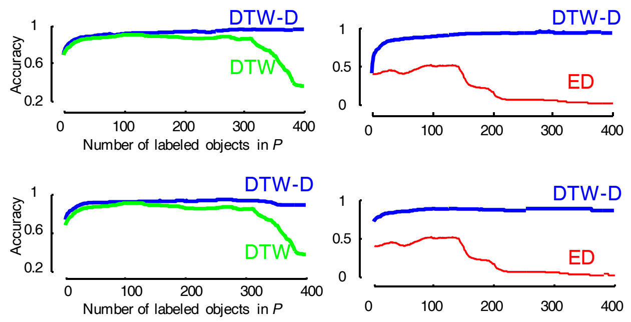

**Figure 14: *top)* Comparison of DTW-D with DTW / DTW-D with ED, to see if DTW-D helps the training process by selecting better exemplars. *bottom*) Comparison of DTW-D with DTW / DTW-D with ED, to see if DTW-D helps the evaluating process by selecting better top *K* nearest neighbors.**

The results show that DTW-D is superior to ED and DTW in both the training (exemplar *choosing*) and evaluation (the later *classification*) processes. The results shown in Figure 14 are generally true for all the experiments done in this work. For brevity, we omit these results for the following applications, but archive them in [36] for interested readers.

### 6.1.2 Comparison to rival methods

We compare our algorithm with the widely-used rival approaches that are specially designed for time series [16][25]. In the comparison, we favor our rival approaches by offering them with fifty more initial labeled examples. That is, through the entire range of comparison, our rivals always have fifty more labeled objects in their labeled dataset than our method. Recall that our algorithm starts with a *single* labeled example, thus the rival methods have a fiftyfold advantage here. Figure 15 shows the results for this dataset.

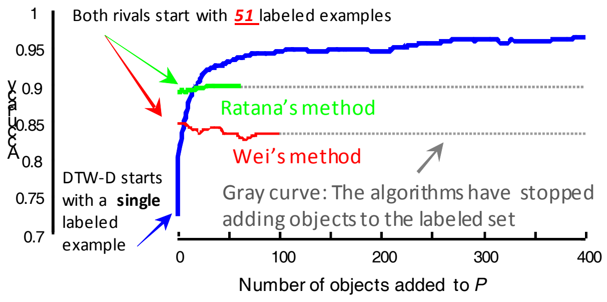

**Figure 15: Comparison of our SSL method using DTW-D with two rival methods in the literature. The curves show the average performance of classifiers over 100 runs**

As we can see, our method was not as good as the rivals at the beginning. This is hardly surprise given that the rival methods had fifty times as many labeled objects. However, using DTW-D, our method intelligently selects objects from the unlabeled dataset *U* to expand the labeled set *P*. After adding just nine objects, we beat Wei's method. This is very impressive because this happened when we have only ten labeled objects while our rivals have sixty. In other words, we evaluate our classifier with the holdout dataset with only ten labeled objects while our rivals have sixty labeled objects. We are doing better here because while both methods have added nine objects, DTW-D made better choices of what to add. Ratana's method is beaten after adding 21 objects.

In addition, as can be seen from Figure 15, our method continues to do better after we beat the rivals, which implies that whatever stopping criterion is used, we will always do better. In this example, Wei's method stops after adding 103 objects. Ratana's method stops after adding 62 objects. No matter if we stop at 62 or 103 or any other position greater than 21, we are always better than the rivals.

It might be imagined that the two rival methods [16][25] do (eventually) make better decisions about which objects to add, but are crippled by too conservative a stopping criteria. However, to be clear, this is not the case. When the two algorithms terminate, they have labeled *everything* in the learning dataset. Thus, there are no actions (wise or unwise) left for them to perform.

In addition to [16][25], there is only one other semi-supervised approach for time series that we are aware of. In [19], the authors introduce a technique that interleaves exemplar selection with feature selection. We do not compare to this work for the following reasons: The work assumes that the positive class objects are similar to each other, *and* the negative class objects are similar to each other. For this reason they test on a subset of the UCR archive datasets, with one class acting as *P* and another class acting as *U*. This is in sharp contrast to our more relaxed assumption that *only* positive class objects are similar to each other. We make no such assumptions about the negative class. Our initial attempts to test their algorithm with our assumptions (the authors generously donated their code), yielded very poor results, but in fairness, the authors made no claims about the utility of their ideas for our problem statement/assumptions.

The comparison results shown in Figure 15 are generally true for all experiments done here. For brevity, we omit these results for the following applications, but archive them in [36] for interested readers.

## 6.2 Historical Manuscript Mining

We can apply "time series" SSL to detect particular image patches in historical manuscripts [32]. These manuscripts often are hand colored over years, thus some warping is needed to detect the similarity of the ("drifting") color distributions, as shown in Figure 16.

The task in this application is to detect examples of the Fugger of the Deer heraldic shield from a huge manuscript [35]. Data used in this experiment includes 67 positive Fugger shields and 828 negative patches selected from the same manuscript [35]. Each image patch is converted to a RGB color histogram, which is normalized using sum-to-one normalization to eliminate the variability of image size. Figure 16.*right* shows two examples of the converted color histograms.

