

Using Dynamic Time Warping Distances as Features for Improved Time Series Classification

Rohit J. Kate

Received: 12 September 2014/ Accepted: 23 April 2015

Abstract Dynamic Time Warping (DTW) has proven itself to be an exceptionally strong distance measure for time series. DTW in combination with one-nearest neighbor, one of the simplest machine learning methods, has been difficult to convincingly outperform on the time series classification task. In this paper, we present a simple technique for time series classification that exploits DTW's strength on this task. But instead of directly using DTW as a distance measure to find nearest neighbors, the technique uses DTW to create new features which are then given to a standard machine learning method. We experimentally show that our technique improves over one-nearest neighbor DTW on 31 out of 47 UCR time series benchmark datasets. In addition, this method can be easily extended to be used in combination with other methods. In particular, we show that when combined with the Symbolic Aggregate Approximation (SAX) method, it improves over it on 37 out of 47 UCR datasets. Thus the proposed method also provides a mechanism to combine distance-based methods like DTW with feature-based methods like SAX. We also show that combining the proposed classifiers through ensembles further improves the performance on time series classification.

1 Introduction

Time series classification has applications in many domains including medical, biological, financial, engineering and industrial (Keogh and Kasetty 2003). Due to the enormous interest in this task, researchers have proposed several methods over the past few decades. These methods broadly fall under two categories: distance-based methods and feature-based methods. Under the

Rohit J. Kate
University of Wisconsin-Milwaukee
2025 E. Newport Avenue
Milwaukee, WI 53211, USA
E-mail: katerj@uwm.edu

distance-based methods, a distance function is first defined to compute similarity between two time series. Then a time series is typically classified as belonging to the same class as its nearest time series present in the training data according to the distance function. Euclidean distance (ED) (Faloutsos et al 1994), dynamic time warping (DTW) (Berndt and Clifford 1994), edit distance (Chen and Ng 2004), longest common subsequence (Vlachos et al 2002) etc. are some examples of distance functions that have been used for time series classification. Under the feature-based methods, a statistical (Nanopoulos et al 2001; Geurts 2001) or symbolic (Lin et al 2007; Ye and Keogh 2009) feature representation is first defined for time series. Then any one of the several machine learning methods is employed that learns to classify time series from the training data using the feature-based representation.

Interestingly, it has been found that in spite of numerous distance measures, feature representations, and specialized machine learning methods that have been proposed for time series classification, the decades old dynamic time warping (DTW) in combination with one-nearest neighbor has not been easy to beat convincingly (Xi et al 2006; Ding et al 2008; Xing et al 2010; Chen et al 2013; Wang et al 2013; Lines and Bagnall 2014). Although one would expect, as in other classification tasks, that a powerful machine learning method will significantly outperform one-nearest neighbor, which is perhaps the simplest machine learning method, this has not been the case for time series classification. We believe that the real problem has been that given the sequential and numeric nature of time series, it is hard to come up with good features for them that would work well with machine learning methods. On the other hand, DTW being a distance measure between two time series, does not lend itself to be used as a feature in representing a time series.

In this paper, we present a technique for time series classification which exploits the strength of DTW on this task as well as uses a powerful machine learning method. The idea is surprisingly simple. We represent a time series in terms of its DTW distances from each of the training examples. Hence the DTW distance from the first training example becomes the first feature, the DTW distance from the second training example becomes the second feature and so on. We then use support vector machine (SVM) (Cristianini and Shawe-Taylor 2000) as the classification algorithm. Instead of relying on the class of the nearest time series, this way the method is able to learn how the class of a time series relates to its DTW distances from various training examples. We present results that show DTW and its window-size constrained version used in this way as features with SVM improves over DTW used directly with one-nearest neighbor on 31 out of 47 ($p < 0.05^1$) UCR time series benchmark datasets (Keogh et al 2011). Not just DTW, we show that even the Euclidean distance when used as features with SVM improves over Euclidean distance when used directly with one-nearest neighbor on 35 out of the 47 datasets ($p < 0.05^1$). Although this idea of using DTW as features has been explored

¹ Using two-tailed Wilcoxon signed-rank test.

previously (Gudmundsson et al 2008), this paper is the first to report that it convincingly outperforms DTW used with one-nearest neighbor.

In addition to being simple and easy to implement, our method can be easily extended to be used in combination with other statistical and symbolic feature-based methods by simply adding them as additional features. To demonstrate this, we experimentally show that when our method is combined with Symbolic Aggregate Approximation (SAX) method (Lin et al 2012), it outperforms it on 37 out of the 47 UCR datasets ($p < 0.05^1$). The combination of DTW and SAX features itself improves over 26 of the 47 datasets when compared to using DTW features alone. We also created ensembles of various classifiers presented in this paper and show that they further improve the performance. All our code to reproduce the results of this paper and our detailed results are available online (Kate 2014).

We want to point out that it is not the DTW features themselves but their use in conjunction with a better machine learning method like SVM that improves the performance over one-nearest neighbor. However, without using DTW distances as features it is not clear how one could use them in a learning method like SVM or combine them with feature-based methods like SAX.

2 Background

This section gives a very brief overview of the DTW distance measure and the SAX representation for time series.

2.1 Dynamic Time Warping

Given two time series $Q = q_1, q_2, \dots, q_i, \dots, q_n$ and $C = c_1, c_2, \dots, c_i, \dots, c_m$, perhaps the simplest distance between them is the Euclidean distance (ED) defined as follows:

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

which can only be computed if $n = m$. Its simplicity, efficiency, and being a distance metric has made Euclidean distance a very popular distance measure for many data mining tasks and it usually performs competitively. However, besides requiring the two time series to be of equal lengths, Euclidean distance has another disadvantage that it is very sensitive to even small mis-matches among the two time series. For example, if one time series is only slightly delayed or shifted from the other but otherwise exactly the same, then Euclidean distance between them will be unreasonably large. Both these disadvantages of Euclidean distance are elegantly overcome by the Dynamic Time Warping (DTW) distance. This technique has been known to the speech processing community for a long time (Itakura 1975; Sakoe and Chiba 1978) and was later introduced to time series problems (Berndt and Clifford 1994).

Given the two time series Q and C , DTW distance is computed by first finding the best alignment between them. To align the two time series, an n -by- m matrix is constructed whose (i^{th}, j^{th}) element is equal to $(q_i - c_j)^2$ which represents the cost to align the point q_i of time series Q with the point c_j of time series C . An alignment between the two time series is represented by a warping path, $W = w_1, w_2, \dots, w_k, \dots, w_K$, in the matrix which has to be contiguous, monotonic, start from the bottom-left corner and end at the top-right corner of the matrix. The best alignment is then given by a warping path through the matrix that minimizes the total cost of aligning its points, and the corresponding minimum total cost is termed as the DTW distance. Hence,

$$DTW(Q, C) = \arg \min_{W=w_1, \dots, w_k, \dots, w_K} \sqrt{\sum_{k=1, w_k=(i,j)}^K (q_i - c_j)^2} \quad (2)$$

The minimum cost alignment is computed using a dynamic programming algorithm. Typically, some constraints are imposed on possible alignments that speed up the computation and also improve the accuracy (Ratanamahatana and Keogh 2004b). One of the simplest constraint is to not allow the warping path to drift very far from the matrix diagonal (Sakoe and Chiba 1978). In this constraint, the distance the path is allowed to wander from the diagonal is restricted to a window (also called Sakoe-Chiba band) of size r , which is a parameter. In Equation 2, this constraint will manifest as a restriction of $|i-j| \leq r$ for every w_k . This window constraint not only speeds up computation of DTW, but somewhat non-intuitively, also improves the accuracy of time series classification (Ratanamahatana and Keogh 2004a). In the rest of the paper, we will refer to this window-size constrained DTW as DTW-R and call the one-nearest neighbor method based on DTW-R as 1NN-DTW-R. It should be noted that when the window size parameter r is set to zero then the only warping path possible in the matrix is the diagonal itself, and in that case DTW-R reduces to the Euclidean distance.

DTW-R has been widely used for time series classification using the simple one-nearest neighbor (1NN) approach.² A test time series is classified as belonging to the same class as its nearest time series in the training set according to the DTW-R distance. This simple classification technique, 1NN-DTW-R, has been well-acknowledged as exceptionally difficult to beat (Xi et al 2006; Ding et al 2008; Xing et al 2010; Chen et al 2013; Wang et al 2013; Lines and Bagnall 2014). Often techniques that are shown to outperform 1NN-DTW-R, are shown to do so either only on a few select datasets or through an unfair comparison. Xi et al (2006) give a brief review of various techniques that were proposed and how they do not satisfactorily beat 1NN-DTW-R. Also, some papers, for example, (Gudmundsson et al 2008) and (Hills et al 2013), do not compare with the 1NN-DTW-R error rates available from (Keogh et al 2011). In some papers the experimental setting is not clear, for example, Baydogan

² Using more than one nearest neighbors has not been found to be helpful.

et al (2013) took average of ten different error rates of their method and it is not clear which of the settings is best.

In this paper, we show that a simple technique that employs DTW and DTW-R as features, which enables it to use SVM, outperforms 1NN-DTW-R on 31 of the 47 UCR time series benchmark datasets (and ties on 1 dataset) which is also statistically significant. We also want to point out that given that our method is simple and extensible, any other method that does well on time series classification can be potentially further improved by combining it with our method. In this paper, we show improvement over the method based on the SAX representation of time series (Lin et al 2012).

