Linear Time Complexity Time Series Classification with Bag-of-Pattern-Features

Xiaosheng Li, Jessica Lin
Department of Computer Science, George Mason University, Fairfax, VA, USA
Email: xli22@gmu.edu, jessica@gmu.edu

***Abstract*—Time series classification has attracted much attention due to the ubiquity of time series. With the advance of technologies, the volume of available time series data becomes huge and the content is changing rapidly. This requires time series data mining methods to have low computational complexities. In this paper, we propose a parameter-free time series classification method that has a linear time complexity. The approach is evaluated on all the 85 datasets in the well-known UCR time series classification archive. The results show that the new method achieves better overall classification accuracy performance than the widely used benchmark, i.e. 1-nearest neighbor with dynamic time warping, while consuming orders of magnitude less running time. The proposed method is also applied on a large real-world bird sounds dataset to verify its effectiveness.**

I. Introduction

The mining of time series data has drawn much interest as time series are everywhere in our daily lives. With the advance of technologies, for example the sensors are becoming more and more accurate and cheaper, large volumes of time series data exist in almost every discipline. This requires the data mining methods to have low computational complexities to handle the large data size.

Time series classification can be formulated as follows: given a set of labeled time series, predict the class labels of previously unknown instances. It is one of the most important research topics in time series data mining. There has been a lot of work on time series classification and most of the effort focuses on improving the classification accuracy. Recently, in [1] the authors conduct a comprehensive experimental comparison of the state-of-the-art time series classification methods, and all of them have super-linear time complexities. The best performing method in accuracy is the Collective of Transformation-based Ensembles (COTE) [2]. This method is an ensemble of 35 classifiers on different data transformations and has a time complexity of $O(n^2m^4)$, where n is the number of training time series instances and m is the length of the time series. This high time complexity makes it inapplicable to large data or real-time analytics where the model may need to be changed frequently.

Although there exists a wealth of work on time series classification, little work focuses on providing a linear time complexity solution. In this paper, we propose a parameter-free Bag-Of-Pattern-Features (BOPF) method for time series classification that has a linear time complexity on the number of training instances and on the length of time series ($O(nm)$). The new approach transforms the time series into the Bag-Of-Patterns representation [3] and considers the patterns in the representation as new features. Then a one-sweep incremental cross validation process is designed to select a subset of the features for effective classification. The centroids and term frequency-inverse document frequency (tf-idf) [4] weights of the new selected features are generated for fast classification.

The new BOPF method is evaluated on all the 85 datasets in the well-known UCR time series classification archive [5], and the results are compared with those of a widely used benchmark method, i.e. 1-nearest neighbor with Dynamic Time Warping (DTW) [6], which has a super-linear time complexity. For DTW, we use the state-of-the-art implementation, UCR-Suite, which is optimized for speed [7].

The experimental results show that the new approach has better overall accuracy performance than DTW while achieving orders of magnitude speedup in actual running time. Further experiments are carried out to verify the effectiveness of the designed mechanisms in the new method.

Xeno-canto [8] is a website that enables sharing of bird sounds across the world. People around the world record the bird sounds in the wild and upload the data to the website. Currently, the website has 359960 recordings covering 9742 species. We download 15 gigabytes of audio files from the website and use the Mel-Frequency Cepstral Coefficients (MFCCs) [9] to transform the audio files into time series data. The resulting time series dataset is much larger than any of the datasets in the UCR time series classification archive, and it is one of the largest datasets for time series classification to date. Using the proposed BOPF method, we achieve competitive classification results in very short time on this large dataset.

The rest of the paper is organized as follows. Section II provides the background and related work. Section III gives the details of our BOPF method. The experimental evaluation of the new method is presented in Section IV. Section V conducts a case study of the new method on the bird sounds data. Finally Section VI concludes the paper.

II. Background and Related Work

A. Definitions and Notations

This subsection lists the definitions and notations used in the paper to precisely describe the problem at hand and present the new method.

Definition 1 Time Series T

A time series T is a sequence of ordered real values $[t_1, t_2, \ldots, t_m]$, where m is the length of the time series.