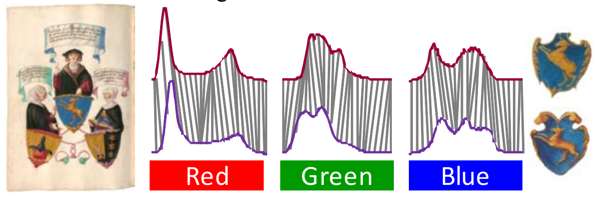

**Figure 16: left) A page from a 16th century text [35] shows three heraldic shields including that of Fugger vom Reh (Fugger of the Deer) granted to Andreas Fugger in 1464. right) Two additional examples of the shield from the same text have been converted to RGB color histograms and compared using DTW.**

Again, we first divide the data into two datasets: a learning dataset with 16 positive objects and 207 negative ones, and a holdout dataset with 51 positive objects and 621 negative ones. We repeat the SSL process 16 times, each time starting from a different training seed. Figure 17 shows the averaged holdout accuracy of the three classifiers for different sizes of *P*.

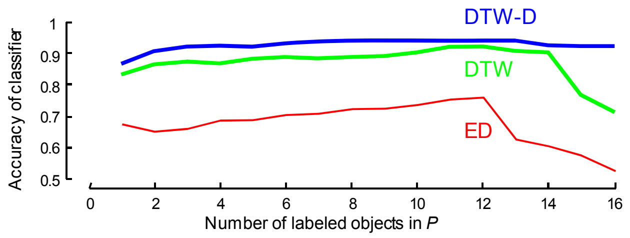

**Figure 17: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset.**

The default holdout accuracy is 51/672, which is about 7.6%. With DTW-D, we can achieve more than 90% accuracy. The performance of the DTW classifier is much better than the ED classifier, which is not surprising since, as we noted before, there is clearly some warping in the color distributions of objects belonging to a same class.

Note that in Figure 17, there is a rapid decrease in the accuracy of the ED classifier when the size of *P* reaches 12. With inspection we find this decrease is caused by the first false positive that is added into *P* at the point (on average). This false positive object in *P* corrupts the concept of the positive class, resulting in more negative objects added to *P,* and thus, decreasing the classifier's accuracy. Although the DTW classifier's invariance to warping enables it to have a higher accuracy as well as to keep improving for a longer time, it too experiences a rapid decrease when the size of *P* reaches 14, where it (on average) mistakenly accepts its first true negative.

## 6.3 Activity Recognition

We finally consider a widely studied benchmark dataset that contains data of 18 different activities, such as `running`, `rope-jumping`, `ironing`, `vacuum-cleaning`, performed by 9 subjects wearing 3 inertial measurement units (IMUs) [21]. We randomly pick one such activity as the positive class and treat the rest as negative.

Figure 18 shows the results for an example experiment where `vacuum cleaning` is considered as the positive activity. We randomly select 400 positive segments and 1,600 negative ones from the dataset, and divide the selected data into two datasets, the learning dataset with 100 positive objects and 400 negative objects, and the holdout dataset with 300 positive objects and 1,200 negatives. The SSL process is repeated 100 times, each time starting from a different training seed. The results show that the DTW-D classifier both starts from a higher baseline and continues to improve over the entire range of values. In contrast, both ED and DTW start from a lower baseline and eventually get worse.

We remind the reader that as with all experiments in this work, the three lines in this figure are based on identical data, identical conditions and identical algorithms. The *only* difference is the distance measure used, thus we can safely attribute *all* improvement observed to DTW-D.

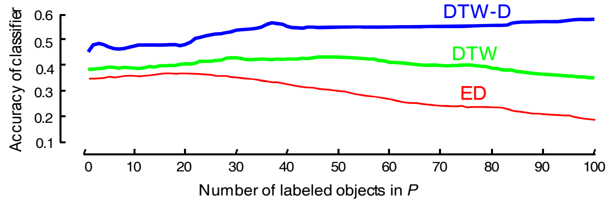

**Figure 18: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset.**

## 7. CONCLUSION AND FUTURE WORK

We have introduced a simple idea that dramatically improves the quality of SSL in time series domains. We have conducted our experiments such that all improvements observed can be *only* attributed to the use of DTW-D.

Our work has the following advantages: It is completely parameter-free, and thus requires no tuning/tweaking. It allows the use of existing state-of-the-art indexing methods and fast similarity search methods [6]. The time and space overhead are inconsequential, as is the coding effort; requiring only a single line of code to be changed. While we choose the simplest SSL method to demonstrate our ideas, they can trivially be used with any SSL algorithm.

Future work includes revisiting the stopping criteria issue in light of DTW-D, and considering other avenues where DTW-D may be useful.

## ACKNOWLEDGEMENTS

We gratefully acknowledge funding from NSF IIS-1161997, Vodafone, FAPESP-2012/07295-3, and all the data donors.