2.2 Symbolic Aggregate Approximation

Symbolic Aggregate Approximation (SAX) is a method of representing time series in a lower dimensional space of symbolic “words” (Lin et al 2007). Given a piece of time series of length n , SAX first normalizes it to have zero mean and unit standard deviation and then divides it into w equal-sized segments, where w is a parameter representing the word size. Next, the mean of each of these segments is computed which forms a Piecewise Aggregate Approximation (PAA) representation. Each segment mean is next mapped to an alphabet from an alphabet size of a (a parameter) using a table lookup that divides the distribution space into a equiprobable regions. The distribution is assumed to be Gaussian which has been found to work best for time series subsequences (Lin et al 2007). Thus the entire process maps a piece of time series of length n to a word of length w consisting of alphabets from an alphabet size of a .

In order to convert an entire time series into its SAX representation, a window of length n is slid over it (with overlaps) and a SAX word is computed for each piece of time series covered by the window. Ignoring the order in which the words are obtained, one thus obtains a “bag-of-words” representation of the time series (Lin et al 2012), a term borrowed from natural language processing and information retrieval. Such a higher-level structural representation of time series is more appropriate for computing similarity over long and noisy time series. Similarity between two time series thus represented can be measured in terms of their Euclidean distance or cosine similarity over their word histogram representations, similar to the way document similarity is computed in information retrieval (Manning et al 2008). Bag-of-words SAX can thus be used for time series classification using one-nearest neighbor classifier. Besides classification, SAX has been also employed for time series clustering and anomaly detection (Keogh et al 2005; Lin et al 2012). The method has attracted much attention and has also been used and improved in multiple ways (Shieh and Keogh 2008; Ordóñez et al 2011; Senin and Malinchik 2013).

In this paper, we first show that the performance of SAX can be improved on time series classification by using it with a better machine learning algorithm, like SVM, instead of 1NN. Unlike in the case of DTW, since SAX is already a feature-based representation where features are the words, this is

straightforward to do and forms a stronger baseline. We next show that when DTW features as proposed in this paper are used in combination with SAX features, the classification performance improves over either method when used alone. Thus the method is able to combine the strengths of SAX and DTW to give the best performance.

3 Using DTW as Features

Given that DTW works so well on time series classification task when used with the simple one-nearest neighbor algorithm, one would want to use DTW with a more advanced machine learning algorithm hoping to improve the performance. But in order to use other machine learning algorithms, one needs to either represent a time series as a vector of features or define a kernel between two time series. Given that DTW is defined between two time series, it cannot be computed as a feature for a time series using only that time series, hence it cannot be directly used as a feature. On the other hand, it is not clear how DTW could be used inside a kernel because a kernel matrix needs to be positive semi-definite so that the implicit feature space is well-defined. There have been attempts to define kernels directly using DTW (Gudmundsson et al 2008) but they obtained results inferior to 1NN-DTW. Although there have been plethora of approaches that use advanced feature-based or kernel-based machine learning methods for time series classification, but they do not exploit DTW’s strength on this task.

In this paper, we propose a feature-based representation of a time series in which each feature is its DTW distance from a training example. Besides features based on DTW, we also use features based on DTW-R and Euclidean distances, as well as various combinations of these features. Once the features are obtained, any machine learning algorithm can be used for classification. One can also use kernel-based methods where a kernel is defined in terms of the dot-product of these explicit features and is thus well-defined.

Formally, given a time series T , and the training data $D = \{Q_1, Q_2, \dots, Q_n\}$, the feature vector, $\text{Feature-DTW}(T)$, of T constructed using DTW will be simply: $\text{Feature-DTW}(T) = (DTW(T, Q_1), DTW(T, Q_2), \dots, DTW(T, Q_n))$. Note that T could be one of the training examples, in which case $\text{Feature-DTW}(T)$ will be its feature-based representation which will be given to the machine learning method during training.

Analogously, if DTW-R or Euclidean distance (ED) is used as the distance measure then we will get corresponding feature vectors $\text{Feature-DTW-R}(T)$ and $\text{Feature-ED}(T)$ respectively. We can combine features $\text{Feature-DTW}(T)$ and $\text{Feature-DTW-R}(T)$ by simply concatenating the two vectors into a larger vector which we denote as $\text{Feature-DTW-DTW-R}(T)$. Similarly, we can combine all three of them which we denote as $\text{Feature-ED-DTW-DTW-R}(T)$. In the rest of the paper, we use this notation to also represent the classifier that use these features, for example, Feature-DTW-R will mean the classifier that uses DTW-R based features.

The rationale for defining such features is as follows. Although both 1NN-DTW and Feature-DTW will get the same input for a query during testing: the DTW distances of the query from each of the training examples, but while 1NN-DTW will use a fixed “nearest neighbor” function over this input to decide the output label, Feature-DTW will use a complex function over this input (for example, a polynomial function with SVM) which it would have specifically tuned to maximize classification accuracy using the training data. For example, it may easily learn during training to down-weight a noisy training example even though it may be the nearest. It has been noted (Batista et al 2011) that some low-complexity time series (for example, an almost constant time series or a sine wave) are often close to everything and hence could easily mislead the nearest-neighbor based approach (Chen et al 2013). But on these cases as well, Feature-DTW may learn to down-weight such training examples. Feature-DTW thus searches for a more complex classification hypothesis than is used by 1NN-DTW and hence we expect Feature-DTW to work better than 1NN-DTW in general. The above holds for ED and DTW-R distances as well. Finally, by combining the features, the method will be able to search for a complex hypothesis in terms of all the included distances and may learn their relative importances to optimize classification accuracy for a particular dataset.

One can use any machine learning method with the features thus defined. We chose to use Support Vector Machines (SVMs) (Cristianini and Shawe-Taylor 2000) for experiments reported in this paper because it is known to work well even with large number of features. As mentioned before, kernel, k , between two examples is then simply defined as the dot-product of the two feature sets. For our experiments, we employed polynomial kernels with SVMs.

This method can be easily extended to work with any other time series classification method that uses features by simply concatenating DTW-based features to those features. In this paper, we demonstrate this by concatenating DTW-based features to SAX features. The machine learning algorithm can then learn from the training data which features are important for a particular dataset and thus combine the strengths of the two methods.

4 Related Work

Most of the previous work that has concentrated on designing features for time series classification has not considered using DTW distances as features. For example, while the method by Fulcher and Jones (2014) searches among thousands of various types of features, it does not include DTW as features. A method automatically constructs features through genetic programming in (Harvey and Todd 2012) but it does not construct DTW features. We however note that any features designed in a previous work can be readily combined with our proposed DTW features by simple concatenation of feature vectors.

There are very few research papers that have explored using the type of DTW-based features as defined in the previous section, and the ones that

have used it report performance worse than 1NN-DTW. The closest work to ours is by Gudmundsson et al (2008). While they employ DTW-based features and also use SVMs, however, there are many differences. Most importantly, they report that their best technique (ppfSVM-NDTW) performs worse than 1NN-DTW (it wins on only 8 of the 20 datasets, Table II of (Gudmundsson et al 2008)). And, in fact, 1NN-DTW itself performs worse than 1NN-DTW-R (Keogh et al 2011) which they do not even compare against. On the other hand, our Feature-DTW-DTW-R method outperforms 1NN-DTW-R on 31 of the 47 UCR datasets. Comparing their results with our results of Feature-DTW-DTW-R method (Table 4) directly, they win on only 1 out of the 20 datasets. We believe the difference in performance is for the following reasons. Instead of simply considering the dot-product of DTW-based features as defining the kernel (well-defined by definition) and then expanding the feature space using simple polynomial kernels as we do, they attempted at creating complex kernels that would be well-defined. They also did not use DTW-R for creating features which we found to contribute to the good performance more than DTW-based features. While they explored complex kernels, they did not explore combining features which we found to be very useful. They also do not show how their method can be used in combination with other time series classification methods.

DTW-based features were also used in different ways in a few other papers. In (Rodríguez and Alonso 2004), a method is used to select DTW features and then decision trees are used for classification. However, it does worse than 1NN-DTW on all the three UCR datasets they used. In (Hayashi et al 2005; Mizuhara et al 2006), the DTW-based features are first embedded in a different space using Laplacian eigenmap technique before classification. The authors did not use UCR datasets and report results on only two other datasets.

In contrast to the related work, in this paper, we present a very simple approach, which is also easily extensible, and through comprehensive experiments we show that it outperforms 1NN-DTW-R on two-third of all the available UCR datasets. In addition, our method can be used in combination with other time series classification methods. We show that when combined with the SAX method it improves over it. To the best of our knowledge, no work has tried to combine DTW and SAX.

5 Experiments

In Subsection 5.1 we describe the experiments in which we used DTW as features. In Subsection 5.2 we describe the experiments in which we combined our method with SAX. In each of these subsections, we first describe the experimental methodology which is followed by the results and its discussion. In Subsection 5.3 we compare all the classifiers together. In the last subsection we experiment with ensembles. Our code and our detailed results are available online for download (Kate 2014).

