Noname manuscript No.
(will be inserted by the editor)

Query-sensitive Distance Measure Selection for Time Series Nearest Neighbor Classification

Alexios Kotsifakos · Vassilis Athitsos · Panagiotis Papapetrou

Received: date / Accepted: date

Abstract Many distance or similarity measures have been proposed for time series similarity search. However, none of these measures is guaranteed to be optimal when used for 1-Nearest Neighbor (NN) classification. In this paper we study the problem of selecting the most appropriate distance measure, given a pool of time series distance measures and a query, so as to perform NN classification of the query. We propose a framework for solving this problem, by identifying, given the query, the distance measure most likely to produce the correct classification result for that query. From this proposed framework, we derive three specific methods, that differ from each other in the way they estimate the probability that a distance measure correctly classifies a query object. In our experiments, our pool of measures consists of Dynamic Time Warping (DTW),

Alexios Kotsifakos
Department of Computer Science and Engineering
University of Texas at Arlington
E-mail: alexios.kotsifakos@mavs.uta.edu

Vassilis Athitsos
Department of Computer Science and Engineering
University of Texas at Arlington
E-mail: athitsos@uta.edu

Panagiotis Papapetrou
Department of Computer and Systems Sciences
Stockholm University
E-mail: panagiotis@dsv.su.se

Move-Split-Merge (MSM), and Edit distance with Real Penalty (ERP). Based on experimental evaluation with 45 datasets, the best-performing of the three proposed methods provides the best results in terms of classification error rate, compared to the competitors, which include using the Cross Validation method for selecting the distance measure in each dataset, as well as using a single specific distance measure (DTW, MSM, or ERP) across all datasets.

Keywords Time series · Classification · Distance Measures

1 Introduction

Collections of time series data are produced in vast quantities and in a wide range of application domains. Such time series data contain valuable information, which can be extracted using efficient knowledge discovery methods. A fundamental task in knowledge discovery is *1-Nearest Neighbor (NN) classification*. According to this task, given a dataset of time series belonging to certain categories/classes (also known as *training* time series) and an unclassified query time series, we identify its class by looking at the class of its nearest neighbor. The nearest neighbor is found by computing the distance between the query and all time series in the dataset via a distance measure, and then selecting the time series with the smallest distance. In such a setting, the key challenge is to define or choose an appropriate distance measure that is expected to retrieve the correct class label for a given query.

Several distance and similarity measures have been developed for performing *whole sequence matching* between two time series [27], such as Dynamic Time Warping (DTW) [17], Edit Distance on Real sequence (EDR) [8], Edit distance with Real Penalty (ERP) [7], Time Warp Edit Distance (TWED) [22], and Move-Split-Merge (MSM) [25]. Moreover, alternative methods have been developed for the same problem, that focus on global or local structural sim-

ilarity, such as SpaDe [9], Shapelets [29], DFT [1], Bag-Of-Patterns [20], and SAX [19]. Nonetheless, none of these methods is guaranteed to be optimal for the task of NN classification. In other words, some measure or method may fail to correctly classify some queries, while some other measure or method may be successful with these queries, and vice versa.

We will now illustrate the previous observation with a motivating example. The UCR time series repository [15] provides a collection of 45 time series datasets from a variety of application domains. More information about these datasets can be found in Section 5.1. Suppose that we have a *pool* of distance measures to choose from, consisting for example of DTW and MSM. Our main question is the following: given a query time series, that we want to classify, which of these two measures is more likely to classify the query correctly under nearest neighbor classification? Unfortunately the answer to this question is not straightforward.

In Figure 1 we present, for each of the 45 datasets in the UCR repository, the number of time series where the class of the NN for DTW is different than that of MSM (red diamond). We also show, for these datasets, how many time series are classified correctly by either DTW or MSM (blue cross and green circle, respectively). Finally, we show the number of time series misclassified by both DTW and MSM (black asterisk). We note that, in each dataset, there are several objects where DTW and MSM produce different class labels. In some of those cases DTW gives the right answer, in some of those cases MSM gives the right answer. Neither of the two distances has a clear advantage.

The more distance measures we have available to choose from, the higher the chance that at least one of those distance measures correctly classifies the given query. Figure 2 illustrates results of our study when more distance measures are added to our pool. Specifically, we show for each dataset the number of misclassified time series when the pool consists of only DTW (blue

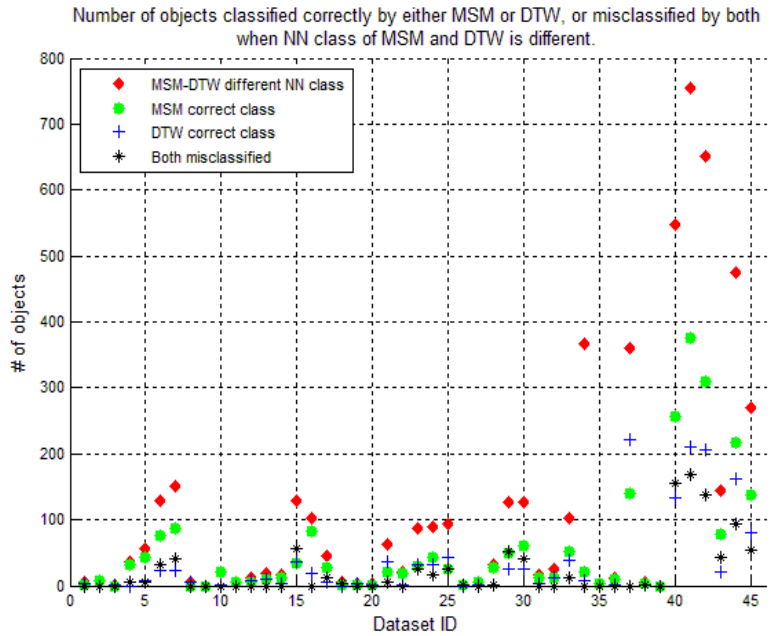

Fig. 1 An example that illustrates the challenging nature of our problem. We used 45 datasets from the UCR time series repository [15] (x axis). On the y axis we show the number of time series for each dataset that are classified correctly by either DTW or MSM, or misclassified by both, when the class of the Nearest Neighbor object is different between DTW and MSM. It can be seen that the number of time series correctly classified by DTW is comparable to that of MSM.

asterisk), or MSM and DTW (black cross), or ERP, MSM, and DTW (red circle). For example, for datasets with IDs 41 and 44, there are respectively 788 and 396 time series misclassified when our pool consists of only DTW. When MSM is also added to the pool, the numbers of time series misclassified by both MSM and DTW drop to 413 and 179, respectively. Finally, when ERP is also included in the pool, the numbers of time series that are misclassified by all three measures drop further to 342 and 145, respectively, for the same datasets.

The typical way of performing nearest neighbor classification is to choose a specific distance measure, and use that distance measure to classify every single test object. However, by looking at Figures 1 and 2, we note that different

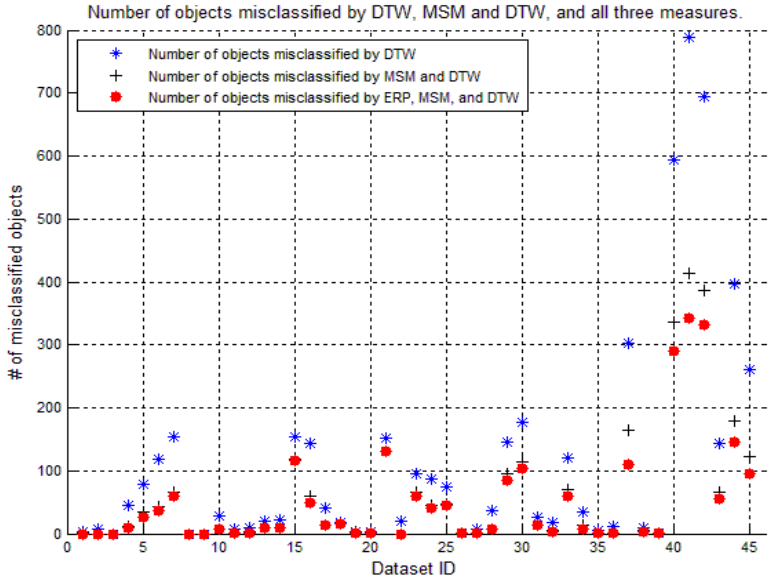

Fig. 2 An example showing that the number of available distance measures may affect the classification accuracy. We used 45 datasets from the UCR time series repository (x axis). On the y axis we show the number of time series for each dataset that are misclassified by all measures in the pool. It can be seen that as the number of measures in the pool increases (from 1 to 3), the number of time series that are still incorrectly classified by all measures in the pool decreases for most of the datasets, or remains the same.

distance measures oftentimes disagree with each other. In such cases, a natural question that arises is whether we can obtain a method that can automatically choose, given a query object, the distance measure that produces the correct answer for that query object. If we had such a method, and it worked perfectly, then we would only misclassify objects for which all distance measures produce a wrong answer. For example, with such a perfect method, the error rates attained for the 45 datasets of Figure 2 would correspond to the red circles, as opposed to the higher rates attained when simply using a single distance measure in all cases.

This is exactly the challenging problem that we study in this paper: given a pool of distance measures and a query time series, how to identify the most appropriate distance measure for NN classification for that query. Note that

the focus of this paper is not to find the best distance measure or "golden" standard method for time series NN classification, but the most suitable one for a given query in a predefined pool of measures.

More precisely, the novelty of our paper can be summarized by the following four key **contributions**:

- To the best of our knowledge, there exists no relevant work towards selecting the best measure, given a query and a pool of measures, for time series NN classification.
- We propose a framework for solving the aforementioned problem, where the selection of the appropriate distance measure is performed by measuring, for each distance measure, the likelihood that the distance measure will produce the correct result for a query. The framework is also query-sensitive, i.e., the selected distance measure can be different for each query.
- Within this framework we propose three methods, that employ different schemes for measuring the likelihood that a distance measure will correctly classify a specific query object. All three methods measure this likelihood based on the concept of the T-*neighborhood* of a query: this T-neighborhood is a set of training objects that are similar to the query in some specific property. The three methods differ from each other in that each method uses a different property to identify training objects similar to the query.
- We provide an extensive experimental evaluation of the proposed methods on a large collection of 45 time series datasets from the UCR repository [15]. In this evaluation, out of the three proposed methods, the method that produces the best performance in terms of classification results is called `homogeneity-based`. This method also outperforms all competitors in a statistically significant manner.

