Open access • Proceedings Article • DOI:10.1145/1150402.1150498

# Semi-supervised time series classification — Source link

Li Wei, Eamonn Keogh

**Institutions:** University of California, Riverside

**Published on:** 20 Aug 2006 - Knowledge Discovery and Data Mining

**Topics:** Semi-supervised learning

# Semi-Supervised Time Series Classification

Li Wei Eamonn Keogh
Department of Computer Science and Engineering
University of California, Riverside
{wli, eamonn}@cs.ucr.edu

## Abstract

The problem of time series classification has attracted great interest in the last decade. However current research assumes the existence of large amounts of *labeled* training data. In reality, such data may be very difficult or expensive to obtain. For example, it may require the time and expertise of cardiologists, space launch technicians, or other domain specialists. As in many other domains, there are often copious amounts of unlabeled data available. For example, the PhysioBank archive contains gigabytes of ECG data. In this work we propose a semi-supervised technique for building time series classifiers. While such algorithms are well known in text domains, we will show that special considerations must be made to make them both efficient and effective for the time series domain. We evaluate our work with a comprehensive set of experiments on diverse data sources including electrocardiograms, handwritten documents, manufacturing, and video datasets. The experimental results demonstrate that our approach requires only a handful of labeled examples to construct accurate classifiers.

## Keywords

Semi-supervised Learning, Time Series, Classification, Data Mining

## 1 INTRODUCTION

Time series data are ubiquitous and are of interest to many communities. Such data can be found in virtually all avenues of human endeavor including medicine, aerospace, finance, business, meteorology, and entertainment [18][26][36][41]. The problem of time series classification has been the subject of active research for decades [11][12][14][22][24][30]. However current methods are limited by the need for large amounts of labeled training data. In reality, such data may be very difficult or expensive to collect. For example, it may require the time and expertise of cardiologists [18], space launch technicians [26], entomologists [41], or other domain experts to manually label the data.

As in many other applications, copious amounts of unlabeled data are often readily available. For example, the PhysioBank archive [18] contains more than 40 gigabytes of ECG data freely available over the web, and hospitals often archive even larger amounts of ECG data for legal reasons. Recent advances in sensor technology have made it possible to collect enormous amounts of data in real time.

In this work we propose a semi-supervised technique for building time series classifiers that takes advantage of the large collections of unlabeled data. As we will demonstrate, our approach requires only a handful of labeled examples to construct accurate classifiers. Furthermore, we are able to leverage off recent advances in time series query filtering to use these classifiers very efficiently, particularly for streaming problems [41].

To enhance the readers' appreciation of the diversity of domains which can benefit from a semi-supervised technique for building time series classifiers, we begin by considering some applications that we will later address experimentally.

**Indexing of handwritten documents:** There has been a recent explosion of interest in indexing handwritten documents [28], driven in large part by Google and Yahoo's stated interest of making large archives of handwritten text searchable [27]. It has recently been shown that simply treating the words as "time series" (see Figure 1) is an extremely competitive approach [28] for classifying (and thus indexing) handwritten documents.

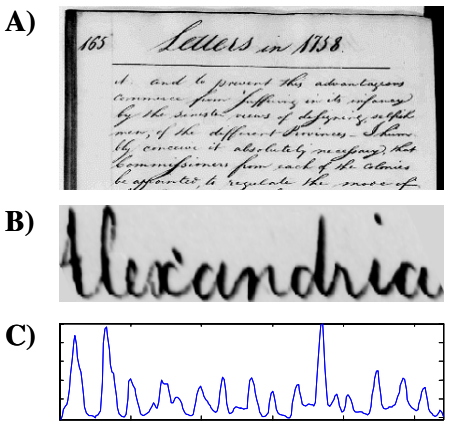


Figure 1: **A)** A sample of text written by George Washington. **B)** The word "Alexandria" after having its slant removed. **C)** A time series created by tracing the upper profile of the word (Image courtesy of Raghavan Manmatha, used with permission)

The fundamental problem in creating highly accurate handwriting classifiers is that they must be trained on each individual's particular handwriting; a classifier built for George Washington will not generalize to Isaac Newton. However the cost of obtaining labeled data for each word, for every individual is very expensive as measured in human time. A semi-supervised approach where a user annotates just a few training examples would have great utility [28].

**Heartbeat Classification:** As noted earlier, the PhysioBank archive [18] contains more than 40 gigabytes of freely available medical data, including EEG, gait, and ECG data. Such large datasets are potential goldmines for a researcher wishing to build a classifier. However, only a tiny subset of this data has been annotated. Furthermore, as with handwriting, some level of personalization can be useful here. Once again, a semi-

supervised approach where a cardiologist annotates just a few training examples, could be of great utility [41].

The rest of this paper is organized as follows. In Section 2 we review background material. We introduce our semi-supervised time series classification algorithm in Section 3. Section 4 sees a comprehensive empirical evaluation. Finally in Section 5 we offer some conclusions and directions for future work.

## 2 BACKGROUND MATERIAL

In order to frame our contribution in the proper context, we begin with a review of the necessary background material.

### 2.1 Value of Unlabeled Data

The idea of using unlabeled data to help classification may sound initially unintuitive. However, several studies in the literature have indicated the utility of unlabeled data for classification [13]. For example, early studies [17][19][34] asserted that unlabeled data should be used whenever available. Castelli [8] and Ratsaby et. al. [37] showed that "*unlabeled data are always asymptotically useful for classification*".

Although unlabeled data *alone* are generally insufficient to yield better-than-random-guess classification, they *do* contain information which can help classification. We can see this with the simple contrived example in Figure 2 (the reader may find it useful to look at Figure 12 to see why this is a "time series" problem). Here we have a dataset of just three labeled instances, although eight unlabeled instances (U) also exist. We need to classify the instance marked with "**?**", which clearly belongs to the F (female) class. However this particular image happens to show the actor in a pose which is very similar to one of the M (male) instances, $M_1$, and is thus misclassified[1].

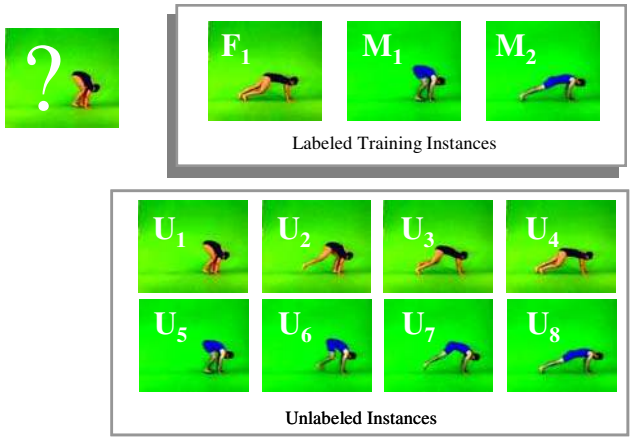


Figure 2: A simple example to motivate semi-supervised classification. The instance to be classified (marked with "?") is actually a F (female) but happens to be closer to a M (male) in this small dataset of labeled instances

Note that while $F_1$ happens not to be a close match to the instance awaiting classification, it *is* a close match to the unlabeled instance $U_4$. Because it is such a good match to this instance, we could simply change the label from $U_4$ to $F_2$, and add it to our dataset of labeled instances. In fact, the basic tenet of semi-supervised learning is that we can do this repeatedly, and thus end up with the situation shown in Figure 3.