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.37

Definition 2 Subsequnce S

A subsequence S of length l of time series T is a sequence of contiguous values taken from T: $S = [t_i, t_{i+1}, \ldots, t_{i+l-1}]$, where $1 \leq i \leq m-l+1$ and $1 \leq l \leq m$. All the subsequences of a certain length from a time series can be extracted by sliding a window of the same length from the start point of the time series to the $(m-l+1)$th point.

Definition 3 Time Series Dataset D

A time series dataset D is a set of time series T_i with their respective class labels y_i, where $i = 1, 2, \ldots, n$ and n is the number of instances in the set. Let r denote the class value so $y_i = r_j$, $j \in \{1, \ldots, c\}$. c is the number of classes and usually $c \ll n$. For convenience let St_C denote the set of time series having the same class label C.

B. Related Work

There has been a great amount of work on time series classification. The state-of-the-art techniques can be categorized in two groups. One group consists of whole series based methods. Most of the methods in this group adopt the 1-nearest neighbor classifier with a certain distance measure for the comparison of time series. Existing research focuses on developing and applying alternative distance measures. Euclidean Distance (ED) is a classic distance metric for time series that has a linear computational time complexity. However, this measure does not consider the misalignment and phase-shift in the data, which are common phenomena in real-world time series. Thus, its classification performance is often inferior compared with other, elastic distance measures.

A widely used elastic distance measure is Dynamic Time Warping (DTW) [6]. It applies a dynamic programming process on the two time series under comparison to find an optimal alignment between the values in the two series. The 1-nearest neighbor classifier combined with DTW is regarded as a hard-to-beat standard baseline [10]. The drawback of DTW is its quadratic time complexity, thus some speed-up techniques are developed. The UCR-Suite [7] leverages early abandoning and cascading lower bounds to speedup the computation of DTW. Other alternative distance measures include Weighted DTW [11], Time Warp Edit [12], and Longest Common SubSequence (LCSS) [13].

Another group of time series classification approaches are short-pattern based. This type of methods extract some short patterns from the time series and use them for classification. Some methods in this group are based on the concept of time series shapelets [14]–[18].

A shapelet is a subsequence in the time series that can best discriminate different classes [14]. In [14], all subsequences in the time series dataset are extracted as candidates and the distances from a candidate to all the time series are calculated. The subsequence with a maximum information gain on splitting the distance values of different classes is regarded as the shapelet. The shapelet is then embedded in a decision tree classifier as the split test for classification. In [16], shapelets are selected using ANOVA F values and in [17], shapelets are learned via optimizing a classification objective function.

Some other methods in the short-pattern group are based on the Bag-Of-Patterns (BOP) [3] representations. The BOP employs a sliding window to extract all the subsequences in the time series and transforms them into symbolic patterns (words). A histogram of the symbolic patterns is built to represent the high-level structure of the time series. Methods based on BOP include BOSS [19] and WEASEL [20].

Some approaches are proposed to promote the scalability of time series classification. Fast Shapelet [21] method transforms the real-value time series into a low-dimension symbolic representation using Symbolic Aggregation approXimation (SAX) [22]. A random projection and hashing process is devised to find the promising shapelet candidates in the symbolic space. In SAX-VSM [23] and BOSS VS [24], the term frequency-inverse document frequency (tf-idf) weighting scheme is used to speed up the classification process. Instead of building a BOP model for each time series, they construct one model for one class of time series, thus improving the speed. These methods show quite significant speed-up, but still none of them has a linear time complexity.

C. Bag-Of-Patterns Representation

Since our method is based on the BOP representation, this subsection briefly introduces this model. Figure 1 provides an example of transforming a real-valued time series into the BOP representation.

In Figure 1 (Left), first the subsequences (red) are extracted from the time series (blue) using a sliding window. In Figure 1 (Center) each of the subsequences is transformed into a symbolic pattern (SAX word) using the Symbolic Aggregation approXimation (SAX) technique [22]. Concretely, the subsequence is z-normalized and divided into ω segments (2 in our case). The mean values of each segment (the magenta and cyan horizontal line in the figure respectively) are calculated.