## 8. REFERENCES

[1] G. Batista, E. J. Keogh, A. Mafra-Neto, E. Rowton, SIGKDD demo: Sensors and Software to allow Computational Entomology, an Emerging Application of Data Mining. *KDD*'11: 761-764, 2011.

[2] G. Batista, X. Wang, E. J. Keogh, A Complexity-Invariant Distance Measure for Time Series, *SDM*'11: 699-710, 2011.

[3] P. Chazal, M. O'Dwyer, R.B. Reilly, Automatic classification of heartbeats using ECG morphology and heartbeat interval features, *IEEE Trans Biomed Eng*, 51: 1196–1206, 2004

[4] S. Cheng, Y. Shi, Q.Qin, Particle swarm optimization based semi-supervised learning on Chinese text categorization, *CEC*: 1-8, 2012

[5] M. David, C. Eugene, Johnson. Mark, Effective Self-Training for Parsing, *HLT-NAACL*: 152-159, 2006

[6] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, *PVLDB* 1(2): 1542-1552, 2008

[7] A. Goldberger, et al. PhysioBank, PhysioToolkit, and PhysioNet: *New Research Resource for Complex Physiologic Signals, Circulation* 101(23): 2000

[8] M. Guillaumin, J. Verbeek, C. Schmid, Multimodal semi-supervised learning for image classification, *CVPR*: 902-909, 2010

[9] M. Herwig, Google's Total Library: Putting the World's Books on the Web, 2007

[10] B. Hu, Y. Chen and E. Keogh, Time Series Classification under More Realistic Assumption, *SDM*, 2013

[11] E. J. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, C. A. Ratanamahatana, C. A. (2011). *The UCR Time Series Classification/Clustering Homepage*

[12] E. J. Keogh, J. Lin, Clustering of time-series subsequences is meaningless: implications for previous and future research. *KAIS* 8(2): 154-77, 2005

[13] E. J. Keogh, W. Li, X. Xi, S. Lee, M. Vlachos, LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures, *VLDB* : 882–893, 2006

[14] S. Lenser, M. Veloso, Non-Parametric Time Series Classification, *ICRA*: 3918-3923, 2005

[15] X. Li, B. Liu Learning to classify text using positive and unlabeled data, *IJCAI*: 587-594, 2003

[16] W. Li, E. Keogh, Semi-supervised time series classification, *ACM SIGKDD:* 2006

[17] A. Lomsadze, et al. Gene identification in novel eukaryotic genomes by self-training algorithm, *Nucleic Acids Res.* 33: 6494–6506, 2005

[18] U. Maulik, D. Chakraborty, A self-trained ensemble with semi-supervised SVM: An application to pixel classification of remote sensing imagery, *Pattern Recognition* 44(3):615-623, 2011

[19] M. N. Nguyen, X. Li, S. Ng, Positive unlabeled learning for time series classification, *IJCAI* (2) , 2011

[20] P. Ordóñez, et al., Visualization of Multivariate Time Series Data in a Neonatal ICU, *IBM Journal of Research and Development*, 2012

[21] PAMAP, Physical Activity Monitoring for Aging People, www.pamap.org/demo.html , retrieved 2012-05-12.

[22] D. Preston, P. Protopapas, C. E. Brodley, Discovering arbitrary event types in time series. *Statistical Analysis and Data Mining 2(5-6)*: 396-411, 2009

[23] M. A. Ranzato, M. Szummer, Semi-supervised learning of compact document representations with deep networks, *ICML*: 792-799, 2008

[24] M. Raptis, K. Wnuk, S. Soatto, Flexible Dictionaries for Action Recognition, *MLVMA/ECCV*, 2008

[25] C. A. Ratanamahatana., D. Wanichsan, Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification. SNPD 2012. 149: 1-14, 2008.

[26] T. M. Rath, R. Manmatha, Word Image Matching Using Dynamic Time Warping, *CVPR* (2): 521-527, 2003

[27] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-Supervised Self-Training of Object Detection Models, *WACV*: 29-36, 2005

[28] A. Sun, R. Grishman, Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters, *COLING (Posters)*: 1194-1202, 2010

[29] P. Sykacek, S.J. Roberts, Bayesian time series classification, *NIPS* : 937-944, 2001

[30] D.M.J. Tax, One-class classification: Concept-learning in the absence of counter-examples. 2001

[31] C. J. Van Rijsbergen, Information Retrieval, 2nd edition, London, England: Butterworths, 1979

[32] X. Wang, L.Ye, E. J. Keogh, C. R. Shelton, Annotating Historical Archives of Images, *IJDLS* 1(2): 59-80, 2010

[33] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, C. A. Ratanamahatana, Fast time series classification using numerosity reduction, *ICML*: 1033-40, 2006

[34] X. Zhu, Semi-Supervised Learning Literature Survey, Technical report, no. 1530, *Computer Sciences*, University of Wisconsin-Madison, 2005

[35] Das Ehrenbuch der Fugger (The secret book of honour of the Fugger) -BSB Cgm 9460, *Augsburg, ca. 1545 - 1548 mit Nachträgen aus späterer Zeit*

[36] Supporting webpage: *https://sites.google.com/site/yanpingdtwd/*