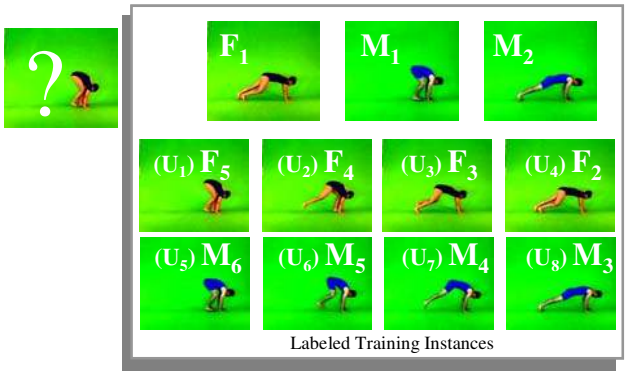


Figure 3: The small dataset of labeled instances shown in Figure 2 has been augmented by incorporating the previously unlabeled examples. Now the instance to be classified (marked with "?") is closest to $F_5$, and is correctly classified

It is important to note that the usefulness of unlabeled data depends on the critical assumption that the underlying models / features / kernels / similarity functions match well with the problem at hand [43]. Otherwise the addition of unlabeled data may degrade the performance of the classifier [1][7][32][40].

### 2.2 Semi-supervised Learning

Learning from both labeled and unlabeled data is called semi-supervised learning (SSL). Because semi-supervised learning requires less human effort and generally achieves higher accuracy, it is of great interest both in theory and in practice. There are many semi-supervised learning methods proposed in the literature. Based on their underlying assumptions, they can be organized into five classes: SSL with generative models, SSL with low density separation, graph-based methods, co-training methods, and self-training methods [9][43].

Generative models are the oldest semi-supervised learning methods. They assume that the data are drawn from a mixture distribution which can be identified by large amounts of unlabeled data. The strength of the generative approach is that knowledge of the structure of the data can be naturally incorporated into the model. It has been applied to diverse domains including text classification [32] and face orientation discrimination [1]. However, to our knowledge, there has been no discussion of the mixture distribution assumption for time series data in the literature.

Low density separation approaches try to leverage off the assumption "*the decision boundary should lie in a low density region*" by pushing the decision boundary away from the unlabeled data. The most common approach to achieve this goal is to use a margin maximization algorithm such as Transductive Support Vector Machines (TSVM). Since finding the exact TSVM solution is NP-hard, several approximation algorithms have been proposed [4][10][15][16][21]. However, the unique structure of time series makes the density measure less meaningful. For example, in [23] Keogh et. al. showed that "*(abnormal time series) do not necessarily live in sparse areas of n-dimensional space*" and "*repeated patterns do not necessarily live in dense parts*".

[1] If viewing this graphic on a monochrome printout, it may be helpful to note that the male actor has a knee length leotard.

Recently graph-based semi-supervised learning methods have received a lot of attention. Based on the assumption that "*the (high-dimensional) data lie (roughly) on a low-dimensional manifold*", these methods represent the data by nodes in a graph, whose edges are the distances between the nodes. After the graph is constructed, several approaches can be used, such as graph mincut [5], Tikhonov Regularization [2], Manifold Regularization [3], etc.. The key problem of this method is that graph construction needs to be hand crafted for each domain, because it encodes prior knowledge. In this paper, we are looking for a general semi-supervised classification framework for time series, so we do not consider graph-based methods.

The idea of co-training was first proposed by Blum and Mitchell [6]. It divides the features of the data into two disjoint sets, with each set being sufficient to train a good classifier. Two classifiers are trained separately on each feature subset, and the predictions of one classifier are used to enlarge the training set of the other. For example, in our contrived problem in Figure 2, one classifier could use the shape features, and the other classifier could use only color features. The underlying assumption of the co-training approach is that features of data are independent and can be divided. However, time series is known to have very high feature correlation [22], which makes the co-training approach infeasible for this type of data.

One of the least studied semi-supervised learning methods is self-training [43]. In self-training, a classifier is first trained by the small amount of labeled data. It then classifies the unlabeled data, and adds the most confidently classified examples (along with their predicted labels) into the training set. The procedure repeats and the classifier is gradually refined. The classifier is actually using its own predictions to teach itself. Because of its generality and very few assumptions, we use self-training as a starting point for our work.

Note that this review of semi-supervised learning is necessarily brief. We refer the interested reader to [9] and [43] for a more detailed treatment.

## 2.3 Time Series Classification

Although we believe that this is the first paper to formally address semi-supervised classification of time series, a thorough literature search and personal experience suggest that people working on real world time series problems have already done this informally. For example in the context of motion capture indexing, Kovar and Gleicher [25] noted that they *"…add robustness to the search by concentrating on finding these closer motions and then using them as new queries in order to find more distant motions*". Likewise in our own experience of building insect classifiers [41], faced with enormous amounts of sound data which contain relatively few labeled examples, we found this a useful technique. Below we place these ideas in a more formal footing.

For concreteness, we begin with a definition of our data type of interest, *time series*.

**Definition 1**. *Time Series*: A time series $T = t_1,\ldots,t_m$ is an ordered set of $m$ real-valued variables.

Time series data usually come in two formats: as a long time series (for example, eight hours recording of a patient's heartbeat) or as a set of short time series (for example, a set of individual abnormal heartbeats). Data miners are typically not interested in any of the *global* properties of a time series. For example, if we are given eight hours ECG data, we are not interested in classifying the whole time series; rather, we are interested in deciding whether each subsection is normal or abnormal. Therefore if we are given a long time series, we convert it into a set of short time series, where each time series in the set is a *subsequence* of the long time series.

**Definition 2**. *Subsequence*: Given a time series $T$ of length $m$, a subsequence $C_p$ of $T$ is a sampling of length $w < m$ of contiguous positions from $T$, that is, $C_p = t_p,\ldots,t_{p+w-1}$ for $1 \le p \le m - w + 1$.

The extraction of subsequences from a time series is achieved by use of a *sliding window*.

**Definition 3**. *Sliding Window*: Given a time series $T$ of length $m$, and a user-defined subsequence length of $w$, all possible subsequences can be extracted by sliding a window of size $w$ across $T$ and extracting each subsequence $C_p$.