These mean values are mapped to symbols according to a set of break points (the grey lines in the figure) that divide the values into equal-probable regions. In our case the alphabet size α of the SAX transformation is 4 (with an alphabet of 'a', 'b', 'c' and 'd') and the subsequence is thus transformed to the word "bc".

In Figure 1 (Right) the counting of all the words forms a histogram that is the BOP representation for the time series. In BOP, as the window slides across the time series, a sequence of words is generated. Only the first encounter of a certain word in the sequence is counted and the following duplicate words are ignored. This process is called numerosity reduction [3]. The sliding window length (subsequence length) l, the number of segments (word length) ω and the alphabet size α are parameters supplied by the user.

There are several advantages of using the BOP model over the raw time series. The averaging and symbolization process in BOP can filter out the noises in the time series, making the model robust. The original positions of the symbolic patterns

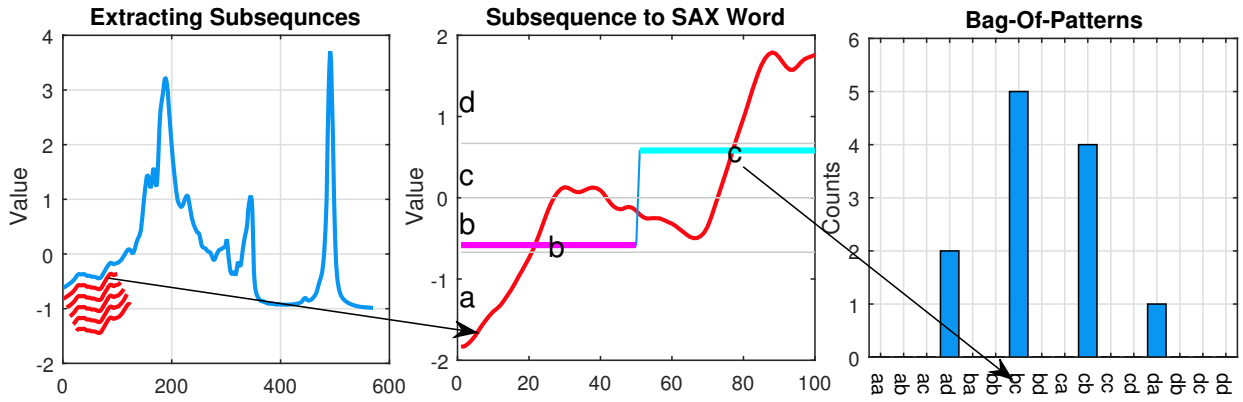

Fig. 1. (Left) Extracting subsequences (red) from a time series (blue). (Center) Transforming a subsequence to a symbolic pattern (word). (Right) Building a histogram of the words from the time series.

are not preserved in the model, thus the model provides invariance to the phase shifts in the time series.

III. Bag-Of-Pattern-Features

For clarity we first present the idea of our Bag-Of-Pattern-Features (BOPF) method with a concrete small example. Then the formal description of the algorithm will be given.

A. A Concrete Illustrative Example

Figure 2 shows the time series classification process on a dataset of 4 instances from 2 different classes. The time series are taken from the CBF dataset from the UCR time series classification archive. Two of the time series (blue) are from the class Bell (Figure 2 Left, the first two rows) and two (red) are from the class Funnel (Figure 2 Left, the last two rows).

In BOPF, the time series are first transformed to the BOP representations (Figure 2 Center). Each of the word in the BOP is regarded as a feature and the respective word count becomes the feature value. Then the centroids and term frequency-inverse document frequency (tf-idf) weights can be calculated (Figure 2 Right).