2 Related Work

Several methods and time series representations have been proposed for time series similarity search [27]. The most common measure for computing the distance between time series is DTW [17], which can be used to compare sequences that may vary in time or speed. Several variants of DTW have also been proposed, such as constrained DTW (cDTW) [24], EDR [8], and ERP [7]. The most attractive property of these algorithms is that they are robust to misalignments along the temporal axis, i.e., to differences in the speed in which observations evolve across time. The first variant, cDTW, avoids the matching of elements that are temporally far away from each other, by bounding the difference of the sequences' indices. ERP is more robust to noisy data compared to EDR, while both can be used for pruning purposes since they satisfy the triangle inequality. Some of these methods, such as DTW and cDTW, have been shown to achieve high accuracy in applications such as time series mining and classification [8, 14].

A novel measure for time series, called MSM, was proposed recently [25]. MSM is metric and uses three fundamental operations, Move, Split, and Merge, which can be applied in some sequence to transform any time series into any other time series. In addition, it is invariant to the choice of origin, as opposed to ERP. Another widely used measure for finding the similarity between two sequences is the Longest Common SubSequence (LCSS) [5, 6]. As its name implies, LCSS finds the largest number of elements that are common in both sequences, and it allows for gaps on both sequences during their alignment. The Edit distance [18], which is a metric measure, can be used to find the distance between two strings. It is defined as the minimum number of edit operations needed to transform one string into the other, with the allowable operations being insertion, deletion, and substitution of a single character.

The Edit distance with several variations has also been used for time series matching, for example in music retrieval [16,26]. Finally, TWED [22] is a metric distance measure whose goal is to find a sequence of edit operations (deleting an element from any time series and matching two elements) allowing for the simultaneous transformation of two time series so as to superimpose them with minimal cost. Additionally, TWED is an elastic measure supporting local time shifting using timestamp differences between compared points. All the above measures could be used in our pool of measures for NN classification.

Another recent line of research has been focusing on identifying discriminant time series subsequence patterns, known as *Shapelets* [29] and variants [13,21,23,28]. Shapelets are mainly used for time series classification, since due to their construction they are expected to be more informative and representative of some class. Other time series representations that capture global or local structural characteristics include SpaDe [9], DFT [1], and SAX [19]. The proposed method is orthogonal to these classification methods; these methods could be employed for NN classification in time series databases, and hence can be added in the pool of measures from which the proposed framework selects the best one for each query.

In order to deal with the curse of dimensionality when performing NN classification, a linear discriminant analysis has been proposed [12] to estimate an effective metric for computing neighborhoods. Based on centroid information, local decision boundaries are determined, the neighborhoods are shrunk in directions orthogonal to these boundaries, and any NN classifier can be performed on the modified neighborhoods. Similarly to this approach, Domeniconi et al. [11] propose an adaptive NN classification method to minimize estimation bias in high dimensions. Based on Chi-squared distance analysis, a flexible metric, which depends on query locations in the feature space, is estimated for computing neighborhoods that are constricted along the most influential

feature dimensions. Both of these methods are designed for patterns that are represented as vectors in Euclidean space, and thus they are not applicable to time series, which is the focus of this work. Furthermore, they do not select the most appropriate query-based measure from a pool of measures, rather they try to find the most influential dimensions. Finally, Athitsos et al. [2] proposed a method for approximate nearest neighbor retrieval by mapping objects from the original space to a real vector space using a set of reference objects. The query-sensitive nature of that work is that the distance measure used in the vector space is a weighted Lp-norm, where the weights are learned via boosting during an expensive pre-processing step. In our work, we tackle a much different problem, that of classifying a time series by selecting the most appropriate distance measure for the given query out of a pool of measures.

3 Background

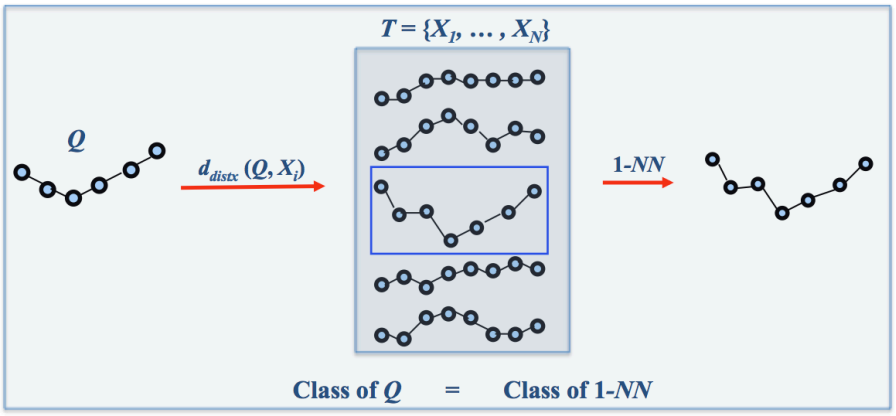

Fig. 3 Illustration of the 1-NN classification task. Given a query Q and a collection of time series $\mathcal{T}$, the class of the retrieved 1-NN (closest) time series will be the class of Q.

Table 1 Notation Table.

Notation	Explanation
$X = (x_1, \ldots, x_{\|X\|})$	A 1-dimensional time series of length $\|X\|$.
$dist$	A distance measure.
$d_{\text{dist}}(X, Y)$	A distance function computing $dist$ between X and Y.
$\mathcal{L} = \{dist_1, \ldots, dist_n\}$	A pool of distance measures.
$\mathcal{T} = \{X_1, \ldots, X_N\}$	A collection of N training time series.
$\mathcal{D}_{X_i,\mathcal{T}}^{dist_x}$	Set of distances between X_i and all time series in $\mathcal{T}$ using $dist_x$.
f^{scheme}	A scheme function.
s_i	The score given by a scheme function f^{scheme} for X_i.
o_i	The closest to X_i training time series for some $dist_x$.
cr_i	The 1-NN classification result for X_i using $dist_x$.
s	An array of all s_i values for some $dist_x$.
o	An array of all o_i values for some $dist_x$.
cr	An array of all cr_i values for some $dist_x$.
s'^{dist_x}	The sorted values of s for some $dist_x$.
o'^{dist_x}	The sorted indices of o based on the sorting of s'^{dist_x}.
cr'^{dist_x}	The sorted indices of cr based on the sorting of s'^{dist_x}.
Q	A query time series.
Q_o	The closest to Q training time series.
Q_s	The score given by a scheme function f^{scheme} for Q.
$pos(Q_s, s'^{dist_x})$	The position of Q_s in s'^{dist_x}.
T	The T-neighborhood parameter.
v_{dist_x}	The T-neighborhood classification vector for some $dist_x$.
$P_E^{dist_x}$	The 1-NN classification error probability for some $dist_x$.
$dist_{min}$	The measure with the lowest classification error probability.
$\mathcal{V}^T$	The set of T-neighborhood classification vectors for all measures.
$pval(\mathcal{V}^T, \alpha)$	The p-value computed by ANOVA with significance threshold α.

In this section we introduce some background definitions and formulate the problem studied in this paper. For the convenience of the reader, in Table 1 we present the notation used throughout the paper.

Let $X = (x_1, \ldots, x_{|X|})$ and $Y = (y_1, \ldots, y_{|Y|})$ be two time series, where $x_i, y_j \in \mathbb{R}$, $\forall (i = 1, \ldots, |X|; j = 1, \ldots, |Y|)$. A time series can be seen as a variable evolving over time, sampled at regular time intervals. Examples of time series are shown in Figure 3. Given a distance measure $dist$, the dis-

tance between X and Y is defined as a function $d_{\text{dist}}(X, Y)$. Additionally, $\mathcal{L} = \{dist_1, \ldots, dist_n\}$ defines a *pool of distance measures*, where each $dist_x$, $x = 1, \ldots, n$, is a distance measure.

Problem Setting. Given a collection of time series $\mathcal{T} = \{X_1, \ldots, X_N\}$, a pool of distance measures $\mathcal{L}$, and a query time series Q, we want to identify the distance measure $dist_x \in \mathcal{L}$ that is most suitable to perform NN classification for Q.

An illustration of the 1-NN classification task is given in Figure 3. The class of the retrieved 1-NN will be the class assigned to Q. In this paper we explore three time series distance measures, which are used to construct our pool $\mathcal{L}$. These measures are: Dynamic Time Warping (DTW) [17], Edit distance with Real Penalty (ERP) [7], and Move-Split-Merge (MSM) [25]. The selection of these measures is based on the following three rationales: (1) DTW is extensively used for time series matching and has been shown to provide excellent classification accuracy results [27], (2) ERP is a variant of DTW and Edit Distance that fixes the non-metric property of DTW, and (3) MSM is a metric distance measure that has been proposed very recently and is shown to outperform ERP and DTW in terms of NN classification accuracy for several datasets. These three measures are described next in more detail.

It should be noted that we do not make any claim that these are the best measures among all existing ones. Our aim is to demonstrate that the proposed framework can achieve very competitive performance to existing measures in terms of NN classification accuracy, since instead of using only a single distance measure it exploits the strengths of each measure in the pool and identifies

the most appropriate one for a given query. Hence, other measures could be used alternatively without any change in the framework.