The most common distance measure for time series is the *Euclidean distance*.

**Definition 4**. *Euclidean Distance*: Given two time series (or time series subsequences) $Q$ and $C$ both of length $n$, the Euclidean distance between them is the square root of the sum of the squared differences between each pair of corresponding data points:

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$

Before calling the distance function, each time series subsequence is normalized to have mean zero and a standard deviation of one, because it is well understood that in virtually all settings, it is meaningless to compare time series with different offsets and amplitudes [11][12][22][23][28][41].

**Definition 5**. *Time Series Classification:* Given a set of unlabeled time series, the task of time series classification is to map each time series to one of the predefined classes.

Time series classification has typically been treated like a classic discrimination problem, for example, the famous problem of distinguishing between "Democrat" and "Republication" in the UCI Vote dataset [31]. However we argue that realistic instances of the problem are much more like text filtering problems [32] in two important ways:

- It is typically not the case that we have two or more well defined classes. Rather we often have a positive class with some structure, say Premature Ventricular Event (from cardiology [18]) or Stuck Poppet Anomaly (from space telemetry launch monitoring [26]), and negative examples that have little or no common structure. The reason why negative examples have no well-defined structure is because every subsequence extracted from a sliding window must be either classified as positive or negative. We cannot in general assume that subsequences not belonging to the positive class look similar to each other. As a consequence, usually there is only one (or some small number of) way(s) to be in the positive class, while there are an essentially infinite number of ways to be in the negative class.
- Like text filtering, it is typically the case that positive labeled examples are rare, but unlabeled data is abundant. For example, ECG data is often collected continuously overnight when patients are sleeping, which makes real-time annotation almost impossible. Usually cardiologists annotate at most the first five minutes of the ECG data.

Based on these two observations, we focus on building binary time series classifiers for extremely imbalanced class distributions, with only a small number of labeled examples from the positive class.

# 3 SEMI-SUPERVISED TIME SERIES CLASSIFICATION

In this section, we begin by introducing the one-nearest-neighbor classifier. Later we show the special considerations necessary to convert it to a semi-supervised framework.

## 3.1 One-nearest-neighbor with Euclidean Distance Classifier

The problem of time series classification has attracted great interest recently. Although many algorithms have been proposed, it has been shown that one-nearest-neighbor with Euclidean distance is very difficult to beat [22]. For example, many different classification techniques have been tried on the famous Control-Chart problem [31], see Table 1. However, to our knowledge, none of them can beat the simple one-nearest-neighbor with Euclidean distance approach. Furthermore, most approaches listed in Table 1 are quite complicated and require many parameters to be set, whereas one-nearest-neighbor with Euclidean distance is parameterless. In addition, Keogh et. al. [22] have conducted an extensive set of experiments to show that one-nearest-neighbor with many other similarity measures can not beat the simple strawman.

Table 1: The error rates for various classification techniques on Control-Chart Dataset

| Approach | Error Rate |
|---|---|
| One-nearest-neighbor with Euclidean distance [22] | **1.3%** |
| First order logic rules with boosting [38] | 3.6% |
| Multi layer perceptron neural network [30] | 1.9% |
| Multiple classifier system [11] | 7.2% |
| Multi-scale histogram approach [12] | 6.0% |

Note that the works in Table 1 *do* make contributions in telling us something about boosting, neural network, or other classification methods. In addition, the authors are to be commended for experimenting on datasets that are in the public domain. Our point is simply that if you want accurate classification of time series, one-nearest-neighbor with Euclidean distance is *very* hard to beat. For this reason, we only consider one-nearest-neighbor with Euclidean distance in this work.

## 3.2 Training the Classifier

In the previous section, we have shown that one-nearest-neighbor with Euclidean distance is very competitive for time series classification. Therefore we have adopted it as our base classifier. Note that it also needs a large labeled training set to work well. Below we show how to apply semi-supervised learning to make it feasible for the situation where only a small set of labeled data is available. The idea is simple. We let the classifier train itself through the following steps:

Step 1. The classifier is trained on the initial training set, where all labeled instances are positive and all unlabeled instances are regarded as negative (recall that an instance must be either positive or negative, see Section 2.3). Note that the size of the training set never changes during the training process, but the labeled set is augmented gradually.

Step 2. The classifier is used to classify the unlabeled data in the training set. For each unlabeled instance, we find its nearest neighbor in the training set. If its nearest neighbor is labeled (as positive of course), the instance will be classified as positive. Otherwise, its nearest neighbor has not been labeled (and thus is negative) and we classify the instance as negative.

Step 3. Among all the unlabeled instances, the one we can most confidently classify as positive is the instance which is closest to the labeled positive examples. This instance, along with its newly acquired positive label, will be added into the positive set. With the training set being adjusted, we go back to Step 1 to refine the classifier. The procedure repeats until some stopping criterion is reached (we will discuss the stopping criterion in more detail later).

The intuition of the idea is straightforward. The labeled positive examples serve as a model which describes what a positive example "looks like". If an unlabeled instance is very similar to a positive example, the probability of it being positive is very high. For example, in our contrived problem in Figure 2, the unlabeled instance $U_4$ is a very close match to the labeled instance $F_1$. Therefore we can label it as F (female) with high confidence. By adding such an example into the positive set, we are refining the description of the positive class, which in turn will help in classifying the unlabeled data. The hope is that the modeling process and the classification process can reinforce each other iteratively and correctly label as many positive examples as possible.

In Table 2 we formalize this idea. Given a set *P* of positively labeled examples and a set *U* of unlabeled examples, the algorithm iterates the following procedure. First, use *P* and *U* to train the one-nearest-neighbor classifier *C* (note again we regard instances in *P* as positive examples and instances in *U* as negative examples). Second, use classifier *C* to classify the unlabeled set *U*. Third, select one unlabeled example which is nearest to *any* instance in set *P* (breaking ties randomly), and add it to *P*.

Table 2: Semi-supervised time series classification algorithm

| | **Function** [*P*] = Semi_Supervised_Classification(*P*, *U*) |
|---|---|
| 1 | **Until** (some stopping criterion) |
| 2 | use *P* and *U* to train the one-nearest-neighbor classifier *C* |
| 3 | use classifier *C* to classify unlabeled set *U* |
| 4 | select the example that *C* most confidently labels as positive |
| 5 | add this example into *P* |
| 6 | delete this example from *U* |
| 7 | **End** |

To be concrete, in Figure 4 we demonstrate our algorithm with a simple two-class toy problem, where initially only one example is known as positive (the solid square in Figure 4). Using our approach, we can correctly classify almost all the examples in the positive class after seventeen iterations, as shown in Figure 4. In contrast, if we simply put the seventeen nearest neighbors of the single labeled example to the positive class, we will get very poor accuracy.