5.1 DTW as Features

5.1.1 Methodology

We used the UCR time series datasets (Keogh et al 2011) which have been widely used as a benchmark for evaluating time series classification methods. These datasets vary widely in their application domains, number of classes, time series lengths, as well as in sizes of the training and testing sets. Although gathered at UCR, many of these datasets were originally created at different places and are thus diverse. We implemented DTW and DTW-R using the simple dynamic programming algorithm. Since our focus was not on timing performance, we did not use any faster algorithm. We used the exact same values for the window-size parameter, r , for DTW-R computations for different datasets as were reported on the web-page (Keogh et al 2011) to be the best values. These were found by search over the training sets. Besides in 1NN-DTW-R, we used the same r values in all the feature based classifiers that used DTW-R features. We used the same training and testing sets as were provided in the datasets. The classification error rates we obtained using our 1NN-DTW, 1NN-DTW-R and 1NN-ED implementations perfectly matched the ones reported on the UCR time series datasets web-page on all the datasets.

For all feature-based methods, we used LIBSVM (Chang and Lin 2011) implementation for SVMs with its default setting (type of SVM as C-SVC, misclassification penalty parameter C as 1, and type of kernel as polynomial). We chose to use its default setting because we did not want to make the results too dependent on parameter values. However, given how SVM’s accuracy depends on the degree of the polynomial kernel, we used three different values for the degree parameter making the polynomial kernel linear, quadratic and cubic. If a lower degree polynomial is used than needed then the performance can suffer from under-fitting, and on the other hand, if a higher degree is used than needed then it can over-fit. We defined our polynomial kernels of degree n as $(1 + k)^n$ where k is the dot-product of the features. Note that this is how polynomial kernels are defined in Weka (Hall et al 2009) which does not require any other parameter besides the degree. In all our results, the best degree for a dataset was determined using a ten-fold cross-validation within the training set and ties were resolved in favor of the smaller degree. All our results were obtained using this setting for SVMs. We also tried Gaussian kernel with its default parameter values, however its performance was very poor in comparison, whether on the internal cross-validation of the training set or on the test set. It would perhaps need parameter tuning to get reasonable results.

In our results, for every comparison between two classifiers, we also report its statistical significance. Given that these comparisons belong to the “two classifier on multiple domains” category, we used Wilcoxon signed-rank test which is the recommended statistical significance test for such comparisons³ (Japkowicz and Shah 2011). Besides considering which system did better on a

³ We compare all classifiers together using Friedman test in Subsection 5.3.

Comparison	Win/Loss/Tie
Feature-ED vs. 1NN-ED	35/12/0
Feature-DTW vs. 1NN-DTW	27/18/2
Feature-DTW-R vs. 1NN-DTW-R	26/20/1

Table 1 Comparing Euclidean (ED), DTW and DTW-R used as distance measures with one-nearest neighbor against using them respectively as features with SVM as the learning method. They were compared on the 47 UCR time series classification datasets. The results shown in bold were found to be statistically significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test.

dataset, this test also takes into account the performance difference between them on the dataset. All our main results were found to be statistically significant. However, for the less important results, and in general, we want to point out that while low p-values for statistical significance should be interpreted as sufficient to reject the null hypothesis (and hence conclude that the two classifiers are different), however, high p-values should not be interpreted as we accept the null hypothesis (and thus conclude that the two classifiers are not different) (Vickers 2010). In the latter case, all we can say is that the data is not sufficient to reject the null hypothesis and consider only the actual performance numbers.

5.1.2 Results and Discussion

We first show that using a distance measure to create features, as proposed in this paper which enables the use of SVM, reduces the classification errors over using them directly with 1NN. Table 1 shows the win/loss/tie numbers for the comparisons for each of the three distance measures, Euclidean, DTW and DTW-R. In our results we are showing only these three numbers for most of the comparisons because it is difficult to show individual errors on every dataset for so many comparisons. We give those details only for the most important comparisons later in Table 4 (but all the detailed results are available on our website (Kate 2014)). In all our one-to-one comparison tables, we show in bold all the results which were found to be statistical significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test. In Table 1 it is interesting to note that even Euclidean distance when used as features with SVM significantly improves the performance. We also found that Feature-ED performs comparable to 1NN-DTW (23/24/0), however it lags behind 1NN-DTW-R (17/29/1). Both Feature-DTW and Feature-DTW-R perform better than their respective distance measures used with 1NN.

Next, we show how the classifier accuracies are affected by combining various features. Table 2 shows one-to-one comparisons between various classifiers. Each cell in the table shows on how many datasets did the method in its row win/lose/tie over the method in its column. Symmetric duplicate comparisons have not been shown for clarity. The first two rows show the familiar results that 1NN-DTW outperforms 1NN-ED (30/15/2) and 1NN-DTW-R

Method	1NN-ED	1NN-DTW	1NN-DTW-R	Feature-DTW	Feature-DTW-R	Feature-DTW-DTW-R
1NN-DTW	30/15/2	-				
1NN-DTW-R	31/5/11	31/13/3	-			
Feature-DTW	35/11/1	27/18/2	24/23/0	-		
Feature-DTW-R	39/7/1	29/18/0	26/20/1	28/16/3	-	
Feature-DTW-DTW-R	42/5/0	34/11/2	31/15/1	33/6/8	29/13/5	-
Feature-ED-DTW-DTW-R	42/5/0	32/14/1	29/16/2	32/12/3	32/13/2	25/18/4

Table 2 Various comparisons between Euclidean (ED), DTW and DTW-R when used as distance measures with one-nearest neighbor and when using them as features in different combinations with SVM as the learning method. All comparisons were done using the 47 UCR time series classification datasets. Each cell in the table shows on how many datasets did the classifier in its row win/lose/tie over the classifier in its column. The symmetric duplicate comparisons have been omitted for clarity. The results shown in bold were found to be statistically significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test.

outperforms 1NN-DTW (31/13/3) and also adds the information that these are statistically significant.

The next two rows show what Table 1 already showed, that Feature-DTW improves over 1NN-DTW and Feature-DTW-R improves over 1NN-DTW-R. In addition, it may be noted that Feature-DTW-R outperforms Feature-DTW (28/16/3). Hence DTW-R, the window-size constrained DTW, is a better distance measure than DTW even when they are used as features. However, the next row shows an interesting result that when both these types of features are combined, Feature-DTW-DTW-R significantly outperforms both Feature-DTW and Feature-DTW-R classifiers individually. And hence not surprisingly, it outperforms 1NN-DTW-R with a wider margin (31/15/1) than either Feature-DTW or Feature-DTW-R. This was also found to be statistically significant.

We consider this result that Feature-DTW-DTW-R outperforms 1NN-DTW-R on 31 of the 47 datasets as an important result of this paper. It may be noted that this performance difference is similar in magnitude with which DTW outperforms Euclidean distance in the one-nearest setting, and hence is very respectable. We further looked into this result. While Feature-DTW-R does better than Feature-DTW on 28 datasets, there are 16 datasets on which Feature-DTW does better than Feature-DTW-R, and we found that on 15 of those datasets Feature-DTW-DTW-R also does better than Feature-DTW-R (the remaining dataset ties). As Feature-DTW-DTW-R does better than Feature-DTW-R on 29 datasets out of the 47 datasets, it is very unlikely that all 15 datasets will fall on the “win” side by chance. This thus indicates that whatever advantage DTW features have over DTW-R features on a few datasets, by combining the two types of features the classifier is able to avail that advantage. Correspondingly, Feature-DTW-R does better than Feature-DTW on 28 datasets and 26 of these are also the datasets on which Feature-DTW-DTW-R does better than Feature-DTW (which it does so on total 33). This is also very unlikely to happen by chance, hence Feature-DTW-DTW-R is able to combine the advantages of both DTW and DTW-R distances. In

Polynomial Kernel	Win/Loss/Tie
Linear	28/17/2
Quadratic	31/16/0
Cubic	26/20/1
Internal cross-validation	31/15/1

Table 3 Comparing Feature-DTW-DTW-R and 1NN-DTW-R by varying the degree of the polynomial kernel used in SVM. The comparison was done using the 47 UCR time series classification datasets. The results shown in bold were found to be statistically significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test.

other words, the classifier is able to learn which features work better for which datasets and hence is giving an overall better performance.

The last row of Table 2 shows that adding Euclidean distance as additional features does not improve by much over the improvements already obtained by Feature-DTW-DTW-R although it does a little better over it on one-to-one comparison (25/18/4).

Since degree of the polynomial kernel used with SVM was our only parameter, we report in Table 3 how its value affects the comparison between Feature-DTW-DTW-R and 1NN-DTW-R. In the last row we report again the comparison when the degree is determined through internal cross-validation. The remaining rows show the results when different degrees are used. An important thing to note is that Feature-DTW-DTW-R outperforms 1NN-DTW-R with all the three degree values. It, however, does best with degree of 2.

We give further details on the comparison between 1NN-DTW-R and Feature-DTW-DTW-R in Table 4 where their classification errors on each of the 47 UCR datasets are reported in sixth and seventh columns respectively (the next three columns will be explained in the following subsections). The r values for 1NN-DTW-R⁴ and the degree of the polynomial kernel used in Feature-DTW-DTW-R (as determined by an internal ten-fold cross-validation within the training set) are also shown in brackets for each dataset. It may be noted that linear kernel was found to be sufficient for most of the datasets. Figure 1 visually shows the classification errors obtained by these two classifiers. Note that the points above the diagonal line are the datasets on which Feature-DTW-DTW-R obtained lower error and the points below the line are the datasets on which 1NN-DTW-R obtained lower error.

Given that our method creates number of features proportional to the number of training examples, one may wonder whether very large number of training examples will lead to a difficult learning problem due to a correspondingly very large feature space. To look into this issue, we plotted learning curves in Figures 2 and 3 respectively for the two datasets NonInvasiveFatalECG_Thorax1 and NonInvasiveFatalECG_Thorax2, which have the largest number of training examples (1800 each) out of the 47 UCR datasets. The curves are shown for the classifiers Feature-DTW-DTW-R and 1NN-DTW-R.