3.1 Dynamic Time Warping

DTW identifies an optimal alignment between two time series and computes the matching cost of that alignment in quadratic computational time. Each time series element is allowed to match with at least one element of the other time series, allowing for local stretching and shrinking along the time axis. Given two time series X and Y, their DTW distance $d_{\text{DTW}}(X, Y)$ is defined recursively using a dynamic programming matrix [4] $Cost$ of size $(|X|+1)\times(|Y|+1)$. A *null* element is added at the beginning of X and Y, and it matches the other null element with zero score and any other element with a score of ∞. Let $Cost_{i,j}$ denote the element at the i-th row and j-th column of $Cost$. Denoting with $L_p(x_i, y_j)$ the L_p norm based distance measure of x_i and y_j, we define $d_{\text{DTW}}(X, Y)$ as follows:

Initialization:

$$Cost_{0,0}(X,Y) = 0 \ , \ Cost_{0,j}(X,Y) = \infty \ , \ Cost_{i,0}(X,Y) = \infty \ .$$

Main Loop:

$$Cost_{i,j}(X,Y) = L_p(x_i, y_j) + \min \left\{ \begin{array}{l} Cost_{i,j-1}(X,Y), \\ Cost_{i-1,j}(X,Y), \\ Cost_{i-1,j-1}(X,Y) \end{array} \right\}$$

$$\forall (i = 1, \ldots, |X|; j = 1, \ldots, |Y|) \ .$$

Output:

$$d_{\text{DTW}}(X,Y) = Cost_{|X|,|Y|}(X,Y) \ .$$

3.2 Edit distance with Real Penalty

ERP [7] is a metric distance measure with quadratic time complexity that can be seen as a variant of the L_1-norm, EDR, and DTW. The main advantage over DTW is that it satisfies the triangle inequality. Specifically, while DTW replicates the value of the previous element when a gap is introduced in either time series, ERP applies a nonnegative constant penalty g for computing the distance between the gap and the corresponding element from the other time series. In the case of non-gap elements, the matching penalty is simply their L_1 norm. Given a nonnegative constant parameter g, function G is used in computing values for the $Cost$ array, where $G(x_i, y_j)$ is defined as follows:

$$G(x_i, y_j) = \begin{cases} |x_i - y_j|, \text{ if } x_i \text{ and } y_j \text{ are not gaps,} \\ |x_i - g|, \text{ if } y_j \text{ is a gap,} \\ |g - y_j|, \text{ if } x_i \text{ is a gap .} \end{cases}$$

The distance $d_{\text{ERP}}(X,Y)$ is now defined as follows:

Initialization:

$$Cost_{0,j}(X,Y) = \sum_{j=1}^{|Y|} |y_j - g|$$

$$Cost_{i,0}(X,Y) = \sum_{i=1}^{|X|} |x_i - g|$$

Main Loop:

$$Cost_{i,j}(X,Y) = \min \left\{ \begin{array}{l} Cost_{i-1,j-1}(X,Y) + G(x_i, y_j), \\ Cost_{i-1,j}(X,Y) + G(x_i, gap), \\ Cost_{i,j-1}(X,Y) + G(gap, y_j) \end{array} \right\}$$

$$\forall (i = 1, \ldots, |X|; j = 1, \ldots, |Y|) \ .$$

Output:

$$d_{\text{ERP}}(X,Y) = Cost_{|X|,|Y|}(X,Y) \ .$$

3.3 Move-Split-Merge

MSM [25] is a metric time series distance measure, robust to misalignments, translation invariant, and has again a quadratic computational time complexity. Given two times series X and Y, MSM transforms X to Y by employing three fundamental operations: Move, Split, and Merge. The Move operation changes the value of a single point of the time series, the Split operation splits a single point of the time series into two consecutive points that have the same value as the original point, while the Merge operation merges two successive equal values into one. Thus, the MSM distance between X and Y, $d_{\text{MSM}}(X,Y)$, is defined as the cost of the lowest-cost transformation of X to Y. Similarly to DTW and ERP, given two time series X and Y, their MSM distance can be computed using dynamic programming. For each (i,j) such that $1 \leq i \leq |X|$ and $1 \leq j \leq |Y|$, $Cost_{i,j}$ is defined to be the MSM distance between the first i elements of X and the first j elements of Y. Given a nonnegative constant parameter c, function C is used in computing values for the $Cost$ array, where

$C(x_i, x_{i-1}, y_j)$ is:

$$C(x_i, x_{i-1}, y_j) = \begin{cases} c, \text{ if } x_{i-1} \leq x_i \leq y_j \text{ or } x_{i-1} \geq x_i \geq y_j \\ c + \min(|x_i - x_{i-1}|, |x_i - y_j|), \text{ otherwise} \end{cases}$$

The distance $d_{\text{MSM}}(X, Y)$ is now defined as follows:

Initialization:

$$Cost_{1,1}(X, Y) = |x_1 - y_1| \ .$$

$$Cost_{i,1}(X, Y) = Cost_{i-1,1}(X, Y) + C(x_i, x_{i-1}, y_1) \ .$$

$$Cost_{1,j}(X, Y) = Cost_{1,j-1}(X, Y) + C(y_j, x_1, y_{j-1}) \ .$$

$$\forall (i = 2, \ldots, |X|; j = 2, \ldots, |Y|) \ .$$

Main Loop:

$$Cost_{i,j}(X, Y) = \min \left\{ \begin{array}{l} Cost_{i-1,j-1}(X, Y) + |x_i - y_j|, \\ Cost_{i-1,j}(X, Y) + C(x_i, x_{i-1}, y_j), \\ Cost_{i,j-1}(X, Y) + C(y_j, x_i, y_{j-1}) \end{array} \right\}$$

$$\forall (i = 2, \ldots, |X|; j = 2, \ldots, |Y|) \ .$$

Output:

$$d_{\text{MSM}}(X, Y) = Cost_{|X|,|Y|}(X, Y) \ .$$

4 Query-sensitive Measure Selection

In this section we propose a framework for solving the problem studied in this paper, as defined in Section 3, and also three methods that are based on this framework.

4.1 Measure-selection Framework

The proposed framework consists of two steps: the *offline* and the *online* step.

4.1.1 Offline step

This is a preprocessing step and is performed only once per dataset and per distance measure. Given a set of N training time series $\mathcal{T} = \{X_1, \ldots, X_N\}$ (also referred to as training objects) and a distance measure $dist_x$, we first compute the distance of each time series $X_i \in \mathcal{T}$ to all other time series $X_j \in \mathcal{T}$, resulting in the following set of distances:

$$\mathcal{D}^{dist_x}_{X_i,\mathcal{T}} = \{d_{dist_x}(X_i, X_j) | \forall X_j \in \mathcal{T}, i \neq j\} \ . \tag{1}$$

Next, for each $X_i \in \mathcal{T}$ we determine its closest time series o_i based on $dist_x$, i.e.,

$$o_i = \text{argmin}\left\{\mathcal{D}^{dist_x}_{X_i,\mathcal{T}}\right\}. \tag{2}$$

Then, we treat each X_i as a query and determine whether it was correctly classified by $dist_x$ using the 1-NN classifier, i.e., by comparing the class label of training object o_i with the actual class label of X_i. The result of this comparison is stored in cr_i:

$$cr_i = \begin{cases} 0, \text{ if } X_i \text{ is correctly classified,} \\ 1, \text{ otherwise.} \end{cases} \tag{3}$$

The last step of off-line preprocessing has to do with defining a property of each training object, that can be used at the online stage to identify, given a query object, the "most similar" training objects. This way, we can use for the query object the distance measure that gives the best classification accuracy in those "most similar" training objects. As explained in Section 4.2, we have used three different schemes for defining such a property. For the purposes of the description of the off-line preprocessing step, in this subsection, we will consider the scheme for defining such a property as a black box.

In particular, we use the term "scheme function", and notation f^{scheme}, as a black box for a function that, given a time series X and the pairwise distances of X to all objects in the training set $\mathcal{T}$ using $dist_x$, produces a score s based on these distances. Note that, if X_i is already part of the training set, we only consider the remaining $N-1$ objects in $\mathcal{T}$. The score s_i of training object X_i is computed as follows:

$$s_i = f^{scheme}(X_i, \mathcal{D}^{dist_x}_{X_i,\mathcal{T}}) \ . \tag{4}$$

Further details about schemes and their functions are discussed in Section 4.2.

As a result, given $dist_x$, for each $X_i \in \mathcal{T}$ we store three values: o_i, s_i, and cr_i. Effectively, this produces three arrays of size N:

- $o = [o_1 \dots o_N]$: for each X_i the index of its closest training time series o_i using $dist_x$,
- $cr = [cr_1 \dots cr_N]$: for each X_i the classification result cr_i determined by the 1-NN classifier using $dist_x$, and
- $s = [s_1 \dots s_N]$: for each X_i the score s_i given by the scheme function f^{scheme} using $dist_x$.

In addition, for ease of the online step, array s is sorted in ascending order, while the indices of arrays o and cr are rearranged accordingly. This sorting procedure results in the rearranged arrays $o' = [o'_1 \dots o'_N]$, $s' = [s'_1 \dots s'_N]$, and $cr' = [cr'_1 \dots cr'_N]$ for a given $dist_x$.

Since the offline step is performed for *each* distance measure $dist_x \in \mathcal{L}$, the corresponding rearranged arrays are denoted as o'^{dist_x}, s'^{dist_x}, and cr'^{dist_x}, respectively.

4.1.2 Online step

Given a query object Q, the online step consists of measuring, for each distance measure $dist_x$, the likelihood that $dist_x$ will correctly classify Q, based on how accurately $dist_x$ classifies training objects similar to Q. Similarity between the query and training objects is measured based on scheme function f^{scheme}, which was used in the off-line preprocessing step to compute a score s_i for each training object X_i.

Distance computation. First, for each distance measure $dist_x \in \mathcal{L}$, the distances of Q to all training objects $X_j \in \mathcal{T}$ are computed, resulting in the following set:

$$\mathcal{D}_{Q,\mathcal{T}}^{dist_x} = \{d_{dist_x}(Q, X_j) | \forall X_j \in \mathcal{T}\} \ . \tag{5}$$

Similarly to the offline step, the closest training time series is identified, i.e.,

$$Q_o = \text{argmin}\left\{\mathcal{D}_{Q,\mathcal{T}}^{dist_x}\right\}, \tag{6}$$

and the score of the scheme function is recorded, i.e.,

$$Q_s = f^{scheme}(Q, \mathcal{D}_{Q,\mathcal{T}}^{dist_x}) \ . \tag{7}$$

The challenge now is to determine the class of Q by deciding which distance measure to "trust" for our NN classifier.