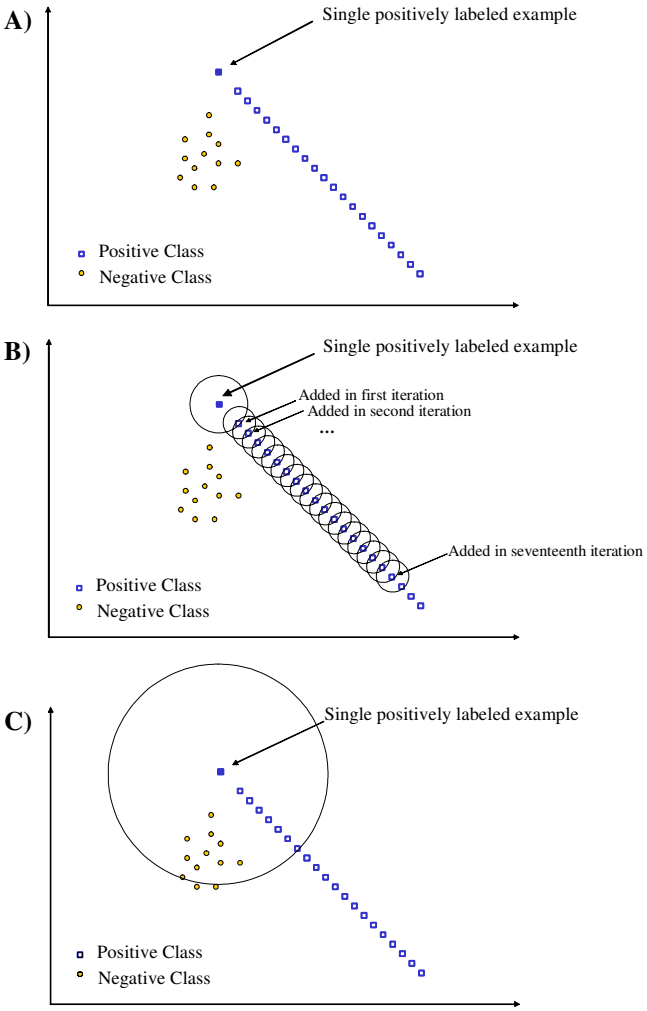


Figure 4: **A)** A simple two-class dataset. **B)** The chaining effect of semi-supervised learning: a positive example is labeled which helps labeling other positive examples and so on. Eventually all positive examples are correctly classified. **C)** If we simply put the seventeen nearest neighbors of the single labeled example to the positive class, we would wrongly include many negative examples into the positive class

### 3.2.1 Stopping Criterion

As we will show in the empirical evaluation, a self-training classifier can achieve high accuracy with only a handful of labeled examples. In this section, we will discuss the stopping criterion for training the classifier, an issue we have deliberately ignored to this point. Ideally we would like the training procedure to stop when the performance (accuracy or precision-recall etc.) of the classifier begins to deteriorate. However, it is very hard (if not impossible) to know the true performance of the classifier, because we do not know the ground truth of the data.

In our case, we are using a distance-based classifier. So the distance statistics may give us some hint about how well the classifier is doing. To develop our intuition, we perform self-training classification on several datasets and look at the minimal distance between two instances in the labeled positive set. For each iteration in the training procedure, we record the precision-recall breakeven point (explained in greater detail in Section 4) and the distance between the closest pair in the labeled positive set. Figure 5 shows the results obtained on the ECG dataset (a detailed description of the ECG dataset can be found in Section 4.1). We can see that the minimal nearest neighbor distance decreases dramatically in the first few iterations, stabilizes for a relatively long time, and drops again. Interestingly, the precision-recall breakeven point achieved by the classifier has a corresponding trend of increasing, stabilizing, and decreasing.

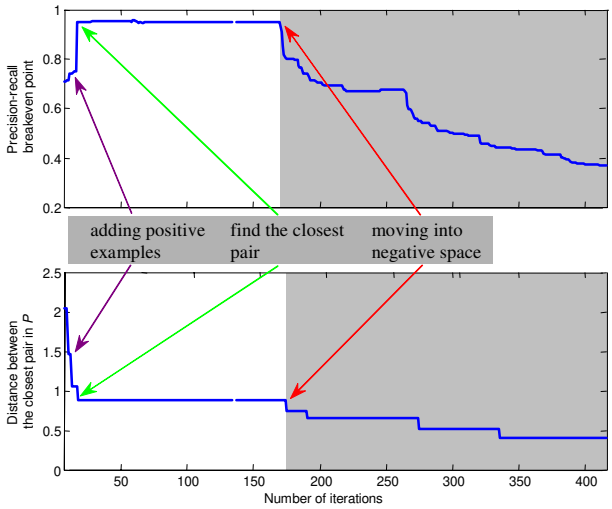


Figure 5: Statistics on ECG dataset

In hindsight, this phenomenon is not surprising. In the first few iterations, the labeled positive set is relatively small. In other words, the known positive space is relatively sparse. By adding more positive examples into it, the space gets denser, and as a result, the minimal nearest neighbor distance decreases gradually. At some point, the closest pair of the positive examples is incorporated in the labeled set. The minimal nearest neighbor distance will be the distance between them. Adding more positive examples will not change the minimal distance (this corresponds to the stabilizing phase). However if a negative example is being labeled as positive, chances are high that we will keep adding negative examples because the negative space is much denser than the positive space. And the closest pair in the labeled positive set will be the pair of two negative examples. Thus we will see a drop of the minimal nearest neighbor distance of the positive set. Figure 6 illustrates the process on a small sample dataset.

Similar observations were made on other datasets. These indicate that, even though the question of when to stop the self-training procedure remains unsolved and is an open problem, we can use the change in the minimal nearest neighbor distance in the labeled positive set as a good heuristic in most cases.

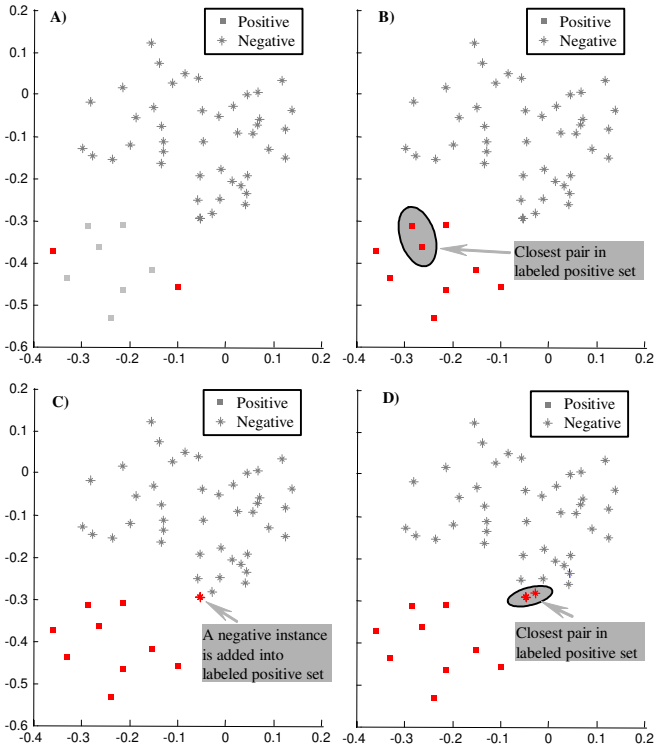


Figure 6: A sample dataset shown in two-dimensional space. **A)** Initially the two solid (red) squares are labeled as positive. **B)** At some point the closest pair in the positive set is added into labeled positive set. **C)** A negative instance is being added into labeled positive set. **D)** The closest pair in labeled positive set changes to two negative instances