⁴ The same r values were also used in Feature-DTW-DTW-R as well as in other feature-based classifiers that used DTW-R.

Dataset	Number of Classes	Size of training set	Size of testing set	Time series length	1NN-DTW-R (r)	Feature-DTW-DTW-R (degree)	SVM-SAX (n, w, a, degree)	Feature-SAX-DTW-DTW-R (degree)	Ensemble-final
50words	50	450	455	270	0.242 (6)	0.255 (3)	0.374 (128,8,3,1)	0.246 (3)	0.211
Adiac	37	390	391	176	0.391 (3)	0.325 (3)	0.366 (32,8,9,3)	0.315 (3)	0.263
Beef	5	30	30	470	0.467 (0)	0.533 (1)	0.4 (24,4,7,2)	0.333 (1)	0.433
Car	4	60	60	577	0.233 (1)	0.15 (1)	0.117 (160,8,9,1)	0.2 (2)	0.15
CBF	3	30	900	128	0.004 (11)	0 (1)	0.0489 (16,4,3,3)	0.003 (1)	0
ChlorineConcentration	3	467	3840	166	0.35 (0)	0.273 (2)	0.342 (8,4,5,3)	0.348 (3)	0.28
CinC.ECG.torso	4	40	1380	1639	0.064 (1)	0.101 (1)	0.104 (16,4,3,1)	0.01 (1)	0.046
Coffee	2	28	28	286	0.179 (3)	0.071 (1)	0.143 (16,4,5,1)	0.107 (1)	0.107
Cricket_X	12	390	390	300	0.236 (7)	0.226 (2)	0.426 (80,4,6,2)	0.208 (1)	0.179
Cricket_Y	12	390	390	300	0.197 (17)	0.159 (1)	0.346 (112,8,3,1)	0.179 (1)	0.133
Cricket_Z	12	390	390	300	0.179 (7)	0.185 (2)	0.39 (128,4,5,2)	0.218 (1)	0.156
DiatomSizeReduction	4	16	306	345	0.065 (0)	0.046 (1)	0.147 (32,4,3,1)	0.105 (1)	0.049
ECG200	2	100	100	96	0.12 (0)	0.09 (3)	0.18 (40,8,4,2)	0.17 (1)	0.12
ECGFiveDays	2	23	861	136	0.203 (0)	0.144 (1)	0.014 (40,8,6,1)	0.093 (2)	0.056
FaceAll	14	560	1690	131	0.192 (3)	0.138 (3)	0.246 (40,8,3,1)	0.193 (2)	0.166
FaceFour	4	24	88	350	0.114 (2)	0.136 (1)	0 (64,8,3,1)	0.034 (1)	0.045
FacesUCR	14	200	2050	131	0.088 (12)	0.117 (1)	0.16 (32,8,3,1)	0.1 (1)	0.093
Fish	7	175	175	463	0.16 (4)	0.171 (1)	0.057 (128,8,7,1)	0.069 (1)	0.063
Gun_Point	2	50	150	150	0.087 (0)	0.047 (2)	0.02 (32,4,9,2)	0 (1)	0.013
Haptics	5	155	308	1092	0.588 (2)	0.545 (2)	0.562 (144,8,4,2)	0.539 (1)	0.477
InlineSkate	7	100	550	1882	0.611 (14)	0.682 (1)	0.885 (8,4,6,2)	0.6 (2)	0.598
ItalyPowerDemand	2	67	1029	24	0.045 (0)	0.037 (1)	0.092 (16,4,4,2)	0.035 (1)	0.04
Lighting2	2	60	61	637	0.131 (6)	0.115 (2)	0.246 (128,8,4,1)	0.18 (1)	0.066
Lighting7	7	70	73	319	0.288 (5)	0.233 (1)	0.397 (128,4,3,1)	0.233 (1)	0.164
MALLAT	8	55	2345	1024	0.086 (0)	0.035 (1)	0.293 (72,4,6,1)	0.076 (1)	0.038
MedicalImages	10	381	760	99	0.253 (20)	0.262 (1)	0.474 (64,4,8,2)	0.297 (1)	0.242
MoteStrain	2	20	1252	84	0.134 (1)	0.107 (1)	0.506 (16,8,6,1)	0.162 (1)	0.075
NonInvasiveFatalECG_Thorax1	42	1800	1965	750	0.189 (1)	0.091 (2)	0.243 (160,4,5,1)	0.117 (1)	0.082
NonInvasiveFatalECG_Thorax2	42	1800	1965	750	0.125 (1)	0.078 (1)	0.131 (160,8,8,1)	0.081 (1)	0.067
OliveOil	4	30	30	570	0.167 (1)	0.267 (3)	0.133 (32,4,7,1)	0.133 (1)	0.1
OSULeaf	6	200	242	427	0.384 (7)	0.343 (3)	0.223 (80,8,5,1)	0.227 (2)	0.236
Plane	7	105	105	144	0 (5)	0.01 (1)	0 (16,4,7,1)	0 (1)	0
SonyAIBORobot_Surface	2	20	601	70	0.304 (0)	0.142 (3)	0.163 (48,8,3,1)	0.144 (1)	0.23
SonyAIBORobot_SurfaceII	2	27	953	65	0.141 (0)	0.22 (1)	0.273 (16,4,8,1)	0.193 (1)	0.14
StarLightCurves	3	1000	8236	1024	0.095 (16)	0.049 (3)	0.042 (160,8,3,3)	0.035 (2)	0.029
SwedishLeaf	15	500	625	128	0.155 (2)	0.101 (2)	0.144 (32,8,6,1)	0.09 (1)	0.09
Symbols	6	25	995	398	0.062 (8)	0.062 (1)	0.057 (48,8,5,1)	0.027 (1)	0.037
Synthetic.control	6	300	300	60	0.017 (6)	0.013 (1)	0.013 (40,8,3,1)	0.01 (1)	0.003
Trace	4	100	100	275	0.01 (3)	0 (1)	0 (40,4,4,1)	0 (1)	0
TwoLeadECG	2	23	1139	82	0.116 (6)	0.144 (1)	0.076 (16,8,8,1)	0.078 (1)	0.079
Two_Patterns	4	1000	4000	128	0.002 (4)	0 (1)	0.007 (96,8,3,3)	0 (1)	0
uWaveGestureLibrary_X	8	896	3582	315	0.227 (4)	0.212 (1)	0.765 (160,4,6,2)	0.201 (3)	0.18
uWaveGestureLibrary_Y	8	896	3582	315	0.301 (4)	0.286 (1)	0.373 (160,4,3,3)	0.317 (1)	0.266
uWaveGestureLibrary_Z	8	896	3582	315	0.322 (6)	0.28 (1)	0.335 (160,4,3,3)	0.283 (1)	0.259
Wafer	2	1000	6174	152	0.005 (1)	0.006 (3)	0.009 (80,8,7,1)	0.004 (2)	0.006
WordsSynonyms	25	267	638	270	0.252 (8)	0.34 (1)	0.458 (160,8,3,1)	0.323 (3)	0.276
Yoga	2	300	3000	426	0.155 (2)	0.141 (3)	0.161 (112,8,6,3)	0.131 (3)	0.122
# Wins					3	10	7	11	27

Table 4 The 47 UCR time series datasets and the classification errors by the classifiers 1NN-DTW-R, Feature-DTW-DTW-R, SVM-SAX, Feature-SAX-DTW-DTW-R and Ensemble-final. The error rate of the best classifier for each dataset is shown in bold. The value of parameter r used in Feature-DTW-DTW-R and Feature-SAX-DTW-DTW-R was the same as in 1NN-DTW-R for each dataset. The values of parameters n , w and a used in Feature-SAX-DTW-DTW-R were same as in SVM-SAX for each dataset. One-to-one direct comparisons between the classifiers in terms of win/loss/tie numbers are shown in Tables 2, 5 and 7.

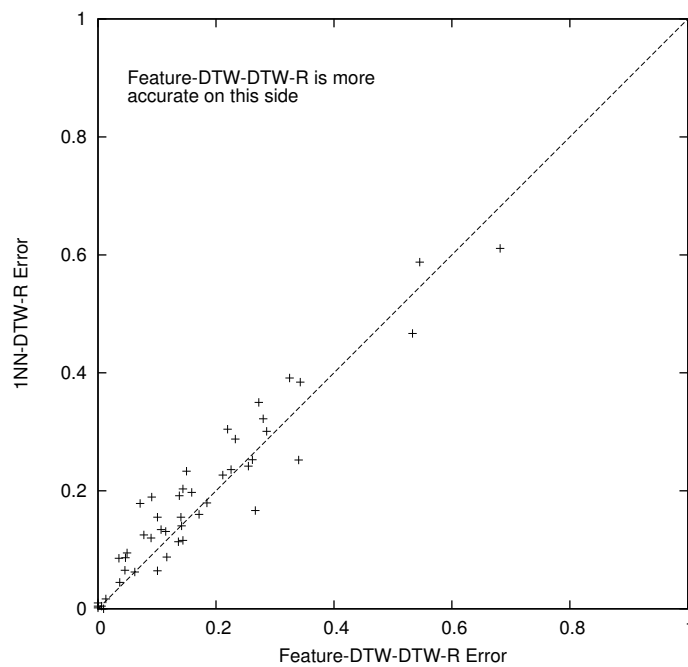


Fig. 1 Comparing classification errors of the classifiers Feature-DTW-DTW-R and 1NN-DTW-R on the 47 UCR time series classification datasets. Points above the diagonal line are the datasets on which Feature-DTW-DTW-R obtained lower error and points below the line are the datasets on which 1NN-DTW-R obtained lower error.