Query classification. To identify the training objects most similar to the query Q, we identify the position of Q_s in each s'^{dist_x} (computed in the offline step) as follows:

$$pos(Q_s, s'^{dist_x}) = \begin{cases} p, & \text{if } s'^{dist_x}_p = Q_s \text{ and } 1 \leq p \leq N, \\ p, & \text{if } s'^{dist_x}_p < Q_s < s'^{dist_x}_{p+1} \text{ and } 1 \leq p < N, \\ N, & \text{if } s'^{dist_x}_N < Q_s \ , \\ 1, & \text{if } s'^{dist_x}_1 > Q_s \ . \end{cases} \quad (8)$$

In other words, we identify the position p in s'^{dist_x} with value equal to Q_s. If Q_s does not appear exactly in s'^{dist_x}, then p corresponds to the position of the last value that is smaller than Q_s (except for the last case).

The proposed framework is based on the premise that objects given similar scores by the scheme function should have similar classification results for a given distance measure. Given a distance measure $dist_x$, the position of Q_s in array s'^{dist_x} is used to identify the most similar training objects, which constitute the neighborhood of Q according to the scheme that is being used. We note that, since scores s_i and Q_s depend on the distance measure $dist_x$, the neighborhood of Q is different for each distance measure.

In order to properly define this neighborhood, we must decide how large the neighborhood needs to be. To decide the size of the neighborhood, we define a parameter T. This parameter T depends on the query, and thus can be different for different queries. However, given a query Q, T is the same for all distance measures. This way, the neighborhoods of Q defined for all distance measures have the same size. From now on we will use the term "T-

neighborhood" to denote this neighborhood of Q, that is defined based on the scheme function and the choice of parameter T.

We will postpone the discussion of how to choose T for a few paragraphs. Once T has been defined, we select $T-1$ training time series from the left and T from the right side of $pos(Q_s, s'^{dist_x})$ in s'^{dist_x}, including $pos(Q_s, s'^{dist_x})$ itself. This results in $2*T$ time series in total (for each $dist_x \in \mathcal{L}$). In case that there are less than $T-1$ or T training time series on the left or right side of $pos(Q_s, s'^{dist_x})$, respectively, we fill out the remaining time series from its other side, so that there are always $2*T$ time series. The classification result for each of the $2*T$ time series can be directly retrieved from cr'^{dist_x}.

At this point, for each distance measure $dist_x \in \mathcal{L}$ we can extract the following vector of classification result values:

$$v_{dist_x} = (cr'^{dist_x}_{pos(Q_s, s'^{dist_x})-T+1}, \ldots, cr'^{dist_x}_{pos(Q_s, s'^{dist_x})+T}) \tag{9}$$

Using vectors v_{dist_x} for all $dist_x \in \mathcal{L}$, we compute the classification error probability for each distance measure as follows:

$$P_E^{dist_x} = \frac{\|v_{dist_x}\|_1}{2*T} \tag{10}$$

Next, we select the distance measure $dist_{min}$ with the lowest such probability and use that measure for the NN classification task for the given query:

$$dist_{min} = \underset{dist_x \in \mathcal{L}}{\operatorname{argmin}} \{P_E^{dist_x}\}. \tag{11}$$

Finally, Q is assigned the class of the closest training time series, as defined by $dist_{min}$.

The intuition for creating vectors v_{dist_x} and computing the error probabilities $P_E^{dist_x}$ for each measure $dist_x \in \mathcal{L}$ for a given T is the following: given

two measures $dist_i$ and $dist_j$, if there are more misclassified training time series in the T-neighborhood of Q using $dist_i$ than in the T-neighborhood of Q using $dist_j$, then this leads in a higher value of $P_E^{dist_i}$. More precisely, the T-neighborhood for $dist_i$ does not provide a good guarantee for correct classification of Q, in contrast to $dist_j$ that classifies more training objects around Q correctly according to the selected scheme. Consequently, $dist_j$ is more suitable for Q, since it is less likely to produce wrong classification.

Defining the T-neighborhood. An appropriate value for T is chosen for all distance measures $dist_x \in \mathcal{L}$ based on statistical significance testing. Specifically, we use the ANalysis Of VAriance (ANOVA) test [10], which is a generalization of the t-test when more than two groups are analyzed.

First, for each $T \in [1, \lceil N/2 \rceil]$, we construct the corresponding set of vectors $\mathcal{V}^T = \{v_{dist_1}, \ldots, v_{dist_n}\}$ for all distance measures in $\mathcal{L}$. Note that the vectors are constructed around $pos(Q_s, s'^{dist_x})$ as described in the previous section. Next, we perform ANOVA with statistical significance threshold α, which produces a p-value:

$$pval(\mathcal{V}^T, \alpha) = ANOVA(\mathcal{V}^T, \alpha) \ , \forall T \in [1, \lceil N/2 \rceil] \ . \tag{12}$$

Using Equation 12, we assign our final threshold T with the value that corresponds to the smallest p-value:

$$T = \underset{T \in [1, \lceil N/2 \rceil]}{\operatorname{argmin}} \left\{ pval(\mathcal{V}^T, \alpha) \right\} . \tag{13}$$

The rationale here is that this way of determining the T-neighborhood gives the most statistically significant NN classification accuracy results when using the set of measures in $\mathcal{L}$ and the training time series for all possible neighborhoods of Q. T is then used for the computation of the classification error

probabilities (Eq. 10), and will hence determine the most appropriate measure (Eq. 11) for the given query Q.

It has to be mentioned that if more than one T values provide the same lowest probability, the smallest one is chosen to define the query neighborhood. Furthermore, in cases where the number of training time series to the left or right of $pos(Q_s, s'^{dist_x})$ in some s'^{dist_x} array is less than $T-1$ or T, respectively, then the remaining number of objects is filled in from the opposite side, so that there are always $2 * T$ objects in each v_{dist_x} vector. This is required in order to perform ANOVA on vectors of equal size, and for all values of T.

In Figure 4 we illustrate the main steps of the proposed framework.

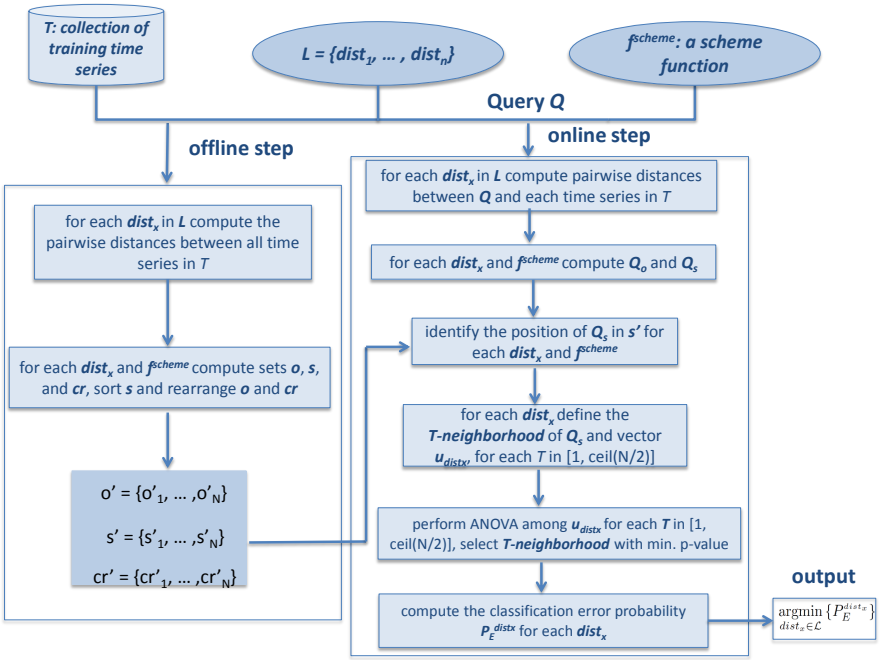

Fig. 4 Illustration of the offline and online steps of the proposed query-sensitive measure selection framework.

4.2 Methods

We present three methods that can be used within our framework for constructing the neighborhood of training objects for a given query. Each method is characterized by a *scheme*, which is a technique for computing the values in the array s and the value of Q_s used in the framework. Each scheme uses a function f^{scheme} to compute these values. Note that we first describe a *basic* scheme that is used as a building block by the two main schemes.

4.2.1 Scheme 0: Basic

This basic scheme constructs s and Q_s based exclusively on *distances*. The intuition here is the following: if a query Q is very close to its NN, then we can be more confident that Q indeed belongs to the class of its NN. If we find that Q is very close to its NN for one distance measure $dist_x$ and not for the others, then we have a reason to trust $dist_x$ more than the other measures, for the specific query. The neighborhood of a query is thus created by the training objects for which the distance from their closest object is close to the distance of the query to its closest training object.

More specifically, in the offline step, after the computation of the set of distances $\mathcal{D}^{dist_x}_{X_i,\mathcal{T}}$ of the training objects $X_i \in \mathcal{T}$ for all distance measures $dist_x \in \mathcal{L}$, we apply the scheme function f^0 as follows:

$$s_i = f^0(X_i, \mathcal{D}^{dist_x}_{X_i,\mathcal{T}}) = min(\mathcal{D}^{dist_x}_{X_i,\mathcal{T}}) \ . \tag{14}$$

In other words, the value s_i returned by the scheme function is the distance of X_i to its closest time series o_i. In a similar manner, during the online step, Q_s is computed for each $dist_x$ using the scheme function as follows:

$$Q_s = f^0(Q, \mathcal{D}^{dist_x}_{Q,\mathcal{T}}) = min(\mathcal{D}^{dist_x}_{Q,\mathcal{T}}) \ . \tag{15}$$

As it can be seen from the above equation, Q_s corresponds to the distance of Q to its closest training object. The remaining steps of the online step proceed as described in the previous section.

4.2.2 Scheme I: Distance ratio-based

This technique for constructing s is an extension of the basic scheme. It is computed by using the ratio of distance from the closest object to the distance from the next closest object with a different class. The intuition behind this scheme is this: if a query Q is much closer to its nearest neighbor, which has a certain class, than to the closest object of a different class, then we are more confident that Q indeed belongs to the class of the nearest neighbor. If this happens for one distance measure $dist_x$ and not for the others, then we have a reason to trust $dist_x$ more than the other measures, for the specific query Q.