### 3.3 Using the Classifier

By the end of the training, much more data in the training set are labeled, and we can use the classifier to classify other datasets. At first glance, this is easy. For each instance to be classified, check whether its nearest neighbor in the training set is labeled or not, and assign it the corresponding class label. However recall that the training set is huge (because of the enormous amount of negative examples). Comparing each instance in the testing set to each example in the training set is untenable in practice.

To make the classification tractable, we modify the classification scheme of the one-nearest-neighbor classifier, using *only* the labeled positive examples in the training set. If an instance to be classified is within $r$ distance to any of the labeled positive examples, it will be classified as positive. Otherwise it is negative. Recently we have successfully applied this scheme to the problem of monitoring streaming time series for a set of predefined patterns [41]. A natural value for $r$ would be the average distance from a positive example to its nearest neighbor. The intuition is that if the positive examples we have seen before tended to be about $r$ apart, then a future positive object will probably also be within $r$ of one (or more) positive example(s) in the training set. Paradoxically, we may be victims of our own success. By greatly enlarging the size of the labeled positive set with our semi-supervised algorithm, it appears that we will greatly increase the time taken to classify new instances. Fortunately this is not the case. We can leverage off an envelope-based lower-bounding technique [41] to speed up the classification procedure. For example, in [41] we applied this technique on an ECG dataset and the speedup achieved is more than 100 times. Because we are focusing on the effectiveness of the semi-supervised learning classifier in this paper, we will not discuss the speedup technique any more. We refer interested readers to [41] for more details.

## 4 EMPIRICAL EVALUATION

In this section, we test our semi-supervised learning classifier with a comprehensive set of experiments on diverse domains. We compare the semi-supervised approach to a naive $k$-nearest-neighbor approach, where the $k$ nearest neighbors of the labeled positive set are classified as positive and others as negative (see Figure 4.C for an example). As the reader may already appreciate, the setting of $k$ is a non-trivial problem, since the classifier does not know in advance how many positive examples there are in the testing set. To help the strawman achieve the best performance, we allow it to search over all possible values of $k$ and only report the *best* result.

The performance of the classifier at each iteration is reported using precision-recall breakeven point. Since the class distribution is highly skewed, accuracy is not a good performance metric. The classifier can simply classify everything as negative to ensure high accuracy. Note that precision-recall breakeven point is a standard information retrieval measure for binary classification [20][32]. Precision and recall are defined as:

$$\text{Precision} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}}$$

$$\text{Recall} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive examples}}$$

The precision-recall breakeven point is the value at which precision and recall are equal [20]. It is a single performance value over all binary classification tasks and it is insensitive to the distribution of the classes.

For simplicity in the experiments we did not evaluate the stopping heuristic described in Section 3.2.1. We just keep training the classifier until it achieves its highest precision-recall and allow a few more iterations after that. For most of the experiments, we use distinct training set and testing set. The Word Spotting dataset and Yoga dataset are too small to be split, so for them we train and test on the same dataset. However we note that it is still non-trivial to classify the training set because most data in the training set are unlabeled.

### 4.1 ECG Dataset

As noted earlier, heartbeat classification has received a lot of attention because of the large amounts of freely available data and the potential applications in medical field. Our first experiment is on an ECG dataset obtained from the MIT-BIH Arrhythmia Database [18]. Each data record in the ECG dataset is a time series of the measurements recorded by one electrode during one heartbeat. The data has been annotated by cardiologists and a label of normal or abnormal is assigned to each data record. Of the 2,026 data records in the dataset, 520 were identified as abnormal and 1,506 were identified as normal. All the data records have been normalized and rescaled to have length 85 (recent results suggest that we lose nothing by rescaling [35]). We randomly split the data, using half for training and half for testing,

as summarized in Table 3. Because usually cardiologists are more interested in the occurrences of the abnormal heartbeats, here abnormal heartbeats are our target (positive class).

Table 3: Number of positive and negative instances in the training set and the testing set for ECG Dataset

| | Training Set | Testing Set |
|---|---|---|
| **Positive (Abnormal)** | 208 | 312 |
| **Negative (Normal)** | 602 | 904 |
| **Total** | 810 | 1,216 |

For the semi-supervised approach, we randomly choose 10 positive examples in the training set as the initial labeled positive set $P$. In each iteration, the semi-supervised algorithm adds one example to the positive set $P$ and uses the adjusted training set to classify the testing set. The precision-recall breakeven point achieved is recorded. Note that the initial labeled set $P$ has an effect on the performance of the classifier (a good initial set $P$ may give the classifier a high precision-recall in the beginning while a bad initial set $P$ may take the classifier more iterations to achieve good performance). To avoid the bias introduced by the initial set, we ran the experiments 200 times and report the results in Figure 7. The bold line is the average performance over the 200 runs. The gray lines bounding it from above and below are one standard deviation intervals. Note that the precision-recall breakeven value increases dramatically in the beginning and stabilizes after about ten iterations. On average, the maximal precision-recall breakeven value achieved by the semi-supervised approach is 94.97%. The shaded area in Figure 7 is where the performance of the semi-supervised classifier deteriorates because it begins to ingest negative examples.

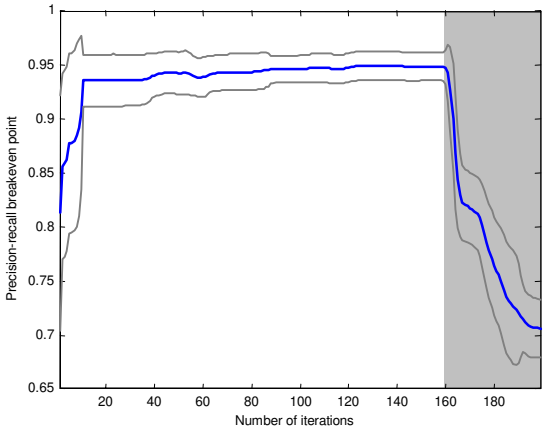


Figure 7: Classification performance on ECG Dataset

We then ran another 200 experiments (each time with the same initial labeled set $P$ as used in the semi-supervised experiment) for the naive $k$-nearest-neighbor approach (with $k$ = 312). However, even with the optimal $k$ value, the $k$-nearest-neighbor approach only achieves an average precision-recall breakeven value of 81.29%, which is much lower than that of the semi-supervised approach. This shows that with the help of the unlabeled data, the semi-supervised approach can greatly increase the performance of the classifier.