Let $Ct_T(w)$ denote the count of a word "w" in the BOP model of a time series T. For a given class C, $Ct_C(w)$ is the sum of $Ct_T(w)$ for all $T \in St_C$. Then the centroid value of a word "w" of a class C is $centroid(w, C)$:

$$centroid(w, C) = \frac{Ct_C(w)}{|St_C|} \tag{1}$$

The class centroid of a class C is $centroid(C)$:

$$centroid(C) = (centroid(w_1, C), \ldots, centroid(w_p, C)) \tag{2}$$

where "w_p" is the pth word in the BOP representation. The approach in [23] [24] is adopted to calculate the tf-idf weights. The term frequency $tf(w, C)$ of a word "w" of a class C is given by:

$$tf(w, C) = \begin{cases} 1 + \log(Ct_C(w)), & \text{if } Ct_C(w) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The inverse document frequency $idf(w)$ measures the uniqueness of the word "w" belonging to a certain class:

$$idf(w) = \log(1 + \frac{c}{wc}) \tag{4}$$

where c is the total number of classes and wc is the number of classes that contain the word "w". The tf-idf weight $tfidf(w, C)$ of a word "w" of a class C is defined as:

$$tfidf(w, C) = tf(w, C) \cdot idf(w) \tag{5}$$

The tf-idf weight of a class C is $tfidf(C)$:

$$tfidf(C) = (tfidf(w_1, C), \ldots, tfidf(w_p, C)) \tag{6}$$

where "w_p" is the pth word in the BOP representation.

In the 1-nearest neighbor method, an unlabeled test instance is compared with all the n labeled instances in the training set, thus having a complexity of $O(n)$ (assuming length is not considered here), whereas in BOPF, a test instance is compared with all the c class centroids and tf-idf weights to determine its predicted labels. Since $c \ll n$, the complexity is reduced to $O(c) = O(1)$.

In the comparison between a BOP vector Q and a class centroid vector S of length p, the squared Euclidean distance is applied to measure their distance $D(Q, S)$:

$$D(Q, S) = \sum_{i=1}^{p} (q_i - s_i)^2 \tag{7}$$

The Squared Cosine similarity is adopted to compare a vector Q and a tf-idf weight S:

$$SC(Q, S) = (\frac{Q \cdot S}{|Q||S|})^2 = \frac{(\sum_{i=1}^{p} q_i s_i)^2}{\sum_{i=1}^{p} q_i \sum_{i=1}^{p} s_i} \tag{8}$$

In BOPF, instead of using all the features for classification, only a subset of features are selected. A one-sweep cross validation process is designed to select the features. First for each of the features, the ANOVA F value [25] is calculated. The result for our example is shown in Figure 3.

The ANOVA F value of a feature equals the mean squared variance of the feature values among different classes divided