To plot the curves, we increased the number of training examples and measured percent accuracies on the same test data. We plotted percent accuracies instead of errors so that the learning curves would look familiar with their upward trends. For both the datasets, it may be noted that with around 600 examples Feature-DTW-DTW-R starts outperforming 1NN-DTW-R's best performance with all the 1800 examples. This shows that all training examples may not be needed to outperform 1NN-DTW-R. For datasets with very large number of training examples, another possibility is to simply restrict the size of the training examples used to create DTW-based features. Thus one may use all the training examples for training, but use only a sample of them for creating features (note that for plotting the learning curves we used reduced number of training examples for both creating features and for training).

Figure 2 shows an interesting criss-crossing of the curves early on indicating that Feature-DTW-DTW-R may need some minimum number of training examples before it outperforms 1NN-DTW-R. This is not surprising since it is learning a more complex hypothesis. However, the same trend is not visible in Figure 3 and from Table 4 it may be noted that Feature-DTW-DTW-R outperforms 1NN-DTW-R even on datasets with very small number of training examples.

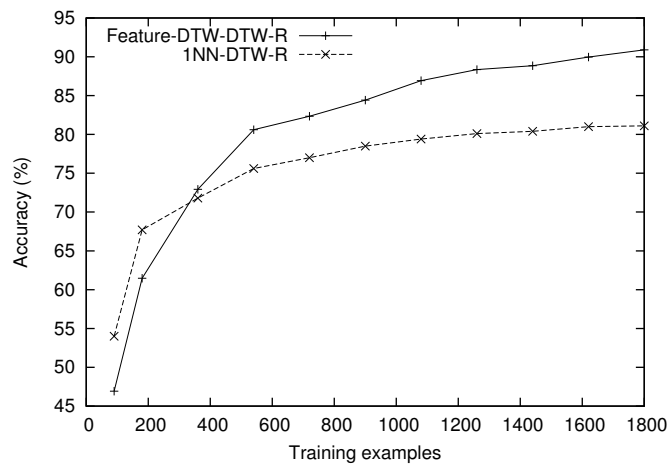


Fig. 2 Learning curves for accuracy of the classifiers Feature-DTW-DTW-R and 1NN-DTW-R on the dataset NonInvasiveFatalECG_Thorax1.

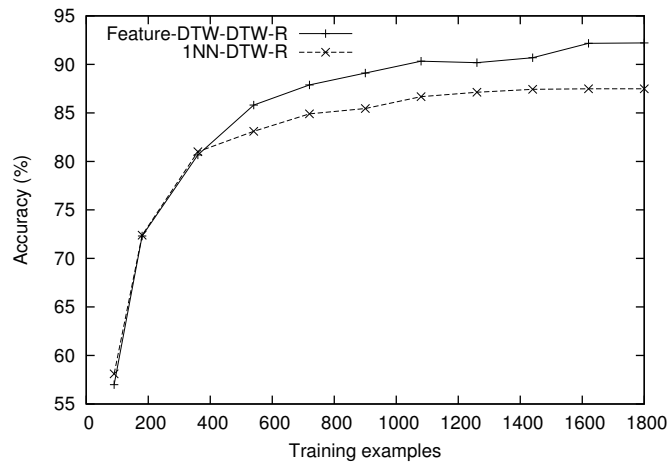


Fig. 3 Learning curves for accuracy of the classifiers Feature-DTW-DTW-R and 1NN-DTW-R on the dataset NonInvasiveFatalECG_Thorax2.

When users have a new time series test set with unknown labels and they want to use a classifier to predict labels, then in order to choose one classifier over another, it is not sufficient to know that one classifier does better than another on multiple other datasets. What a user really needs is the ability to predict ahead of time which classifier will do better on their particular dataset. Simply showing that a new classifier is better on multiple datasets and hoping that a new dataset will be one of them is falling for a subtle version of the Texas sharpshooter fallacy (Batista et al 2011). In order to avoid this, Batista et al (2011) recommend plotting expected accuracy gain of one classifier over another (obtained by cross-validation on the training data)

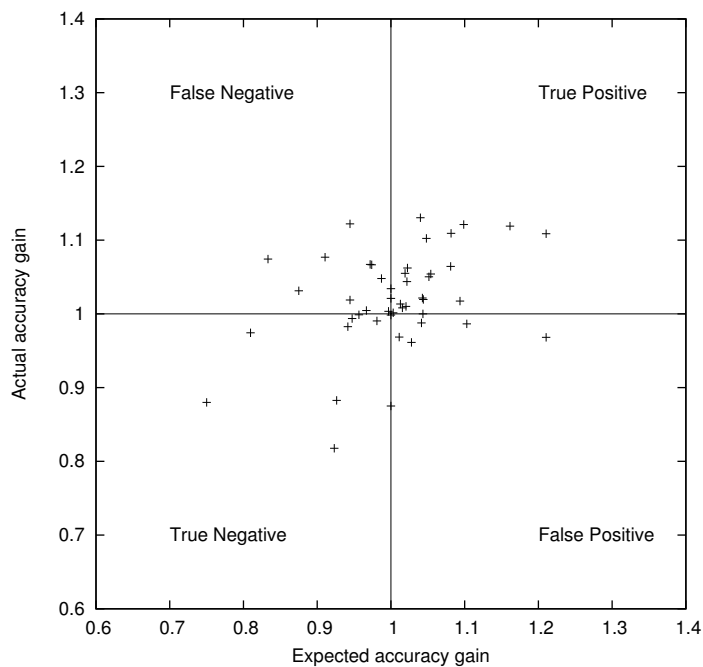


Fig. 4 Texas sharpshooter plot showing expected accuracy gain calculated on the training data versus actual accuracy gain on the test data if Feature-DTW-DTW-R classifier is used instead of 1NN-DTW-R classifier over the 47 UCR time series classification datasets.

versus actual accuracy gain (obtained on the test data) for each of the datasets. Accuracy gain is defined as the ratio of the accuracies of the two classifiers. If the classifier which is expected to do better actually does better on many datasets then we can say that we will be able to reliably predict the better classifier on a new dataset.

In Figure 4 we plot expected and actual accuracy gains of Feature-DTW-DTW-R classifier over 1NN-DTW-R classifier for the 47 UCR datasets. Expected accuracy gain for each dataset was computed by dividing the ten-fold cross-validation accuracy obtained by Feature-DTW-DTW-R on the training set by the ten-fold cross-validation accuracy obtained by 1NN-DTW-R on the training set. As before, we used the exact same values for the window-size parameter, r , for DTW-R computations (for both the classifiers) as were reported on the web-page (Keogh et al 2011) to be the best values. For the Feature-DTW-DTW-R classifier, the best cross-validation accuracy obtained by trying linear, quadratic and cubic degrees of the polynomial kernel was used for each dataset. The exact same degree and r value were later used for computing accuracy on the test set for each dataset. The actual accuracy gain for each dataset was computed by dividing the accuracy obtained by Feature-DTW-DTW-R on the test set by the accuracy obtained by 1NN-DTW-R on the test set.

The plot in Figure 4 shows four regions which are marked according to the labels of a contingency table and are interpreted as follows. In the true positive region, we claimed from the training data that Feature-DTW-DTW-R will give better accuracy on the test data and we were correct. This happened for 19 datasets. In the true negative region, we correctly claimed that the new classifier will decrease accuracy, this happened for 8 datasets. In the false negative region, although we claimed that the new classifier will decrease accuracy, it actually increased. This happened for 10 datasets. These cases represent missed opportunity of gaining accuracy using the new classifier. Finally, in the false positive region we claimed that the new classifier will do better but it did worse. This is the worst and the only scenario in which we end up losing accuracy because of using the new classifier. This happened for 5 datasets (CinC_ECG_torso, FacesUCR, MedicalImages, TwoLeadECG and fish). The worst of these datasets was TwoLeadECG on which we were expecting a big 21% improvement in accuracy but we ended up losing 4%. This could be because that dataset has only 23 training examples which may be insufficient for a good prediction. Five datasets fell on the borders of the regions. We note that out of the four regions, the true positive region got the most datasets and the false positive region got the fewest ones. From the plot we can say that whenever we predicted that Feature-DTW-DTW-R will do better than 1NN-DTW-R, it did so for most of the datasets (19 out of 24 datasets, ignoring the border cases).