## 4.2 Word Spotting Dataset

In second experiment, we consider classification of handwritten documents. We test on the Word Spotting dataset, which was created by Rath and Manmatha for word image matching [36]. It contains 2,381 word images from 10 handwritten pages. We take the images of 50 common words such as "the", "and", etc. and obtain 905 instances in total. Each word image is represented by a four dimensional time series which describes the profile of the image. For example, in Figure 1, we have shown the upper profile of the word "Alexandria". For simplicity we only consider the first dimension of each image, which is of an average length of 270. Here we focus on the two-class problem of differentiating the word "the" from others. In total, there are 109 images for word "the" and 796 images for other words. In this experiment, we use the same 905 images both for training and testing, as summarized in Table 4.

Table 4: Number of positive and negative instances in the training set and the testing set for Word Spotting Dataset

| | Training Set | Testing Set |
|---|---|---|
| **Positive (Word "the")** | 109 | 109 |
| **Negative (Other words)** | 796 | 796 |
| **Total** | 905 | 905 |

As before, for the semi-supervised approach, each time we randomly choose 10 positive examples in the training set as the initial labeled positive set $P$ and record the precision-recall breakeven point for each iteration. We repeated the experiment 25 times and the results are shown in Figure 8. The bold line is the average performance over the 25 runs, and the gray lines are one standard deviation intervals. We can see that the performance increases steadily at the beginning, reaches its maximal value 86.2% at about fifty iterations, and then begins to decrease (the shaded area in Figure 8). We then ran the same 25 experiments for the naive $k$-nearest-neighbor approach (with $k$ = 109). On average, the precision-recall breakeven value obtained by the $k$-nearest-neighbor approach is only 79.52%.

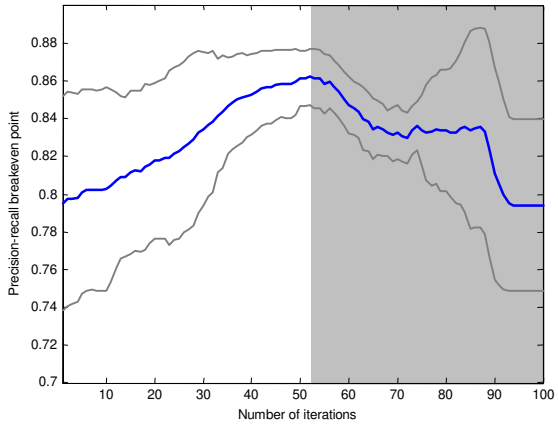


Figure 8: Classification performance on Word Spotting Dataset

Handwritten text is an intuitive domain so we spend more time analyzing its results. For example, it is instructive to take a closer look at what happened during the training procedure. Figure 9 shows the changes of the rankings of two instances during the training process, where Image 19 is a positive example and Image 585 is a negative example. The ranking of an instance is determined by its distance to the labeled positive set – the larger the distance, the higher the ranking. So an instance with higher ranking has lower probability to eventually be classified as positive. In Figure 9, as training begins, Image 19 has a relatively high ranking, while Image 595 has a relatively low ranking. This represents a bad initial labeled set, where Image 19 happened to

be similar to none of the examples in the initial labeled positive set, while Image 595 is similar to one or more of them. Fortunately, even with a bad start, the semi-supervised learning classifier is able to correctly label more positive examples, which in turn helps it model the positive examples better. As a result, the ranking of Image 19 decreases and the ranking of Image 585 increases after several iterations.

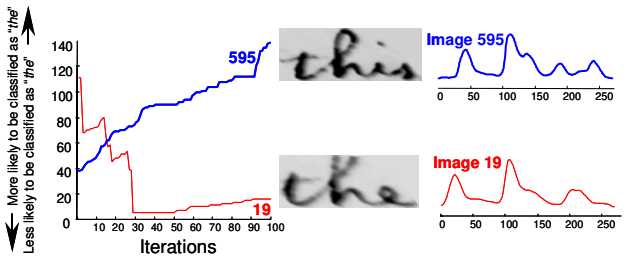


Figure 9: Ranking changes of two instances in Word Spotting dataset during semi-supervised training

## 4.3 Gun Dataset

The gun dataset contains two-dimensional time series extracted from video of two actors either aiming a gun or simply pointing at a target. The two dimensions correspond to the X and Y coordinates of the actors' right hand. For simplicity we only consider the Y-axes here. The dataset contains four classes:

Class A: Actor 1 with gun

Class B: Actor 1 without gun (point)

Class C: Actor 2 with gun

Class D: Actor 2 without gun (point)

Here we focus on the two-class problem of differentiating Actor 1 with gun from others – (A) vs. (B+C+D). In total, there are 57 instances in Class A, and 190 instances in other classes. Each instance has the same length of 150. Again, we randomly split the data, using half as the training set and half as the testing set, as summarized in Table 5.

Table 5: Number of positive and negative instances in the training set and the testing set for Gun Dataset

| | Training Set | Testing Set |
|---|---|---|
| **Positive (Class A)** | 27 | 30 |
| **Negative (Class B,C,D)** | 95 | 95 |
| **Total** | 122 | 125 |

In this experiment, we start with one labeled positive example and train the classifier. We ran the experiment 27 times (once for each positive example) and show the results in Figure 10. Starting with only one labeled example, the classifier is able to identify other positive examples and achieves a maximal precision-recall breakeven point of 65.19% on average. One may notice that the variance of this experiment is higher than that of the previous ones (in Figure 10 the two gray lines are farther away from the bold line). This is because we start with a single labeled example, which increases the bias of the initial labeled set.

We ran the same experiments using the $k$-nearest-neighbor classifier (with $k$ = 27), the average precision-recall breakeven point achieved is 55.93%. This again shows the superior of our semi-supervised approach: it only needs a small number of labeled examples (as few as one in this case) to build accurate classifier.