In particular, for each X_i the scheme identifies its closest object o_i by Equation 2. Furthermore, $\mathcal{D}_{X_i,\mathcal{T}}^{dist_x}$ is sorted, resulting in $\mathcal{D}_{X_i,\mathcal{T}}^{\prime dist_x}$, and the corresponding objects are scanned until an object of a different class than that of o_i is found; let us call this object X_l and let l be the index of this object in $\mathcal{D}_{X_i,\mathcal{T}}^{\prime dist_x}$.

The value produced by the corresponding scheme function f^I is given below:

$$\begin{aligned} s_i = f^I(X_i, \mathcal{D}_{X_i,\mathcal{T}}^{dist_x}) &= f^0(X_i, \mathcal{D}_{X_i,\mathcal{T}}^{dist_x}) / d_{dist_x}(X_i, X_l) \\ &= d_{dist_x}(X_i, o_i) / d_{dist_x}(X_i, X_l) \ . \end{aligned} \tag{16}$$

Finding this ratio for all training objects and then sorting these ratios in ascending order ends up in array $s^{\prime dist_x}$.

During the online step, $\mathcal{D}_{Q,\mathcal{T}}^{dist_x}$ is sorted resulting in $\mathcal{D}_{Q,\mathcal{T}}^{\prime dist_x}$. Then, the value of Q_s is computed as the ratio of its distance from the closest training object

Q_o to its distance from the first object that has a different class than that of Q_o, say $Q_{l'}$ with l' being the index of $Q_{l'}$ in $\mathcal{D}'^{dist_x}_{Q,\mathcal{T}}$:

$$\begin{aligned} Q_s = f^I(Q, \mathcal{D}^{dist_x}_{Q,\mathcal{T}}) &= f^0(Q, \mathcal{D}^{dist_x}_{Q,\mathcal{T}}) / d_{dist_x}(Q, Q_{l'}) \\ &= d_{dist_x}(Q, Q_o) / d_{dist_x}(Q, Q_{l'}) \ . \end{aligned} \tag{17}$$

Based on the intuition for this method, ideally, if Q_s is small then its T-neighborhood will comprise objects with low ratios, and these objects (or most of them) are not misclassified. Consequently, in such a case there is higher probability of classifying the query correctly.

4.2.3 Scheme II: Homogeneity-based

Using the same notation as in the previous method, another approach is to count the number of objects from o_i to X_l and from Q_o to $Q_{l'}$, respectively. We note that objects o_i and Q_o are included in the counting, while objects X_l and $Q_{l'}$ are not. Hence, the scheme function f^{II} for a training object X_i is defined as follows:

$$s_i = f^{II}(X_i, \mathcal{D}^{dist_x}_{X_i,\mathcal{T}}) = l - 1 \ . \tag{18}$$

And similarly, for the query Q:

$$Q_s = f^{II}(Q, \mathcal{D}^{dist_x}_{Q,\mathcal{T}}) = l' - 1 \ . \tag{19}$$

So, essentially, we count the number of objects that belong to the same class as that of the NN. By doing so, we are in practice measuring the *homogeneity* of an object's neighborhood. Each s'^{dist_x} thus consists of the sorted (in ascending order) homogeneity values of all training objects under measure $dist_x$.

An ideal situation would be to obtain a high value for Q_s making it lie within several objects with high homogeneity values in s'^{dist_x}. This would

imply that all of these objects have many neighbors of the same class, and if this class is the correct one for all objects (or for most of them) then the probability of the query being classified correctly would increase.

Since our framework follows a statistical analysis based on the significance produced by ANOVA, there may be scenarios where, although Q_s may be small for the Distance ratio-based or high for the Homogeneity-based scheme in the ideal situations described above, the T-neighborhood with the highest statistical significance may be large. This means that the neighborhood of the query may even include objects that are misclassified. This fact shows that there is a tradeoff between getting a "good" value for the selected scheme and a "proper" relatively small value for T.

It has to be noted that all of the aforementioned scheme-based measure selection methods are based on the following concept: objects of the same class create regions well separated from objects of other classes. This inter-class dissimilarity, however, is not always the case, since there may be outlier objects making the classification task harder whatever the classifier may be.

5 Experiments

5.1 Experimental Setup

Datasets. We experimented on the 45 time series datasets available on the UCR time series archive [15]. The number of training and test time series ("train size", "test size"), the length of each time series ("seq. length"), and the number of classes ("class num.") for each dataset are shown in Table 2. We should note that these datasets do not include objects with missing values.

Methods. We have applied the three proposed methods, i.e., `Basic`, `Distance ratio-based`, `Homogeneity-based`, on a pool of measures $\mathcal{L}$ consisting of

DTW, ERP, and MSM (described in Sections 3.1, 3.2, and 3.3). For comparison, we also evaluate the performance of the following competitors:

- **MSM:** Using MSM as the distance measure in all datasets.
- **DTW:** Using DTW as the distance measure in all datasets.
- **ERP:** Using ERP as the distance measure in all datasets.
- **Cross Validation:** Using cross-validation to choose, separately for each dataset, which distance measure to use (among MSM, DTW, and ERP) for that dataset.

We designed the `Cross Validation` method as follows: (1) for each dataset we first computed the classification accuracy (percentage of time series correctly classified) for DTW, ERP, and MSM on the training set using leave-one-out cross validation, and (2) the method outputs the classification accuracy on the test set of the measure with the best accuracy on the training set. If more than one measures provide the same highest (best) classification accuracy on the training set, then the accuracy of `Cross Validation` is the average of the accuracies of these measures on the test set.

Regarding the parameters of the measures, MSM has one free parameter, namely c, which is the cost of every Split and Merge operation. For each of the datasets, the value for c was chosen from the set $\{0.01, 0.1, 1\}$, using leave-one-out cross-validation on the training set and comparing the three classification accuracies. In addition, it has been shown that no greater value of c may achieve good classification results, while these values are sufficient to produce very competitive classification accuracies compared to DTW and ERP [25]. For DTW the L_p used was the Euclidean distance, and the penalty g of ERP was set to 0 [7,25]. Finally, for the statistical significance testing, the significance threshold α of ANOVA was set to 0.05.

Evaluation Measures. Since the task at hand is NN classification, for each dataset, we evaluate all of the aforementioned methods in terms of *classification error rate*, which is defined as the percentage of the test time series that are misclassified using the NN classifier.

At this point we note that if in our framework there are ties among the measures in the minimum error probability (computed by Eq. 10) for a query, we select the measure with the smallest classification error rate on the training set. In case of further tie we use all measures involved in the tie, by taking the average classification result of these measures (recall that 0 corresponds to correct classification, whereas 1 to misclassification). For example, if for a query the minimum error probability is the same for both MSM and ERP, the respective error rates on the training set are the same, and MSM misclassifies the given query while ERP correctly classifies it, then the classification result for this query is considered to be 0.5.

Apart from the classification error rate, we evaluated the *efficiency* of the `Homogeneity-based` method, which is the one that yields at least as good or better classification accuracies as the baseline method on the largest number of datasets compared to the other proposed methods. In particular, we analyzed the *runtimes* for all of its parts, which are reported in Figure 6.

The framework and the proposed methods were implemented in Matlab, while DTW, MSM, and ERP were implemented in Java. The experiments were performed on a PC 64-bit running Linux, a Dual-Core AMD Opteron(tm) Processor 8220 SE at 2.8GHz using a single threaded implementation.

Training and Test Sets. For several datasets given in [15] we observed that the error rates on the training and test sets were very different for each of the distance measures, showing that the training set was not representative of the test set. For example, for the training set of the *FaceAll* dataset the

error rate for DTW was 6.79%, for MSM 1.07%, and for ERP 2.5%, while for the test set the error rates were much different, i.e., 19.23%, 18.88%, and 20.2%, respectively. For the training set of *OSU* the error rates for DTW and ERP were 33% and 30.5%, while for the test set they were 40.91% and 39.7%, respectively. Another example is the *GunPoint* dataset, for which the error rate of DTW on the training set was 18% and of ERP it was 8%, whereas for the test set it was just 9.33% for DTW and 4% for ERP. The differences are also apparent in the *Fish* dataset, where DTW had error rates 26.29% and 16.57% for the training and test set, MSM had 13.71% and 8%, and ERP achieved 17.14% and 12%.

Since measure selection is about learning statistics for each dataset, we had to be fair on selecting the train and test time series. Thus, based on the iid (independent identically distributed) criterion, for each dataset, all time series originally provided as training and test sets in [15] were merged, and then for each class half of them were randomly picked and included in the training set while the remaining ones were put in the test set. More formally, assuming that the number of time series of a class is M, $\lfloor M/2 \rfloor$ randomly selected time series comprise the training set, while the rest $M - \lfloor M/2 \rfloor$ objects form the test set. Following this procedure, and, for example, for the aforementioned datasets, the error rates achieved on the training and test sets for all measures were much closer to one another, implying that the training sets are much more representative of the test sets compared to the original split. For the *FaceAll* dataset DTW achieved 3.30% and 3.90% error rates, MSM 1.16% and 1.06%, and ERP 1.52% and 1.86%, and for the *OSU* dataset DTW had error rates 33.18% and 35.14%, and ERP 30% and 31.98% for the training and test set, respectively. With regard to the *GunPoint* dataset, the error rates for DTW were 8% for both training and test sets, and for ERP they were 2% and 3%. Finally, referring to the *Fish* dataset MSM achieved 11.43% and 10.29%

error rates for the training and test sets, DTW 25.71% and 23.43%, and ERP 17.71% and 14.29%. The sizes of the two sets for each dataset are shown in columns "train size" and "test size" of Table 2, respectively.

The datasets we used for our experiments are available here: `http://vlm1.uta.edu/~akotsif/query_sensitive_measure_selection`.

Table 2 Description of the 45 datasets from the UCR repository that were used in our experiments. The table shows for each dataset: the number of training and test objects, the length of each sequence in the dataset, and the number of classes.