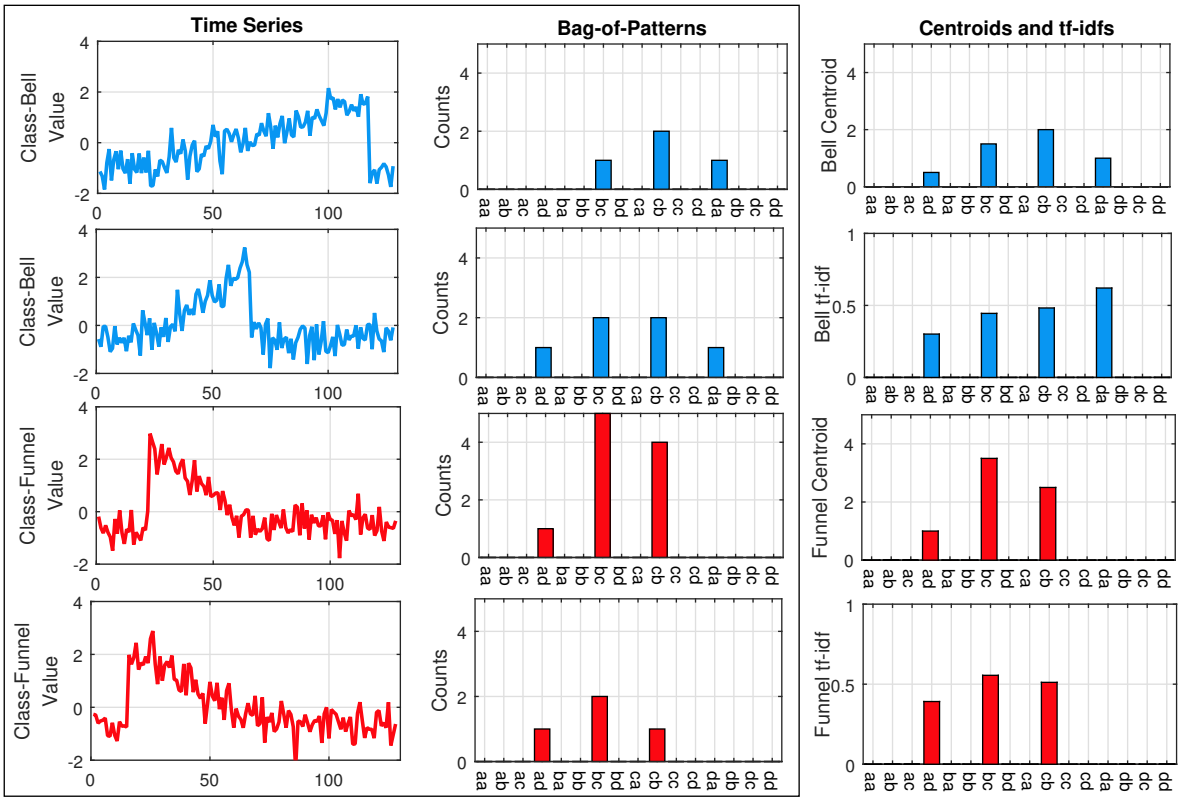

Fig. 2. An illustrative example of the BOPF classification process. (Left) Time series from 2 classes. (Center) BOP representations of the time series on the left. (Right) The centroids and tf-idf weights of the two classes generated from the BOP representations.

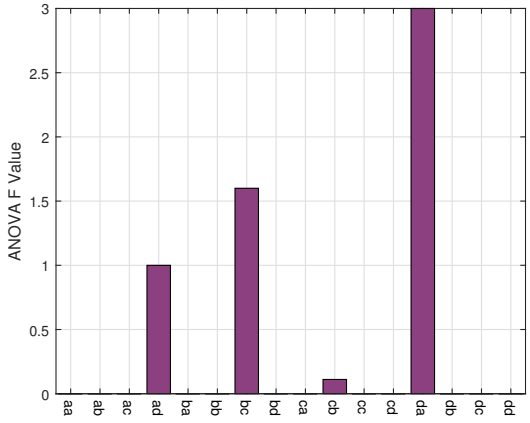

Fig. 3. ANOVA F values for the features in the illustrative example.

by the mean squared variance within the same class. Thus a large F value indicates that the respective feature can discriminate different classes well. In our example the feature "da" only appears in the Bell class so it receives a large F value. Note that the F values for the features whose values are zero in all the instances are set to 0. We can store only the non-all-zero features to save space.

Assume there are k features with non-zero F values, then k times leave-one-out cross validation are performed. In the ith cross validation, all the training instances with their i best features (by F values we have calculated) are used. In our example, in the first round, only "da" is used to conduct the cross validation to receive a validation accuracy. Then the feature sets {"da", "bc"}, {"da", "bc", "ad"}, {"da", "bc", "ad", "cb"} are used sequentially. The feature set that generates the best validation accuracy is chosen as the Bag-Of-Pattern-Features. Larger size set is preferred when there is a tie.

In the cross validation process, the distance between an instance and the centroid vectors are calculated to decide the validation label of the instance. From equation (7), the distance can be computed incrementally. For instance, in our example and in the third round of cross validation, to calculate the distance between two vectors Q and S with feature set {"da", "bc", "ad"}, we can reuse the distance between Q and S with feature set {"da", "bc"} that we got from the previous round. We can just calculate the squared difference of the two "ad" feature values and add it to the previous distance to get the new distance for the third round. Thus in this way, each of the feature value needs to be accessed only once during the

k times cross validation, which greatly reduces the running time. The same incremental calculation can also be applied when using the tf-idf weight according to equation (8).