5.2 Combining with SAX

5.2.1 Methodology

In order to demonstrate that our method of using DTW distances as features can be easily combined with other methods of time series classification, we show experimental results of combining it with the SAX method (Lin et al 2012). While the code for computing SAX was available,⁵ we could not find code for computing bag-of-words representation for time series. Hence we implemented it ourselves. But even with the same parameter values we could not replicate the 1NN-SAX classification results of (Lin et al 2012) for 17 of the 20 UCR datasets for which they present results. Perhaps this is because of differences in implementations details. Hence we refer to our results as 1NN-SAX and their results as Lin-1NN-SAX. Since they obtained their best results using the Euclidean distance to compute nearest neighbors (Table 7 of (Lin et al 2012)), we only compared with their results for Euclidean distance and also used Euclidean distance in our 1NN-SAX. We used our own internal five-fold cross-validation within the training data to find the best parameter values for n (out of 8, 16, 24, ..., 160), w (out of 4 and 8) and a (out of 3, 4, 5, ..., 9). These possible values of the parameters form a superset of the parameter values from (Lin et al 2012). Once the best parameter values were found for

⁵ <http://www.cs.gmu.edu/~jessica/sax.htm>

Method	1NN-DTW-R	Feature-DTW-DTW-R	Lin-1NN-SAX	1NN-SAX	SVM-SAX
Feature-DTW-DTW-R	31/15/1	-			
Lin-1NN-SAX	9/11/0	8/11/1	-		
1NN-SAX	12/35/0	8/38/1	9/11/0	-	
SVM-SAX	19/27/1	12/32/2	12/7/1	36/8/3	-
Feature-SAX-DTW-DTW-R	33/13/1	26/18/3	13/6/1	41/3/3	37/7/3

Table 5 Various comparisons between SAX representation used with one-nearest neighbor (including original results from Lin et al. (2012), Lin-1NN-SAX) and when using SAX features by themselves (SVM-SAX) and in combination with DTW-DTW-R features (Feature-SAX-DTW-DTW-R) with SVM as the learning method. Results for 1NN-DTW-R and Feature-DTW-DTW-R are also shown for comparison. All comparisons were done using the 47 UCR time series classification datasets except the comparisons with (Lin et al 2012) which had results for only 20 UCR datasets. Each cell in the table shows on how many datasets did the classifier in its row win/lose/tie over the classifier in its column. The symmetric duplicate comparisons have been omitted for clarity. The results shown in bold were found to be statistically significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test.

1NN-SAX for each of the datasets, for fair comparisons we used the same values in other classifiers that used SAX representation. For the classifiers that used SVMs, the best degree of the polynomial kernel (linear, quadratic or cubic) was determined by cross-validation within the training data.

Besides using 1NN, we used SVM with SAX bag-of-words feature representation which we call SVM-SAX classifier. This is a stronger baseline than 1NN-SAX. We used LIBSVM (Chang and Lin 2011) implementation for SVMs with its default values as before. Finally, we added DTW and DTW-R features to SAX features to combine these methods which we call Feature-SAX-DTW-DTW-R classifier that also used SVM. The value for the parameter r was same as was used in 1NN-DTW-R and obtained from (Keogh et al 2011). Note that for a time series, its SAX features are computed using only that time series, while its DTW and DTW-R features are computed as its distances from the training examples.

5.2.2 Results and Discussion

Table 5 shows the comparisons between SAX methods when SAX features are used by themselves and in combination with DTW and DTW-R features. From the third row, it may be noted that our 1NN-SAX implementation is comparable to that of Lin-1NN-SAX (Lin et al 2012) with win/loss/tie numbers of 9/11/0. However, both classifiers do worse than 1NN-DTW-R and Feature-DTW-DTW-R as can be seen from the first two columns. From the fourth row, it can be seen that SVM-SAX significantly outperforms 1NN-SAX (36/8/3) which is not surprising given that SVM is a more powerful learning algorithm than 1NN in general. Finally, the last row shows that Feature-SAX-DTW-DTW-R significantly outperforms even SVM-SAX (37/7/3) and also improves on Feature-DTW-DTW-R (26/18/3). This shows that the proposed method of using DTW as features can be successfully combined with SAX.

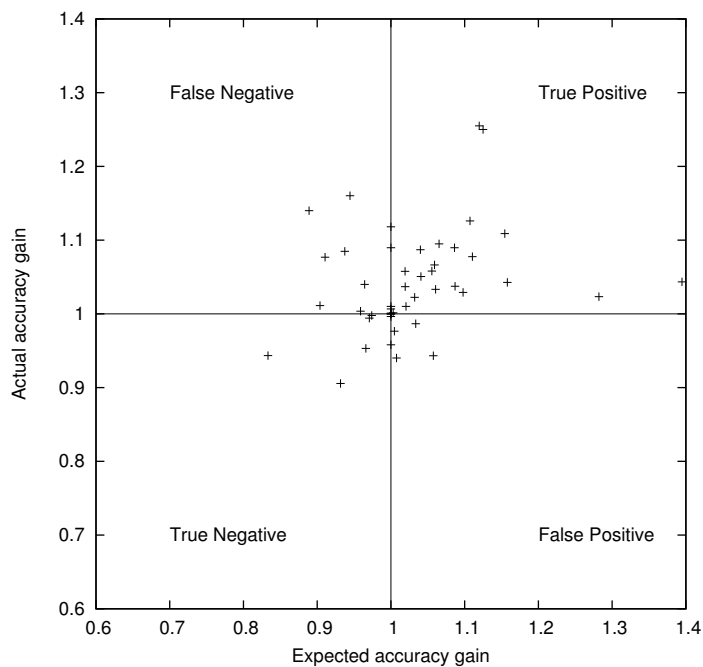


Fig. 5 Texas sharpshooter plot showing expected accuracy gain calculated on the training data versus actual accuracy gain on the test data if Feature-SAX-DTW-DTW-R classifier is used instead of 1NN-DTW-R classifier over the 47 UCR time series classification datasets.

Also noteworthy is that the combined method Feature-SAX-DTW-DTW-R outperforms 1NN-DTW-R by a wider margin (33/13/1) than Feature-DTW-DTW-R (31/15/1) (although both outperformed statistically significantly). We point out that Feature-SAX-DTW-DTW-R is our best non-ensemble classifier reported in this paper which is clearly because it is utilizing SAX bag-of-words features and DTW distances together with a powerful machine learning method of SVM.

Table 4 includes the errors obtained by SVM-SAX and Feature-SAX-DTW-DTW-R on all the 47 UCR datasets. We looked into on which datasets which types of features contribute in improving the performance. We found that on the 12 datasets on which SVM-SAX does better than Feature-DTW-DTW-R, on 11 of those Feature-SAX-DTW-DTW-R also does better than Feature-DTW-DTW-R (which it does on total 26 datasets). Correspondingly, on the 33 datasets on which Feature-DTW-DTW-R does better than SVM-SAX, on 32 of those Feature-SAX-DTW-DTW-R also does better than SVM-SAX (which it does on total 37 datasets). Given that it is very unlikely to obtain such numbers by chance, as in Subsection 5.1.2, we conclude that when combined sets of features are given to the method it is able to learn which features work best for which datasets and thus combine their strengths.

In Figure 5 we plot expected and actual accuracy gains of Feature-SAX-DTW-DTW-R classifier over 1NN-DTW-R classifier for the 47 UCR datasets. The same methodology as described in Subsection 5.1.2 was used to obtain this plot. For Feature-SAX-DTW-DTW-R classifier, the same values for the SAX parameters were used as were obtained before by cross-validation over the training sets by the 1NN-SAX classifier. In this plot, 22 datasets fell in the true positive region, 5 in the true negative region, 7 in the false negative region and 4 in the false positive region (these were ECG200, FacesUCR, MedicalImages and uWaveGestureLibrary_Y). The remaining 9 classifiers fell on the borders between the regions, many of them close to point (1,1) which are not of interest. As before, out of the four regions, the true positive region got the most datasets, on these we predicted correctly that the new classifier will do better. The false positive region got the fewest datasets, on these we predicted that the new classifier will do better but it did worse. From the plot we can say that whenever we predicted that Feature-SAX-DTW-DTW-R will do better than 1NN-DTW-R, it did so for most of the datasets (22 out of 26 datasets, ignoring the border cases).

Although timing performance was not the focus of this work, Table 6 shows the training and testing times (without internal cross-validation) taken by various classifiers on all 47 UCR time series classification datasets together. The classifiers were run on a server with Intel Xeon X5675 3 GHz processor and 2GB RAM. The table shows the CPU user times in minutes (m) and seconds (s). For DTW computations, we used the basic quadratic time dynamic programming implementation and did not employ any speed-up techniques. It can be seen from the first four rows of the table that the testing times of Feature-DTW-R and Feature-DTW are only a few minutes more than the testing times of 1NN-DTW-R and 1NN-DTW respectively. Both 1NN and feature-based classifiers require the same $O(st)$ number of DTW computations for s number of training examples and t number of testing examples. Hence from the table it is clear that during testing the feature-based classifiers spent most of the time in computing the distances and the overhead of using them as features with SVM was only a few minutes. However, there is a big difference in the training times of these classifiers. While 1NN classifiers require zero time for training because the training of 1NN involves simply storing all the training examples, but the feature-based classifiers require $O(s^2)$ DTW computations during training for s training examples. But we want to point out that for many applications training can be done off-line in advance and often it is the testing time which is more critical than the training time. Additionally, as pointed out earlier using learning curves, not all training examples may be needed for creating DTW features which can also help reduce the computational time of training.

It may be noted that Feature-DTW-R and Feature-DTW classifiers took more time to test than to train because many of the UCR datasets have a lot more testing examples than training examples and thus correspondingly requiring more DTW computations during testing than during training. Also, for all the classifiers including 1NN-DTW-R and 1NN-DTW, most of the

Classifier	Training time	Testing time
1NN-DTW-R	0	417m 44s
Feature-DTW-R	103m 15s	421m 10s
1NN-DTW	0	1370m 44s
Feature-DTW	480m 13s	1379m 17s
Feature-DTW-DTW-R	585m 41s	1783m 23s
SVM-SAX	1m 07s	2m 56s
Feature-SAX-DTW-DTW-R	583m 26s	1829m 24s
Feature-SAX-DTW-DTW-R (pre-computed distances)	2m 15s	5m 58s

Table 6 Training and testing times of various classifiers on all 47 UCR time series classification datasets. The times are shown in minutes (m) and seconds (s).

time was taken by the three largest datasets (StarLightCurves, NonInvasiveFetalECG_Thorax1 and 2). It may be noted that any DTW speed-up technique would have improved time taken by all 1NN and feature-based classifiers.