ID	Dataset	train size	test size	seq. length	class num.
1	*Synthetic*	300	300	60	6
2	*GunPoint*	100	100	150	2
3	*CBF*	465	465	128	3
4	*FaceAll*	1122	1128	131	14
5	*OSU*	220	222	427	6
6	*SwedishLeaf*	555	570	128	15
7	*50Words*	442	463	270	50
8	*Trace*	100	100	275	4
9	*TwoPatterns*	2499	2501	128	4
10	*Wafer*	3582	3582	152	2
11	*FaceFour*	55	57	350	2
12	*Lightning-2*	60	61	637	2
13	*Lightning-7*	70	73	319	7
14	*ECG*	99	101	96	2
15	*Adiac*	387	394	176	37
16	*Yoga*	1650	1650	426	2
17	*Fish*	175	175	463	7
18	*Beef*	30	30	470	5
19	*Coffee*	27	29	286	2
20	*OliveOil*	29	31	570	4
21	*ChlorineConc.*	2153	2154	166	3
22	*ECG_torso*	708	712	1639	4
23	*Cricket_X*	384	396	300	12
24	*Cricket_Y*	384	396	300	12
25	*Cricket_Z*	384	396	300	12
26	*Diatom Red.*	160	162	345	4
27	*ECG5Days*	442	442	136	2
28	*FacesUCR*	1122	1128	131	14
29	*Haptics*	231	232	1092	5
30	*InlineSkate*	324	326	1882	7
31	*ItalyPower*	547	549	24	2
32	*MALLAT*	1200	1200	1024	8
33	*MedicalImages*	568	573	99	10
34	*MoteStrain*	635	637	84	2
35	*SonySurface*	310	311	70	2
36	*SonySurfaceII*	490	490	65	2
37	*StarLightC.*	4617	4619	1024	3
38	*Symbols*	508	512	398	6
39	*TwoLeadECG*	580	582	82	2
40	*uWaveGest_X*	2238	2240	315	8
41	*uWaveGest_Y*	2238	2240	315	8
42	*uWaveGest_Z*	2238	2240	315	8
43	*WordsSynon.*	450	455	270	25
44	*ECGThorax1*	1871	1894	750	42
45	*ECGThorax2*	1871	1894	750	42

5.2 Experimental Results

Classification Accuracy. The performance of the proposed methods in terms of classification accuracy is shown in Tables 3 and 4, and Figure 5.

The classification error rates of all proposed methods and competitors are shown in Table 3. For each dataset, the classification error rates for MSM, DTW, and ERP for both the training and test sets are shown, along with the c value used for MSM on that dataset. As mentioned in Section 5.1, the value of c that was chosen is the one providing the smallest classification error rate for the training set. In cases where more than one values of c provided the same error rate, in Table 3 we present all of these values, and we randomly picked the one shown in italics. All rates shown are in percent and the numbers in bold indicate the smallest error rates for each dataset when comparing the proposed methods and `Cross Validation`. The last column of Table 3 ("All Miscl.") presents the number of test time series per dataset that are not classified correctly by any distance measure. This is important, because these objects cannot be classified correctly by any distance measure selection scheme.

Table 4 summarizes the experimental results. One key conclusion is that, out of the three proposed methods, the `Homogeneity-based` method performs the best. A second key conclusion is that the `Homogeneity-based` also performs better than all four competitors.

Looking at the results of Table 4 in more detail, we first note that the DTW and ERP methods performed the weakest. All three of our proposed methods outperform DTW and ERP in at least 35 out of the 45 datasets, and these results are statistically significant with a p value less than 0.001. However, the `Basic` method performs somewhat worse than `Cross Validation` and MSM. There are actually somewhat more datasets where these competitors are more accurate than the `Basic` method, than datasets were the `Basic` method is

more accurate than these competitors. The `Distance ratio-based` method performs slightly better than `Cross Validation` and MSM. Still, the differences in accuracy between the proposed `Basic` and `Distance ratio-based` method on the one hand, and the competitors `Cross Validation` and MSM on the other hand are not statistically significant, having p values between 0.390 and 0.736.

On the other hand, the proposed `Homogeneity-based` method emerges as a clear winner. With respect to `Cross Validation`, the `Homogeneity-based` method attains lower error rates for 23 out of the 45 datasets, and higher error rates in only 10 out of the 45 datasets. With respect to MSM, the `Homogeneity-based` method achieves lower error rates for 24 out of the 45 datasets, and higher error rates in 10 out of the 45 datasets. With respect to DTW and ERP, the `Homogeneity-based` method gives lower error rates for 37 and 39 out of the 45 datasets respectively, and higher error rates in only 2 and 1 out of the 45 datasets respectively. The difference in accuracy between the `Homogeneity-based` method and each of the four competitors is statistically significant, with p values of 0.022 with respect to `Cross Validation`, 0.015 with respect to MSM, and less than 0.001 with respect to DTW and ERP.

These experimental results also emphasize the importance of choosing the scheme function for our framework. There is a significant difference in performance between the best performing scheme (`Homogeneity-based`) and the other two schemes (`Basic` and `Distance ratio-based`). This result also points to a direction for future work, namely to design schemes performing even better than the `Homogeneity-based` scheme.

In Table 5 we present the probabilities that are the outcome of ANOVA when the input vectors are the classification results of each of the proposed methods against `Cross Validation` (denoted as "C.V.") for all test time series. For completeness, we also present the probabilities when comparing the

classification results of `Basic`, `Distance ratio-based`, and `Homogeneity-based` with each of the distance measures (MSM, DTW, ERP). It has to be mentioned that, the lower the probabilities are, the more different the vectors of classification results are. When the probabilities are high the vectors are very similar, and when the probabilities are 1 we can conclude that the vectors are identical, as it happens, for example, between each of the proposed methods and `Cross Validation` for datasets with IDs 8, 9, and 10, as the corresponding error rates are 0.00%, 0.00%, and 0.28%, respectively. In other words, the same distance measure may be selected (for all test time series) by each of the proposed methods and `Cross Validation`, e.g., for datasets 8 and 10 DTW and MSM is selected, respectively, by the proposed methods and `Cross Validation`. Note that for dataset 9 all measures provide 0.00% error rate in the training and test sets, hence all methods yield the same vectors of classification results.

Runtime. In Figure 6, for each dataset, we present the average runtimes of a query for all parts of the `Homogeneity-based` method: MSM, DTW, ERP Computation, selecting the "best" T ("Homogeneity-based scheme"), and determining the most suitable distance measure ("Remaining Time"). Adding up the runtimes of all parts gives the total time, which is also shown.

It can be clearly seen that, for any dataset, the bottleneck of total runtime is the computation of distances for the three measures. DTW takes more time than MSM and ERP for all datasets, which is obvious, e.g., for datasets 22, 30, 32, 37, 44, and 45. The reason for the higher total runtime per query for datasets 16, 22, 29, 30, 32, 37, 40-42, and 44, 45 is the number of their training time series to which the MSM, DTW, and ERP distances have to be evaluated for the query, and also the large length of their time series (as Table 2 indicates).

Table 3 Nearest Neighbor classification error rates attained by the three proposed methods (denoted as `Dist-Ratio`, `Hom.`, and `Basic`), and `Cross Validation` on each of the 45 datasets in the UCR repository of time series datasets. In addition, the table shows for each dataset: the classification error rate of MSM, DTW, and ERP for the training and test sets and the value of c used by MSM on that dataset, which yielded the lowest error rate on the training set (when more than one values are given, the one in italics was randomly chosen). All rates are in percent and the numbers in bold indicate the smallest classification error rates for each dataset when comparing the three proposed methods and `Cross Validation`. The number of test time series per dataset that are misclassified by all distance measures is also provided in the last column.