When calculating the distance between an instance and the centroid that has the same label, as the instance's feature values contributed to the calculation of the centroid values, the direct distance between them may contain bias. To eliminate this bias, the instance's contribution part is deducted from the centroid before calculating their distance.

Figure 4 shows the result of the above cross validation process on our illustrative example. The centroid-based leave-one-out cross validation accuracies on the feature sets {"da"}, {"da", "bc"}, {"da", "bc", "ad"}, {"da", "bc", "ad", "cb"} are 100%, 75%, 75% and 50% respectively. So only "da" is selected as the Bag-Of-Pattern-Feature (Figure 4 Left). Figure 4 (Center) gives the results when using tf-idf and Figure 4 (Right) shows the final feature centroids and tf-idfs for the two classes.

When classifying a new unlabeled test time series, first it is transformed to the BOP representation. Only the features we selected during the training phase are used. The test feature vector is compared to the learned class centroids or tf-idfs to determine its predicted label. Between centroid and tf-idf, we can pick the one that generated better cross validation accuracy in the training phase for testing.

The effectiveness of using a subset of features from the above process instead of using all the features is verified experimentally in Section IV-C. The results on 85 datasets show that just using the subset significantly outperforms employing all the features. Employing the subset can also reduce the classification time as fewer features are used in the classification computation.

B. BOPF Algorithm

1) Efficient SAX Calculation: To transform a time series to the BOP representation, the SAX word of each subsequence needs to be computed. In our algorithm, the cumulative sum technique in [15] is applied to calculate the SAX word of an arbitrary subsequence in $O(1)$ time (given the cumulative sums).

Given a time series T, its cumulative sum series U and cumulative squared sum series V are computed as:

$$u_j = \sum_{i=1}^{j} t_i \tag{9}$$

$$v_j = \sum_{i=1}^{j} t_i^2 \tag{10}$$

where $j = 1, \ldots, m$ and u_0, v_0 are set to 0.

Given a subsequence $x = [t_i, \ldots, t_{i+l-1}]$, the mean μ_x and standard deviation σ_x of x can be computed as:

$$\mu_x = \frac{u_{i+l} - u_{i-1}}{l} \tag{11}$$

$$\sigma_x = \sqrt{\frac{v_{i+l} - v_{i-1}}{l} - \mu_x^2} \tag{12}$$

x is divided into ω segments and each segment is transformed to a symbol. The normalized mean value of a segment $y = [t_i, \ldots, t_j]$ is μ_y:

$$\mu_y = \frac{\frac{u_j - u_{i-1}}{j-i+1} - \mu_x}{\sigma_x} \tag{13}$$

Then μ_y is compared to the fixed break points to receive the respective symbol [22].

2) Parameter Selection: In the BOP method, the subsequence length l, SAX word length ω and SAX alphabet size α are user-set parameters. In BOPF, α is set to 4 as the previous research suggests this value is suitable for most of the datasets [3] [22]. A grid search on the combinations of l and ω is performed to select the best values. The ω candidate set is $wd = \{3, 4, 5, 6, 7\}$ and the l candidate set is $wl = \{0.025, 0.05, 0.075, \ldots, 1\}m$, i.e. an arithmetic sequence from $0.025m$ to m with a common difference of $0.025m$, m is the length of the time series. Duplicate values and values less than 10 are removed from wl. In total at most 200 ω and l combinations are tested.

The pseudo-code of BOPF is given in Algorithm 1. Given a training time series dataset D and a test time series Ts, the algorithm returns the predicted label of Ts as output. Lines 1-13 are for the training phase and Lines 14-19 are for the testing phase.