The fifth row of Table 6 shows that the training and testing times of Feature-DTW-DTW-R classifier is very close to the sums of the corresponding times of Feature-DTW and Feature-DTW-R classifiers. This is mainly because this classifier requires both DTW and DTW-R computations. In our implementation, we compute them separately but it may be possible to compute these two distances together avoiding some duplicate computations and thus reduce the time complexity perhaps close to that of DTW computation. The next row shows that SVM-SAX is extremely fast and requires only a few minutes to train as well as to test (we had employed hash tables for SAX features). The training and testing times of Feature-SAX-DTW-DTW-R classifier shown in the next row are close to that of Feature-DTW-DTW-R.

Finally, the last row shows that if we pre-compute all the DTW distances, Feature-SAX-DTW-DTW-R classifier takes only a few minutes to both train and test. Hence our proposed feature-based methods that use SVMs do not add significant computational time over what is needed for DTW computations. However, with longer time series and with more examples, the computational times for DTW and SVM will be the bottleneck in scaling it up. But we want to point out that any method that improves efficiency of DTW computations or any faster machine learning algorithm that scales with the data can be directly used by our method.

5.3 Comparing All Classifiers

In the previous subsections we showed results in which we compared two classifiers at a time and tested for their statistical significance using the Wilcoxon signed-rank test. In this section, we show results comparing all the classifiers together. We compared total thirteen classifiers that included baseline classifiers (1NN-ED, 1NN-DTW, 1NN-DTW-R, 1NN-SAX and SVM-SAX), their feature-based versions (Feature-ED, Feature-DTW and Feature-DTW-

R) and the combinations of their feature-based versions and SAX (Feature-DTW-DTW-R, Feature-ED-DTW-DTW-R, Feature-SAX-DTW-R, Feature-SAX-DTW-DTW-R, Feature-ED-SAX-DTW-DTW-R).

Friedman test is the recommended statistical test for comparing multiple classifiers on multiple datasets (Demšar 2006; Japkowicz and Shah 2011). To compute Friedman statistic, we first computed rank of each of the thirteen classifiers on each of the 47 datasets. In case of ties, the average of the tied ranks was assigned to each of the tied classifiers. For k classifiers and N datasets, if R_j denotes the average rank of the j th classifier over the N datasets, then Friedman statistic, χ_F^2 , is computed as:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \quad (3)$$

Demšar (2006) points out that χ_F^2 is undesirably conservative and recommends the following better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \quad (4)$$

Our value for the above statistic came out to be 16.72 which is far above the critical value of 2.22 (for $p < 0.01$ from F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom). Hence we can conclude that there is a significant difference among the classifiers on the 47 datasets.

We also performed Nemenyi post-hoc tests to compare all the classifiers. Figure 6 plots all the thirteen classifiers according to their average ranks on the 47 UCR datasets. In the figure, for the sake of shortness we have used prefix ‘‘F’’ instead of prefix ‘‘Feature’’ to denote the corresponding classifiers. The 1NN-ED classifier was the worst performing with an average rank of 10.35 and Feature-SAX-DTW-DTW-R was the best performing with an average rank of 4.18. We group the classifiers which showed no statistical difference in average ranks among themselves at $p < 0.05$ level using the Nemenyi test (critical difference of 2.66 for thirteen classifiers). The groups are shown in the figure with thick horizontal lines. It can be seen that the top performing group consists of all the classifiers that used DTW-R distances as features. The lowest group included the baseline classifiers except for 1NN-DTW-R which fell in the two intermediate groups.

We want to point out that performance difference of some pairs of classifiers which were found to be statistically significant using the Wilcoxon signed-rank test (Tables 2 and 5) were not found to be statistically significant by the Nemenyi test. For example, this is true even for the pairs 1NN-ED and 1NN-DTW as well as for the pairs 1NN-DTW and 1NN-DTW-R. However, if we compare fewer than thirteen classifiers using the Nemenyi test then some of these pairs of classifiers show significant differences. We also point out that while Wilcoxon signed rank test takes into account the difference in performances between the classifiers over each of the datasets, the Nemenyi test only

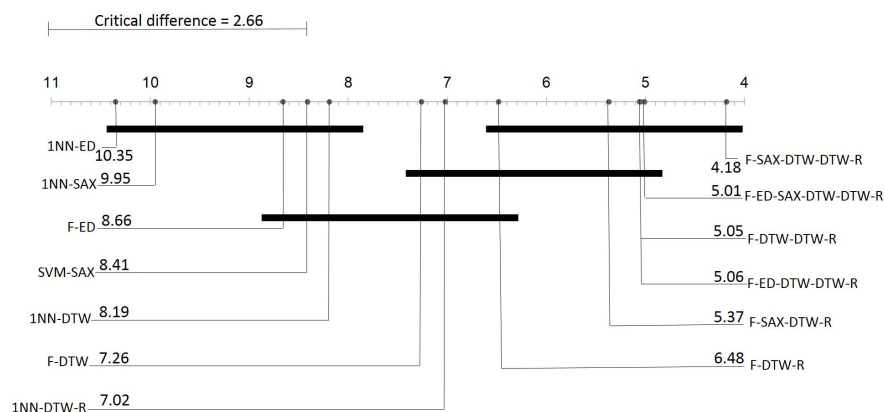


Fig. 6 Comparing thirteen classifiers using their average ranks on the 47 UCR time series classification datasets. The feature-based classifiers are represented with prefix “F”. Groups of classifiers that are not significantly different from each other at $p < 0.05$ using the Nemenyi test (corresponding to the critical difference of 2.66) are shown connected.

considers their ranks and not the differences in performances. Hence while Nemenyi test can give a bigger picture comparing all the classifiers, to compare two individual classifiers it is better to rely on the results of the Wilcoxon signed-rank test.

5.4 Experiments with Ensembles

Ensemble-based classifiers combine multiple classifiers to do classification and typically outperform the component classifiers (Dietterich 2000). Lines and Bagnall (2014) created ensembles of 1NN time series classifiers that use different elastic distance measures, including DTW-R, and show that they significantly outperform 1NN-DTW-R. We decided to create ensembles of the classifiers presented in this paper to see if they would further improve the results. We used the *Proportional* weighting scheme to create ensembles which was shown to work best for time series classification (Lines and Bagnall 2014). In this scheme, for every dataset, each component classifier is given a normalized weight proportional to its cross-validation accuracy on the training set (we used ten-fold cross-validation). Then for a test example, an output label is assigned the same weight as that of the classifier that outputs it, and in case multiple classifiers output a label then that label is assigned the weight equal to the sum of those classifiers’ weights. The label with the highest weight is considered as the ensemble’s output label. Thus in this scheme, if all component classifiers disagree on a test example’s label then the output of the ensemble will be same as the output of the most accurate component classifier. But in other cases, the decision of the most accurate classifier will be over-riden if other classifiers agree on a different label and their combined weight exceeds the weight of the most accurate classifier.

Method	1NN-DTW-R	Feature-SAX-DTW-DTW-R	Ensemble-baselines	Ensemble-features	Ensemble-baselines-features
Ensemble-baselines	38/4/5	17/27/3	-		
Ensemble-features	33/10/4	26/17/4	28/15/4	-	
Ensemble-baselines-features	42/3/2	27/16/4	35/5/7	26/16/5	-
Ensemble-final	42/3/2	31/11/5	40/5/2	30/8/9	27/13/7

Table 7 Comparisons of various ensembles among themselves and with 1NN-DTW-R and Feature-SAX-DTW-DTW-R classifiers. Ensemble-baselines is an ensemble of 1NN-DTW, 1NN-DTW-R and SVM-SAX; Ensemble-features is an ensemble of Feature-DTW, Feature-DTW-R and SVM-SAX; Ensemble-baselines-features is an ensemble of the foregoing five component classifiers; and Ensemble-final is an ensemble of the same five component classifiers and Feature-SAX-DTW-DTW-R. All comparisons were done using the 47 UCR time series classification datasets. Each cell in the table shows on how many datasets did the classifier in its row win/lose/tie over the classifier in its column. The results shown in bold were found to be statistically significant at $p < 0.05$ using the two-tailed Wilcoxon signed-rank test.

Table 7 shows the results of comparing various ensembles among themselves and with 1NN-DTW-R (strongest baseline) and Feature-SAX-DTW-DTW-R (our best non-ensemble classifier) on the 47 UCR time series classification datasets. We created the first ensemble using three baseline classifiers: 1NN-DTW, 1NN-DTW-R and SVM-SAX, and call it Ensemble-baselines. It can be seen from the first row that it outperforms 1NN-DTW-R with win/loss/tie of 38/4/5. This is not surprising because an ensemble typically outperforms its component classifiers. But from the first row it can also be seen that this ensemble does worse than Feature-SAX-DTW-DTW-R (win/loss/tie: 17/27/3 with $p = 0.058$). This shows that a classifier that merely combines the outputs of the baseline classifiers does not perform as well as a classifier that combines their distance features through SVM.