ID	Basic	Dist. Ratio	Hom.	Cross Valid.	train			test			MSM	All Miscl.
					MSM	DTW	ERP	MSM	DTW	ERP	c	
1	1.67	**1.00**	**1.00**	2.00	1.67	1.33	1.00	1.33	1.00	2.00	*0.1*,1	0
2	3.00	3.00	**2.00**	3.00	3.00	8.00	2.00	0.00	8.00	3.00	0.1	0
3	**0.00**	**0.00**	**0.00**	**0.00**	0.43	0.00	0.00	0.22	0.00	0.00	0.1	0
4	1.15	1.15	**1.06**	**1.06**	1.16	3.30	1.52	1.06	3.90	1.86	1	10
5	**19.37**	20.72	**19.37**	**19.37**	15.91	33.18	30.00	19.37	35.14	31.98	0.1	26
6	11.58	11.93	**11.05**	11.58	12.25	21.44	12.97	11.58	20.70	12.63	1	36
7	19.44	**18.36**	20.73	19.22	20.36	35.29	29.86	19.22	33.05	28.51	1	59
8	**0.00**	**0.00**	**0.00**	**0.00**	4.00	0.00	17.00	5.00	0.00	11.00	0.1	0
9	**0.00**	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	1	0
10	**0.28**	**0.28**	**0.28**	**0.28**	0.11	0.50	0.20	0.28	0.78	0.36	1	8
11	11.98	9.65	7.89	**7.60**	1.82	1.82	1.82	3.51	14.04	5.26	0.01,0.1,*1*	2
12	11.48	**8.20**	13.11	14.75	16.67	15.00	13.33	18.03	14.75	14.75	*0.01*,1	2
13	28.08	**19.18**	20.55	27.40	32.86	25.71	32.86	26.03	27.40	34.25	0.01,*1*	10
14	19.80	**16.83**	17.33	**16.83**	9.09	14.14	9.09	13.86	21.78	19.80	1	9
15	38.32	**36.80**	**36.80**	39.34	35.66	39.28	37.73	39.34	38.83	40.86	1	116
16	4.85	5.15	4.97	**4.79**	4.42	8.67	5.58	4.79	8.67	5.88	0.1	49
17	10.86	10.86	**9.71**	10.29	11.43	25.71	17.71	10.29	23.43	14.29	1	13
18	56.67	**53.33**	**53.33**	**53.33**	60.00	63.33	60.00	53.33	60.00	53.33	1	16
19	17.24	20.69	**13.79**	17.24	22.22	22.22	25.93	20.69	13.79	20.69	0.01	2
20	**6.45**	9.68	12.90	12.90	20.69	24.14	24.14	12.90	9.68	12.90	0.01	2
21	7.06	7.15	7.29	**7.01**	7.71	5.67	7.85	7.71	7.01	7.85	0.1,*1*	130
22	**0.14**	**0.14**	**0.14**	**0.14**	0.00	2.54	0.14	0.14	2.67	1.12	1	0
23	23.99	**21.21**	23.74	23.99	22.14	17.97	23.70	24.75	23.99	25.51	1	60
24	21.46	**17.17**	18.94	21.97	22.92	17.45	23.70	19.19	21.97	19.70	1	41
25	18.69	17.68	**15.15**	18.69	27.60	20.57	28.91	22.98	18.69	23.23	1	44
26	**0.62**	**0.62**	**0.62**	**0.62**	0.00	0.63	0.63	0.62	1.23	1.23	1	1
27	**0.68**	0.90	1.02	**0.68**	0.45	1.13	1.13	0.68	1.58	2.26	0.01,*1*	2
28	0.89	0.98	**0.80**	0.89	1.16	3.48	1.87	0.89	3.19	1.24	1	7
29	**52.16**	55.60	**52.16**	**52.16**	55.41	57.14	58.01	52.16	62.50	55.17	1	84
30	43.87	45.09	**42.64**	42.94	45.06	50.93	45.99	42.94	53.99	44.17	1	104
31	3.64	3.73	**3.46**	3.74	4.94	5.67	4.94	3.28	4.92	4.19	1	13
32	**1.17**	**1.17**	**1.17**	1.42	0.10	2.25	0.75	1.42	1.50	1.17	1	4
33	19.37	**18.15**	19.02	19.02	20.60	26.06	24.47	19.02	21.12	21.47	0.1	60
34	3.30	**2.98**	3.14	3.30	2.52	5.51	3.31	3.30	5.49	4.08	0.1	7
35	0.96	0.64	0.96	**0.32**	2.26	4.52	2.90	0.32	1.61	1.61	1	1
36	**0.41**	**0.41**	0.61	**0.41**	1.63	4.29	3.67	0.41	2.45	1.63	1	1
37	6.47	**5.93**	6.49	6.56	8.43	7.06	10.48	8.34	6.56	10.11	0.1	109
38	1.56	1.56	**1.17**	1.37	1.18	1.97	1.77	1.37	1.95	2.34	0.01,*0.1*	4
39	**0.17**	**0.17**	**0.17**	**0.17**	0.00	0.00	0.34	0.17	0.17	0.34	0.01	1
40	21.88	**20.40**	20.67	21.03	21.27	24.66	23.06	21.03	26.52	22.05	1	291
41	28.13	27.90	**26.74**	27.86	28.06	35.17	31.95	27.86	35.18	30.71	1	242
42	27.05	25.85	**25.71**	26.43	27.97	32.98	28.55	26.43	31.03	29.55	1	331
43	**18.46**	19.78	19.78	19.12	19.33	32.67	28.00	19.12	31.43	25.05	1	55
44	16.26	15.73	**15.10**	17.16	19.08	20.36	18.12	18.06	20.91	17.16	1	145
45	10.61	9.87	**9.82**	10.72	10.69	14.27	12.03	10.72	13.73	10.82	1	96

Comparing the proposed methods, since the framework is the same for all of them, the average DTW, MSM, and ERP distance computation times for all queries of each dataset are essentially the same for all methods. In addition,

Table 4 Number of datasets for which each method yields lower (*Better*), equal (*Tie*), or higher (*Worse*) classification error rate compared to always choosing one distance measure, i.e., MSM, DTW, or ERP, and compared to Cross Validation (denoted as "C.V."). For completeness, the p-value of the ANOVA test is also presented.

	Basic				Distance ratio-based				Homogeneity-based			
	MSM	DTW	ERP	C.V.	MSM	DTW	ERP	C.V.	MSM	DTW	ERP	C.V.
Better	14	35	37	11	20	37	36	18	24	37	39	23
Tie	12	6	5	18	8	6	3	11	11	6	5	12
Worse	19	4	3	16	17	2	6	16	10	2	1	10
p-value	.390	< .001	< .001	.341	.627	< .001	< .001	.736	.015	<.001	< .001	.022

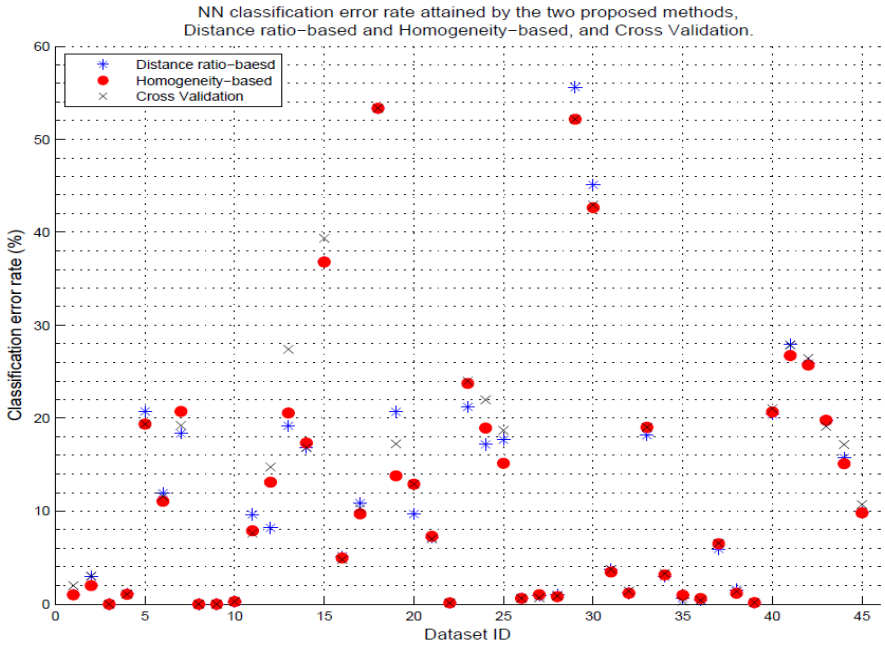

Fig. 5 Nearest Neighbor classification error rates attained by `Distance ratio-based` and `Homogeneity-based`), and `Cross Validation` on each of the 45 datasets in the UCR repository of time series datasets.

the "Remaining Time" part is negligible for all of them. As a result, the only part that basically differentiates the `Homogeneity-based` method and the rest is the procedure for finding the "best" value for T. However, since the total runtime is dominated by the distance computations, all proposed methods have approximately the same total runtime per query.

Finally, regarding the baseline `Cross Validation` method, the total runtime for a query depends on whether there is a clear distance measure winner

Table 5 The probabilities that are the outcome of the ANOVA statistical test when the input vectors are the classification results (for all test time series) of each of the proposed methods against MSM, DTW, ERP, and Cross Validation (denoted as "C.V.") are presented for each of the 45 datasets.

ID	Basic				Dist. Ratio				Hom.			
	MSM	DTW	ERP	C.V.	MSM	DTW	ERP	C.V.	MSM	DTW	ERP	C.V.
1	0.656	1	0.083	0.083	0.656	1	0.083	0.083	0.706	0.480	0.318	0.318
2	0.083	0.058	1	1	0.158	0.033	0.320	0.320	0.083	0.058	1	1
3	0.318	1	1	1	0.318	1	1	1	0.318	1	1	1
4	0.318	0.000	0.021	0.318	1	0.000	0.013	1	0.318	0.000	0.021	0.318
5	0.180	0.000	0.000	0.180	1	0.000	0.000	1	1	0.000	0.000	1
6	0.564	0.000	0.528	0.564	0.257	0.000	0.160	0.257	0.000	0.000	0.377	0.000
7	0.318	0.000	0.000	0.318	0.090	0.000	0.000	0.090	0.318	0.000	0.000	0.318
8	0.025	1	0.001	1	0.025	1	0.001	1	0.025	1	0.001	1
9	1	1	1	1	1	1	1	1	1	1	1	1
10	1	0.000	0.083	1	1	0.000	0.083	1	1	0.000	0.083	1
11	0.018	0.024	0.024	0.022	0.058	0.034	0.083	0.058	0.019	0.164	0.035	0.073
12	0.033	0.159	0.103	0.103	0.321	0.742	0.709	0.709	0.209	0.419	0.419	0.419
13	0.167	0.013	0.004	0.013	0.288	0.024	0.017	0.024	0.683	0.863	0.236	0.863
14	0.158	0.158	0.057	0.108	0.127	0.251	0.227	0.177	0.033	0.508	1	0.517
15	0.025	0.268	0.005	0.025	0.018	0.277	0.003	0.018	0.415	0.778	0.086	0.415
16	0.157	0.000	0.109	0.157	0.318	0.000	0.047	0.318	0.564	0.000	0.027	0.564
17	0.319	0.000	0.083	0.319	0.319	0.000	0.011	0.319	0.319	0.000	0.083	0.319
18	0.000	0.161	0.000	0.000	0.000	0.161	0.000	0.000	0.326	0.326	0.326	0.326
19	1	0.326	1	0.663	0.326	1	0.326	0.663	0.326	0.663	0.573	0.494
20	0.325	1	0.325	0.325	1	0.572	1	1	0.161	0.325	0.161	0.161
21	0.103	0.083	0.047	0.083	0.199	0.034	0.096	0.034	0.039	0.763	0.015	0.763
22	1	0.000	0.008	1	1	0.000	0.008	1	1	0.000	0.008	1
23	0.019	0.041	0.006	0.041	0.528	0.835	0.275	0.835	0.681	1	0.415	1
24	0.249	0.001	0.086	0.001	0.882	0.028	0.640	0.028	0.250	0.564	0.371	0.564
25	0.003	0.372	0.001	0.372	0.000	0.003	0.000	0.003	0.038	1	0.022	1
26	0.000	0.319	0.319	0.000	0.000	0.319	0.319	0.000	0.000	0.319	0.319	0.000
27	0.318	0.180	0.014	0.318	0.180	0.276	0.016	0.180	1	0.045	0.008	1
28	0.318	0.000	0.318	0.318	0.318	0.000	0.096	0.318	1	0.000	0.206	1
29	0.183	0.021	0.862	0.183	1	0.001	0.179	1	1	0.001	0.287	1
30	0.209	0.000	0.675	0.209	0.848	0.000	0.476	0.848	0.565	0.000	0.876	0.565
31	0.318	0.042	0.318	0.318	0.655	0.021	0.158	0.406	0.415	0.071	0.083	0.249
32	0.083	0.414	1.000	0.083	0.083	0.414	1	0.083	0.083	0.414	1	0.083
33	0.370	0.044	0.010	0.370	1.000	0.163	0.052	1	0.528	0.292	0.128	0.528
34	0.480	0.002	0.162	0.480	0.763	0.005	0.201	0.763	1	0.008	0.336	1
35	0.318	0.083	0.180	0.318	0.158	0.158	0.415	0.158	0.158	0.318	0.158	0.158
36	0.000	0.004	0.034	0.000	0.318	0.007	0.096	0.318	0.000	0.004	0.034	0.000
37	0.000	0.013	0.000	0.013	0.000	0.799	0.000	0.799	0.000	0.317	0.000	0.317
38	0.318	0.318	0.249	0.318	0.564	0.045	0.058	0.564	0.318	0.415	0.206	0.318
39	0.000	0.000	0.318	0.000	0.000	0.000	0.318	0.000	0.000	0.000	0.318	0.000
40	0.201	0.000	0.008	0.201	0.530	0.000	0.005	0.530	0.059	0.000	0.775	0.059
41	0.938	0.000	0.000	0.938	0.093	0.000	0.000	0.093	0.109	0.000	0.003	0.109
42	0.339	0.000	0.000	0.339	0.206	0.000	0.000	0.206	0.253	0.000	0.000	0.253
43	0.366	0.000	0.001	0.366	0.318	0.000	0.001	0.318	0.180	0.000	0.000	0.180
44	0.002	0.000	0.028	0.028	0.000	0.000	0.003	0.003	0.016	0.000	0.116	0.116
45	0.046	0.000	0.080	0.046	0.041	0.000	0.056	0.041	0.527	0.000	0.717	0.527