Algorithm 1 Bag-Of-Pattern-Features (BOPF)

Input:
D: training time series dataset
Ts: time series to be classified
Output:
h: predicted label of Ts

1: $[T, Label] = loadData(D)$
2: $[U, V] = CumulativeSum(T)$
3: **for** each $\omega \in wd$ **do**
4: **for** each $l \in wl$ **do**
5: $T_{BOP} = BOP(T, \omega, l, U, V)$
6: $T_F = ANOVA(T_{BOP}, Label)$
7: $[F_1, T_{cen}, AC_1] = CV_{cen}(T_{BOP}, Label, T_F)$
8: $[F_2, T_{tfidf}, AC_2] = CV_{tfidf}(T_{BOP}, Label, T_F)$
9: $Co_{cen}.add(\omega, l, F_1, T_{cen}, AC_1)$
10: $Co_{tfidf}.add(\omega, l, F_2, T_{tfidf}, AC_2)$
11: **end for**
12: **end for**
13: $Co = Selection(Co_{cen}, Co_{tfidf})$
14: **for** each $co \in Co$ **do**
15: $[\omega, l, F, T_{cen}(T_{tfidf})] = co.extract()$
16: $Ts_{BOP} = BOP(Ts, \omega, l)$
17: $H_{co} = Classify(Ts_{BOP}, F, T_{cen}(T_{tfidf}))$
18: **end for**
19: $h = majorityVote(H)$

In Line 1, the dataset is loaded and the time series and labels are stored in T and $Label$ respectively. Line 2 computes the cumulative sums U and V for calculating SAX words. Line 3-4 perform the grid search on ω and l discussed before. Line