Next, we created an ensemble that combines classifiers Feature-DTW, Feature-DTW-R and SVM-SAX, and call it Ensemble-features. Note that we included SVM-SAX again in this ensemble because it is a feature based classifier besides being a baseline. As it can be seen from the second row of the table, it outperforms the Ensemble-baselines which is not surprising as it combines more accurate component classifiers. However, it is interesting to note that it also outperforms Feature-SAX-DTW-DTW-R (win/loss/tie: 26/17/4 with $p = 0.079$).

We created the next ensemble that combines classifiers 1NN-DTW, 1NN-DTW-R, Feature-DTW, Feature-DTW-R and SVM-SAX, and call it Ensemble-baselines-features. It outperforms the previous two ensembles (third row) which is not surprising given that it includes their component classifiers. Finally, we added classifier Feature-SAX-DTW-DTW-R to the foregoing five component classifiers to create an ensemble we call Ensemble-final. It outperforms all other ensembles, including Ensemble-baselines-features. This shows that Feature-SAX-DTW-DTW-R is an independent accurate classifier and adds value to the ensemble of the other component classifiers. This ensemble out-

performs 1NN-DTW-R with win/loss/tie of 42/3/2. It may also be noted from Table 7 that all our ensembles outperform 1NN-DTW-R, this is again not surprising given that they include 1NN-DTW-R as a component classifier (except for Ensemble-features which includes Feature-DTW-R). We have included detailed results of Ensemble-final on all the 47 UCR datasets in the last column of Table 4. That table also shows the error rate of the best classifier for each dataset in bold. Its last row shows the number of wins (including ties) over all the datasets. Ensemble-final obtains most wins (27). The detailed results of other ensembles are available through our web-site (Kate 2014). It should be noted that for the ensembles we have presented, all the DTW distances can be computed once which then could be used by all their component classifiers. Given that the component classifiers are very fast with pre-computed DTW distances, hence these ensembles do not have time complexity far from that of say Feature-SAX-DTW-DTW-R classifier.

We found that our Ensemble-final performs better than the best ensemble (PROP) of Lines and Bagnall (2014) (win/loss/tie: 27/16/3 with $p < 0.05$). In future, one may combine our component classifiers with theirs to form bigger ensembles. One may also create feature-based classifiers using all the distance measures they used and then use them to create ensembles.

6 Future Work

The contribution of this paper is not just a method that works very well on time series classification, but a method that can also be easily combined with other machine learning based methods to further improve the classification performance. While we demonstrated this with SAX, our method can be easily combined with methods that use any type of statistical or symbolical (including shapelets (Ye and Keogh 2009)) features by simply adding them as additional features. Many applications have particular patterns of time series from which application-specific features could be extracted that could help in classification. For example, Electrocardiogram (ECG) time series have known up and down patterns of heartbeats from which important features could be extracted that could help in classifying healthy heartbeats from unhealthy ones (Nugent et al 2002). Such features could be easily added to the DTW based features in our method and this may further improve the classification performance. This is not easy to do with the 1NN-DTW approach. Also, any kernel based time series classification method could also be combined with our approach by simply defining a new kernel which is a function (Hausler 1999) of the existing kernel and our DTW feature based kernel. We also note that our method could take advantage of any existing or future improvement in efficiency or accuracy on either DTW computation or on the underlying machine learning algorithm.

One may also explore if other distance measures or other variations of DTW and their combinations can further improve the performance when used as features. We used SVM as our machine learning method, one may explore if

other machine learning methods would do better on certain types of datasets. Using distance based features for time series clustering or for searching time series data is another possibility of future work. Given that methods for time series regression (either to predict a future value or a numerical score) often use statistical features, one may also consider adding DTW-based features to them. Finally, getting more insights into the conditions under which this method outperforms 1NN-DTW-R and the conditions under which it lags behind will also be a fruitful future work.

7 Conclusions

We presented a simple approach that uses DTW as features and thus combines the strengths of DTW and advanced machine learning methods for time series classification. Experimental results showed that it convincingly outperforms the DTW distance based one-nearest neighbor method that is known to work exceptionally well on this task. The approach can also be easily combined with many existing time series classification methods thus combining the strengths of DTW and those methods. We showed that when combined with SAX method, it significantly improved its performance. We also showed that ensembles of these classifiers further improve results on time series classification.

8 Acknowledgments

We thank editor Eamonn Keogh and the anonymous reviewers for their feedback to improve this paper.

References

- Batista G, Wang X, Keogh EJ (2011) A complexity-invariant distance measure for time series. In: *SDM, SIAM*, vol 11, pp 699–710
- Baydogan M, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Recognition and Machine Intelligence* 35(11):2796–2802
- Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: *KDD workshop, Seattle, WA*, vol 10, pp 359–370
- Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27
- Chen L, Ng R (2004) On the marriage of lp-norms and edit distance. In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment*, pp 792–803
- Chen Y, Hu B, Keogh EJ, Batista G (2013) DTW-D: Time series semi-supervised learning from a single example. *Proceedings of the Nineteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2013)* pp 383–391
- Cristianini N, Shawe-Taylor J (2000) *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7:1–30

- Dietterich TG (2000) Ensemble methods in machine learning. In: Multiple classifier systems, Springer, pp 1–15
- Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1(2):1542–1552
- Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases, In: *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pp 419–429
- Fulcher BD, Jones NS (2014) Highly comparative feature-based time-series classification. *IEEE Transactions in Knowledge and Data Engineering*
- Geurts P (2001) Pattern extraction for time series classification. In: *Principles of Data Mining and Knowledge Discovery*, Springer, pp 115–127
- Gudmundsson S, Runarsson TP, Sigurdsson S (2008) Support vector machines and dynamic time warping for time series. In: *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)*. *IEEE International Joint Conference on*, IEEE, pp 2772–2776
- Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. *ACM SIGKDD explorations newsletter* 11(1):10–18
- Harvey D, Todd MD (2014) Automated feature design for numeric sequence classification by genetic programming. *IEEE Transactions on Evolutionary Computation*. doi:10.1109/TEVC.2014.2341451
- Hausser D (1999) Convolution kernels on discrete structures. Tech. rep., Technical report, UC Santa Cruz
- Hayashi A, Mizuhara Y, Suematsu N (2005) Embedding time series data for classification. In: *Machine Learning and Data Mining in Pattern Recognition*, Springer, pp 356–365
- Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2013) Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery* pp 1–31
- Itakura F (1975) Minimum prediction residual principle applied to speech recognition. *Acoustics, Speech and Signal Processing*, *IEEE Transactions on* 23(1):67–72
- Japkowicz N, Shah M (2011) *Evaluating learning algorithms: a classification perspective*. Cambridge University Press
- Kate RJ (2014) UWM time series classification webpage. <http://www.uwm.edu/~katerj/timeseries>
- Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery* 7(4):349–371
- Keogh E, Lin J, Fu A (2005) HOT SAX: Efficiently finding the most unusual time series subsequence. In: *Fifth IEEE international conference on Data Mining (ICDM)*, IEEE, pp 226–233
- Keogh E, Zhu Q, Hu B, Hao Y, Xi X, Wei L, Ratanamahatana CA (2011) The UCR time series classification/clustering homepage. http://www.cs.ucr.edu/~eamonn/time_series_data
- Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15(2):107–144
- Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems* 39(2):287–315
- Lines J, Bagnall A (2014) Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery* pp 1–28
- Manning CD, Raghavan P, Schütze H (2008) *Introduction to information retrieval*, vol 1. Cambridge university press Cambridge
- Mizuhara Y, Hayashi A, Suematsu N (2006) Embedding of time series data by using dynamic time warping distances. *Systems and Computers in Japan* 37(3):1–9
- Nanopoulos A, Alcock R, Manolopoulos Y (2001) Feature-based classification of time-series data. *International Journal of Computer Research* 10(3)
- Nugent C, Lopez J, Black N, Webb J (2002) The application of neural networks in the classification of the electrocardiogram. In: *Computational Intelligence Processing in Medical Diagnosis*, Springer, pp 229–260
- Ordóñez P, Armstrong T, Oates T, Fackler J (2011) Classification of patients using novel multivariate time series representations of physiological data. In: *Machine Learning and*

- Applications and Workshops (ICMLA), 2011 10th International Conference on, IEEE, vol 2, pp 172–179
- Ratanamahatana CA, Keogh E (2004a) Everything you know about dynamic time warping is wrong. In: Third Workshop on Mining Temporal and Sequential Data, pp 22–25
- Ratanamahatana CA, Keogh E (2004b) Making time-series classification more accurate using learned constraints. In: Proceedings of SIAM International Conference on Data Mining (SDM '04), pp 11–22
- Rodríguez JJ, Alonso CJ (2004) Interval and dynamic time warping-based decision trees. In: Proceedings of the 2004 ACM symposium on Applied computing, ACM, pp 548–552
- Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on* 26(1):43–49
- Senin P, Malinchik S (2013) SAX-VSM: Interpretable time series classification using sax and vector space model. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, pp 1175–1180
- Shieh J, Keogh E (2008) i SAX: Indexing and mining terabyte sized time series. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 623–631
- Vickers A (2010) What is a P-value anyway?: 34 stories to help you actually understand statistics. Addison-Wesley
- Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Data Engineering, 2002. Proceedings. 18th International Conference on, IEEE, pp 673–684
- Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery* 26(2):275–309
- Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on Machine learning, ACM, pp 1033–1040
- Xing Z, Pei J, Keogh E (2010) A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter* 12(1):40–48
- Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 947–956