in the error rate on the training set or not. In case that one of the three measures provides the smallest classification error rate on the training set, then this measure is evaluated. Thus, given a query of e.g., dataset with ID x, the total runtime is practically given by the point of the curve corresponding to this measure at position x (of the horizontal axis) in Figure 6. Similarly, if more than one measures attain the lowest classification error rate, then all

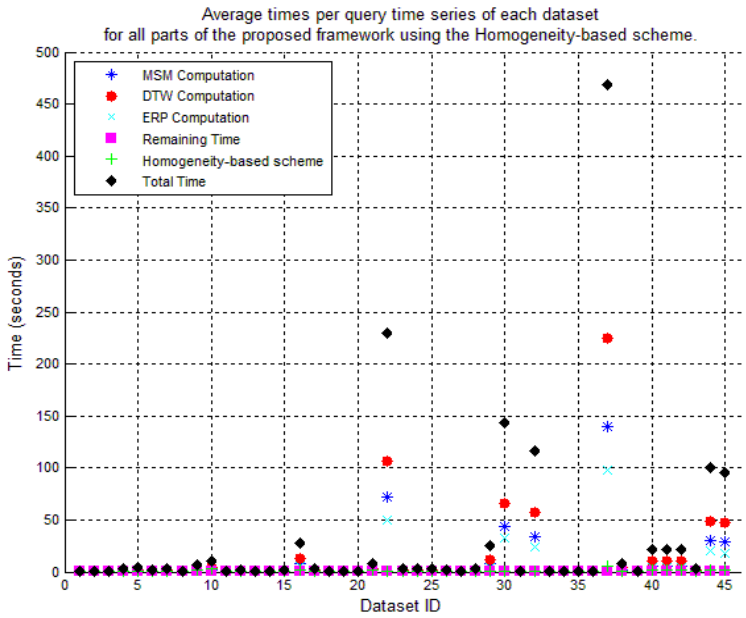

Fig. 6 Average runtimes per query, for each dataset, for all parts of the Homogeneity-based measure selection method: the computations of MSM, DTW, ERP, Homogeneity-based scheme, and the final measure selection part. The total average runtime per query for each dataset (summing up all of the above parts) is also shown.

these measures contribute to the total runtime for a query, which is the sum of the distance computation times of the query to all training time series for the tied measures. Hence, if, in the worst case, all measures in the pool need to be evaluated for the query, then the total runtime of `Cross Validation` is approximately the same as that of our methods.

An astute reader may argue that the runtime comparison of `Homogeneity-based` against `Cross Validation` is unfair since we could alternatively have used existing speedup methods for DTW [3], or even powerful pruning techniques such as cDTW with the LB_Keogh lower bound [14]. Nonetheless, we argue that any speedup achieved by each method used by `Cross Validation` is also equally beneficial for the methods that are based on our framework. This is due to the fact that our framework is using the exact same set of dis-

tance measures, and thus any speedup obtained by `Cross Validation` can essentially be exploited by our framework as well.

6 Conclusions and Future Work

In this paper we studied the problem of selecting, given a query, the most appropriate time series distance measure out of a pool of such measures, so as to perform time series NN classification. We demonstrated that the problem is important and challenging, and proposed a novel framework to solve it, based on the behavior of each distance measure on what we call the T-neighborhood of the query, i.e., a set of training objects similar to the query according to some property. Based on this framework, we proposed three specific schemes for identifying the T-neighborhood of the query, namely the Distance-based, Distance ratio-based, and Homogeneity-based schemes.

In our experiments, the proposed Homogeneity-based method produced state-of-the-art performance. Based on classification error rates on 45 datasets, the Homogeneity-based method outperformed all competitors (namely, the individual distance measures DTW, MSM, and ERP, as well as the standard measure-selection method of Cross Validation) in a statistically significant manner.

For future work, we plan to investigate adding additional measures to the set of measures used in our pool. We believe that including additional measures will make our methods achieve even smaller classification error rates. Finally, we shall explore new schemes for defining the T-neighborhood, as well as alternative statistical tests.

Acknowledgements This work was partially supported by National Science Foundation grants IIS-1055062, CNS-1059235, CNS-1035913, and CNS-1338118.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: In Proc. of FODO, pp. 69–84. Springer Verlag (1993)
2. Athitsos, V., Hadjieleftheriou, M., Kollios, G., Sclaroff, S.: Query-sensitive embeddings. ACM Trans. Database Syst. **32**(2) (2007)
3. Athitsos, V., Papapetrou, P., Potamias, M., Kollios, G., Gunopulos, D.: Approximate embedding-based subsequence matching of time series. In: SIGMOD, pp. 365–378 (2008)
4. Bellman, R.: The theory of dynamic programming. Bull. Amer. Math. Soc. **60**(6), 503–515 (1954)
5. Bergroth, L., Hakonen, H., Raita, T.: A survey of longest common subsequence algorithms. In: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00), SPIRE '00, pp. 39– (2000)
6. Bollobás, B., Das, G., Gunopulos, D., Mannila, H.: Time-series similarity problems and well-separated geometric sets. In: Symposium on Computational Geometry, pp. 454–456 (1997)
7. Chen, L., Ng, R.: On the marriage of l_p-norms and edit distance. In: VLDB, pp. 792–803 (2004)
8. Chen, L., Özsu, M.T.: Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp. 491–502 (2005)
9. Chen, Y., Nascimento, M.A., Chin, B., Anthony, O., Tung, K.H.: Spade: On shape-based pattern detection in streaming time series. In: Proceedings of the IEEE International Conference of Data Engineering (ICDE), pp. 786–795 (2007)
10. Cox, D.: Principles of Statistical Inference. Cambridge University Press (2006)
11. Domeniconi, C., Peng, J., Gunopulos, D.: Locally adaptive metric nearest-neighbor classification. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(9), 1281–1285 (2002)
12. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. Pattern Analysis and Machine Intelligence, IEEE Transactions on **18**(6), 607–616 (1996)
13. Hu, B., Chen, Y., Keogh, E.: Time series classification under more realistic assumptions. In: SDM, pp. 578–586 (2012)
14. Keogh, E.: Exact indexing of dynamic time warping. In: VLDB, pp. 406–417 (2002)
15. Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L., Ratanamahatana, C.: The UCR time series classification/clustering homepage: www.cs.ucr.edu/~eamonn/time_series_data/ (2011)

16. Kotsifakos, A., Papapetrou, P., Hollmén, J., Gunopulos, D.: A subsequence matching with gaps-range-tolerances framework: A query-by-humming application. PVLDB **4**(11), 761–771 (2011)
17. Kruskall, J.B., Liberman, M.: The symmetric time warping algorithm: From continuous to discrete. In: Time Warps. Addison-Wesley (1983)
18. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics **10**(8), 707–710 (1966)
19. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD '03, pp. 2–11 (2003)
20. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. J. Intell. Inf. Syst. **39**(2), 287–315 (2012)
21. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12, pp. 289–297 (2012)
22. Marteau, P.F.: Time warp edit distance with stiffness adjustment for time series matching. Pattern Analysis and Machine Intelligence, IEEE Transactions on **31**(2), 306–318 (2009)
23. Mueen, A., Keogh, E., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, pp. 1154–1162 (2011)
24. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. Transactions on ASSP **26**, 43–49 (1978)
25. Stefan, A., Athitsos, V., Das, G.: The move-split-merge metric for time series. Transactions on Knowledge and Data Engineering (2012)
26. Unal, E., Chew, E., Georgiou, P., Narayanan, S.: Challenging uncertainty in query by humming systems: a fingerprinting approach. Transactions on Audio Speech and Language Processing **16**(2), 359–371 (2008)
27. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.J.: Experimental comparison of representation methods and distance measures for time series data. Data Minining and Knowledge Discovery **26**(2), 275–309 (2013)
28. Xing, Z., Pei, J., Yu, P.S., Wang, K.: Extracting interpretable features for early classification on time series. In: SDM, pp. 247–258 (2011)
29. Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min. Knowl. Discov. **22**(1-2), 149–182 (2011)