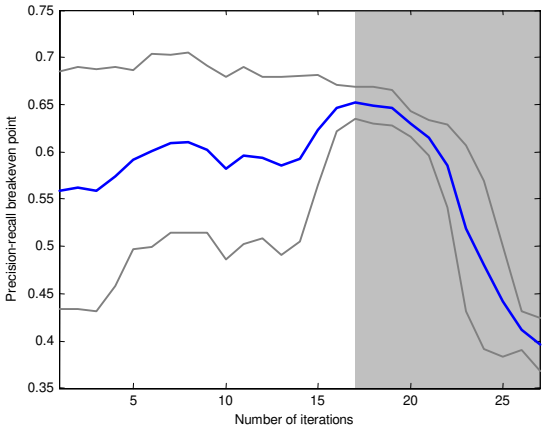


Figure 10: Classification performance on Gun Dataset

## 4.4 Wafer Dataset

The wafer dataset is a collection of time series containing a sequence of measurements recorded by one vacuum-chamber sensor during the etch process of silicon wafers for semiconductor fabrication [33]. Each wafer has an assigned classification of normal or abnormal. The abnormal wafers are representative of a range of problems commonly encountered during semiconductor manufacturing. Of the 7,164 time series in wafer dataset, 762 were identified as abnormal and 6,402 were identified as normal. We randomly picked half the dataset as the training set and used the other half as the testing set. Table 6 summarizes the contents of the training and testing set. As in the ECG experiment, the abnormal data are our target.

Table 6: Number of positive and negative instances in the training set and the testing set for Wafer Dataset

| | Training Set | Testing Set |
|---|---|---|
| **Positive (Abnormal)** | 381 | 381 |
| **Negative (Normal)** | 3,201 | 3,201 |
| **Total** | 3,582 | 3,582 |

For the semi-supervised approach, we ran the experiment 50 times, each time starting with one randomly chosen labeled positive example. The average performance is shown as the bold line in Figure 11. As we can see, the performance increases dramatically during the first few iterations and achieves a maximal precision-recall breakeven point of 73.17% on average.

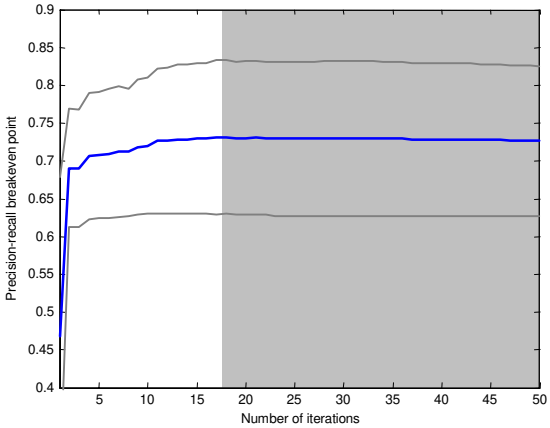


Figure 11: Classification performance on Wafer Dataset

We ran the same experiments using the *k*-nearest neighbor classifier (with $k$ = 381), and the average precision-recall achieved is only 46.87%.

### 4.5 Yoga Dataset

For our last experiment, we revisit the classification problem in Figure 2 on a realistic dataset. The dataset was obtained by capturing two actors transiting between yoga poses in front of a green screen. It has been shown recently that in many domains it can be useful to convert images into pseudo time series. Therefore we have converted the motion capture data into time series by a well-known technique as in Figure 12.

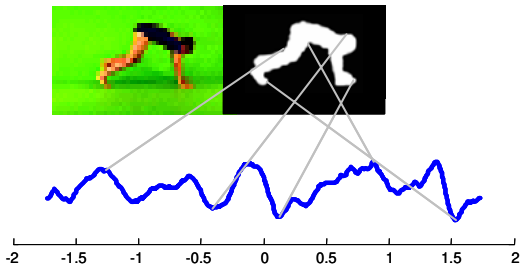


Figure 12: Shapes can be converted to time series. The distance from every point on the profile to the center is measured and treated as the Y-axis of a time series

In total, we have 316 time series with an average length of 426. Among them, 156 time series came from the female actor and 150 time series came from the male actor. We use the same dataset both for training and testing, as shown in Table 7.

Table 7: Number of positive and negative instances in the training set and the testing set for Yoga Dataset

| | **Training Set** | **Testing Set** |
|---|---|---|
| **Positive (Female)** | 156 | 156 |
| **Negative (Male)** | 150 | 150 |
| **Total** | 306 | 306 |

We ran the experiment 10 times, each time randomly choosing one positive example as labeled. The results are shown in Figure 13. As we can see, the precision-recall breakeven point increases steadily with the number of iterations, and gets to a maximum of 89.04% on average. While the same experiments on the naive *k*-nearest-neighbor approach (with $k$ = 156) only achieves an average precision-recall of 82.95%.

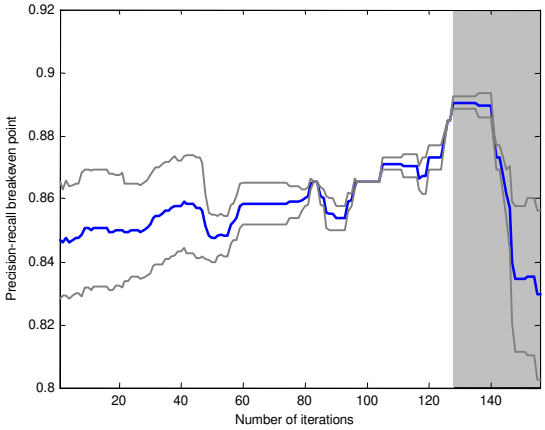


Figure 13: Classification performance on Yoga Dataset

## 5 CONCLUSIONS

It is well known that building accurate classifiers requires large quantities of labeled data and such labeled data is often difficult to obtain. To mitigate this discrepancy, we propose a semi-supervised learning framework to build accurate time series classifiers when only a small set of labeled examples is available. While there are many semi-supervised algorithms in other domains, their underlying assumptions rarely hold for time series data. Special considerations have been taken to make the semi-supervised classification both efficient and effective for the time series domain. The experimental results show that the reduction in the number of labeled examples needed can be dramatic: our self-training classifiers require only a handful of labeled examples to achieve high precision-recall. This suggests that the self-training method of using unlabeled data has a potential for significant benefits in time series classification.

There are many directions in which this work may be extended. We intend to perform a thorough investigation on the stopping criterion for the training process. In addition, we plan to extend our framework to other distance measures which have been shown to be effective, for example, Dynamic Time Warping (DTW) [35]. Finally, we are conducting a field study of insect classification using the semi-supervised approach.

## 6 ACKNOWLEDGEMENTS

We gratefully acknowledge the datasets donors. We also acknowledge insightful comments from Dr. Christian Shelton. Thanks also to Helga Van Herle M.D. for her expertise in cardiology, Dr. Raghavan Manmatha for help with the Word Spotting dataset, and Xiaopeng Xi for help with the Yoga dataset.

**Reproducible Research Statement:** In the interests of competitive scientific inquiry, all datasets used in this work are freely available at the following URL [42]. This research was partly funded by the National Science Foundation under grant IIS-0237918.

## 7 REFERENCES

[1] Baluja, S. (1998). Probabilistic modeling for face orientation discrimination: learning from labeled and unlabeled data. in *Neural Information and Processing Systems*, pp. 854-860, 1998.

[2] Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. *COLT*, 2004.

[3] Belkin, M., Niyogi, P., & Sindhwani, V. (2004). Manifold regularization: a geometric framework for learning from examples. Technical Report TR-2004-06, University of Chicago.

[4] Bennett, K. & Demiriz, A. (1999). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11, pp. 368-374, 1999.

[5] Blum, A. & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *proceedings of 18th International Conference on Machine Learning*, 2001.

[6] Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *proceedings of the Annual Workshop on Computational Learning Theory, 11th Annual Conference on Computational Learning Theory*, Madison, Wisconsin, pp. 92-100, 1998.

[7] Bruce, R. (2001). Semi-supervised learning using prior probabilities and EM. Presented at the *International Joint*

*Conference of AI Workshop on Text Learning: Beyond Supervision*, Seattle, Washington, 2001.

[8] Castelli, V. (1994). *The relative value of labeled and unlabeled samples in pattern recognition*. PhD thesis, Stanford University, CA, 1994.

[9] Chapelle, O., Scholkopf, B., & Zien, A. (2006). *Semi-Supervised Learning*. In press. MIT Press.

[10] Chapelle, O. & Zien, A. (2005). Semi-supervised classification by low density separation. In *proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

[11] Chen, L. & Kamel, M. S. (2005). Design of Multiple Classifier Systems for Time Series Data. *Multiple Classifier Systems*, pp. 216-225, 2005.

[12] Chen, L., Özsu, M. T., & Oria, V. (2005). Using Multi-Scale Histograms to Answer Pattern Existence and Shape Match Queries. In *proceedings of 17th International Conference on Scientific and Statistical Database Management*, 2005.

[13] Cohen, I., Cozman, F. G., Sebe, N., Cirelo, M. C., & Huang, T. (2004). Semisupervised learning of classifiers: theory, algorithms, and their application to human-computer interaction. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 26, no. 12, pp. 1553-1567, December 2004.

[14] Cohen, W. (1993). Efficient pruning methods for separate-and-conquer rule learning systems. In *proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France. pp. 988-994, 1993.

[15] Demirez, A. & Bennett, K. (2000). Optimization approaches to semisupervised learning. *Applications and algorithms of complementarity*. Boston: Kluwer Academic Publishers, 2000.

[16] Fung, G. & Mangasarian, O. (1999). Semi-supervised support vector machines for unlabeled data classification, Technical report 99-05, Data Mining Institute, University of Wisconsin Madison, 1999.

[17] Ganesalingam, S. & McLachlan, G. J. (1978). The efficiency of a linear discriminant function based on unclassified initial samples. Biometrika, vol. 65, pp. 658-662, December, 1978.

[18] Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P., Mark, R., Mietus, J., Moody, G., Peng, C., & He, S. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101(23): pp. 215-220, 2000.

[19] Hosmer, D. W. (1973). A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample. *Biometrics*, vol. 29, pp. 761-770, December 1973.

[20] Joachims T. (1998). Text categorization with support vector machines: learning with many relevant features. In *proceedings of 10th European Conference on Machine Learning*, pp. 137-142, 1998.

[21] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *proceedings of 16th International Conference on Machine Learning*, pp. 200-209, 1999.

[22] Keogh, E. & Kasetty, S. (2002). On the need for time series data mining benchmarks: A survey and empirical demonstration. In *proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 102-111, 2002.

[23] Keogh, E., Lin, J., & Fu, A. (2005). HOT SAX: Efficient finding the most unusual time series subsequence. In *proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pp. 226-233, 2005.

[24] Kibler, D. & Langley, P. (1988). Machine learning as an experimental science. In *proceedings of the 3rd European Working Session on Learning*. pp. 81-92, 1988.

[25] Kovar, L. & Gleicher, M. (2004). Automated extraction and parameterization of motions in large datasets. In *proceedings of SIGGRAPH '04*, pp. 559–568, 2004.

[26] Landford, J. P. & Quan, A. (2002). Evolution of knowledge-based applications for launch support. In *proceedings of Ground System Architecture Workshop*, El Segundo, CA, 2002.

[27] Levy, S. (2004). Google's Two Revolutions. *Newsweek*. Dec. 27 / Jan. 3 issue, 2004.
[Available at www.msnbc.msn.com/id/6733225/site/newsweek]

[28] Manmatha, R. & Rath, T. M. (2003). Indexing of Handwritten Historical Documents - Recent Progress. In: Proc. of the 2003 Symposium on Document Image Understanding Technology (SDIUT), Greenbelt, MD, pp. 77-85, April 9-11, 2003.

[29] Moreno, P. J. & Agarwal, S. (2003). An experimental study of EM-based algorithms for semi-supervised learning in audio classification. In *proceedings of the ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data*, Washington, DC, 2003.

[30] Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (2001). Feature-based Classification of Time-series Data. *International Journal of Computer Research*, pp. 49-61, 2001.

[31] Newman, D.J., Hettich, S., Blake, C.L., & Merz, C.J. (1998). UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[32] Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3), pp. 103 – 134, 2000.

[33] Olszewski, R. T. (2001). *Generalized feature extraction for structural pattern recognition in time-series data*. PhD thesis, Carnegie Mellon University, 2001.

[34] O'Neill, T. J. (1978). Normal discrimination with unclassified observations. Journal of the American Statistical Association, vol. 73, no. 364, pp. 821-826, 1978.

[35] Ratanamahatana, C. A. & Keogh. E. (2004). Everything you know about Dynamic Time Warping is wrong. In *proceedings of the Third Workshop on Mining Temporal and Sequential Data*, in conjunction with *the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 22-25, 2004.

[36] Rath, T. & Manmatha, R. (2003). Word image matching using dynamic time warping. In *proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vol. II, pp. 521-527, 2003.

[37] Ratsaby J. & Venkatesh S. S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. In *proceedings of the Eighth Annual Conference on Computational Learning Theory*, pp. 412-417, 1995.

[38] Rodríguez, J. J., Alonso, C. J., & Boström, H. (2000). Learning First Order Logic Time Series Classifiers: Rules and Boosting. In *Proceedings of 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000),* pp. 299-308, 2000.

[39] Rosenberg C. & Hebert M. (2002). Training object detection models with weakly labeled data. BMVC 2002. In the *electronic proceedings of the 13th British Machine Vision Conference*, United Kingdom, 2002.

[40] Shahshahani, B. & Landgrebe, D. (1994). Effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 5, pp. 1087-1095, 1994.

[41] Wei, L., Keogh, E., Van Herle, H., & Mafra-Neto, A. (2005). Atomic Wedgie: Efficient Query Filtering for Streaming Time Series. In *proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pp. 490-497, 2005.

[42] Wei, L. (2006). http://www.cs.ucr.edu/~wli/selfTraining/

[43] Zhu, X. (2005). Semi-supervised learning literature survey. Technical report, no. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.