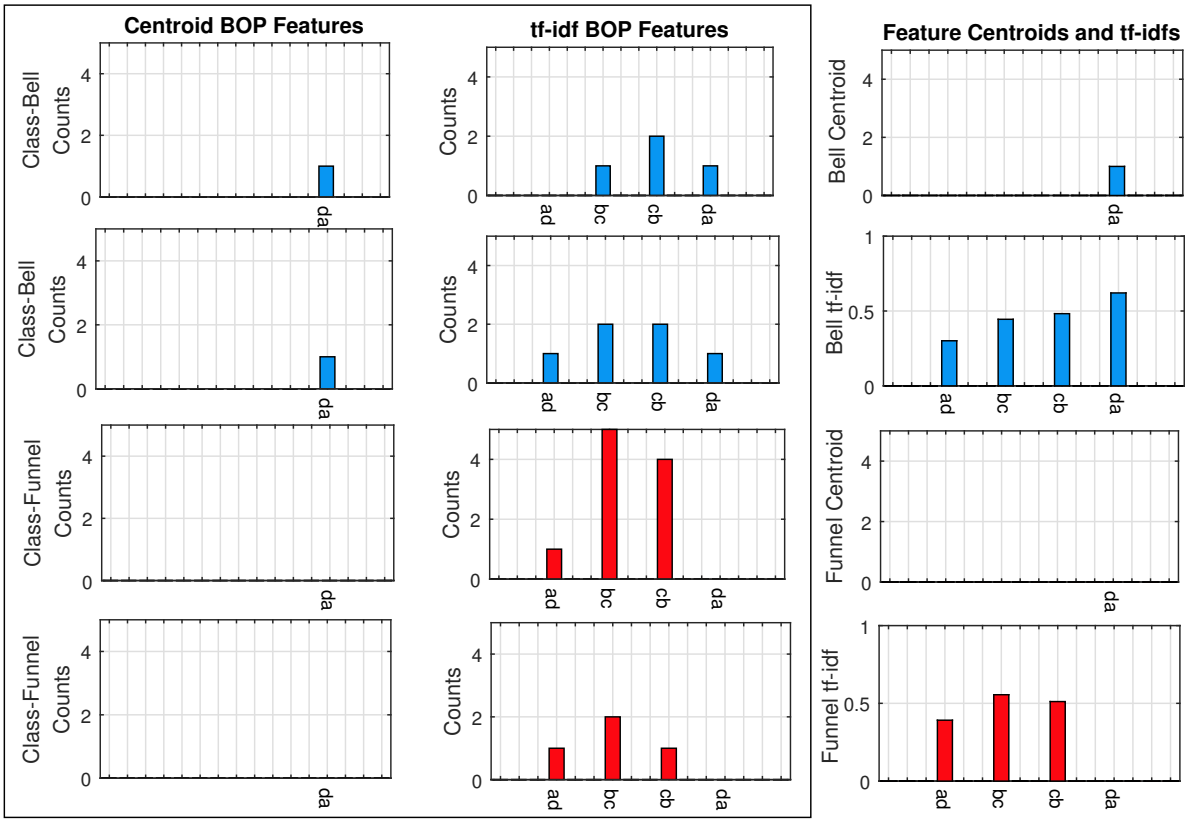

Fig. 4. The Bag-Of-Pattern-Features generated from the illustrative example. (Left) features generated according to centroid (each row for the respective instance). (Center) features generated according to tf-idf. (Right) the feature centroids and tf-idfs.

5 transforms time series T to its BOP representation T_{BOP}. The ANOVA F values for each word are calculated to T_F in Line 6. In Line 7 the cross validation based on centroids is conducted. The selected features F_1, feature centroids T_{cen}, cross validation accuracy AC_1 and the respective ω and l are stored to a combination set Co_{cen} in Line 9. The same process with tf-idf is carried out in Lines 8 and 10.

In Line 13, the best (in validation accuracy) 30 (15% out of 200) parameter combinations (denoted as Co_1) from Co_{cen} are selected. The average accuracy (denoted as $AACo_1$) of the 30 validation accuracies is computed. Similarly Co_2 and $AACo_2$ from Co_{tfidf} are received. If $AACo_1 > 0.7 \max(AACo_1, AACo_2)$, then Co_1 is selected for testing. Similarly if $AACo_2 > 0.7 \max(AACo_1, AACo_2)$, Co_2 is selected and added to Co in Line 13.

In Lines 14-18, each of the combinations co selected in Line 13 is used to transform the test series Ts to its BOP representations Ts_{BOP}. The respective features F, centroid T_{cen}, and tf-idf weight T_{tfidf} are used to predict the label H_{co} of Ts. Finally in Line 19 the majority vote of H is the output h of the algorithm.

The design of taking the top performing parameter combinations and using them to produce a majority vote predication aims at improving the test accuracy. The experiments in Section IV-C verifies the effectiveness of this strategy.

3) Time Complexity: In Algorithm 1, assume the number of instances is n and the length of time series is m in dataset D, the computation of cumulative sum is in linear time complexity ($O(nm)$). The number of grid search combinations is at most 200. Transforming the time series to BOP models takes $O(nm)$ time. Computing the ANOVA F values takes $O(n)$ time as the number of total features is fixed. With the incremental computation process introduced in Section III-A, the cross validation takes $O(n)$ time. So the total training time of BOPF is $O(nm)$.

To verify this theoretical result, BOPF is run on training sets with different numbers of instances and different lengths. The time series are taken from the StarLightCurves dataset in the UCR time series classification archive. The training time of fixing the length to 1000 and changing the instance number from 50 to 1000 is presented in Figure 5.

In Figure 5 the x-axis value of a red dot is the number of training instances and the y-axis value is the average training time of 10 runs on the respective training size. Linear regression curve fitting is performed on the data and the black line in the figure is the fit line. The R^2 value of the fitting, which is the coefficient of determination, is 0.99897. This value is very close to 1, indicating the two variables (training

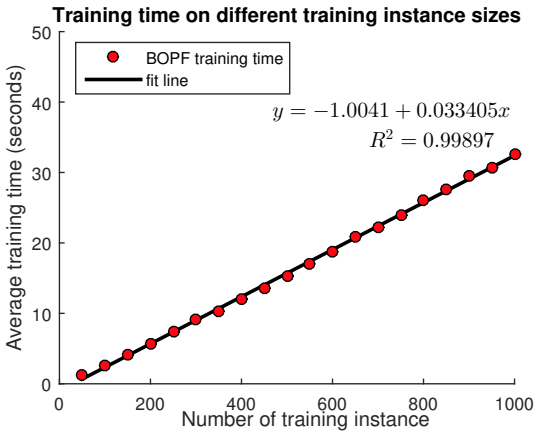

Fig. 5. Training time of BOPF on different numbers of training instances.

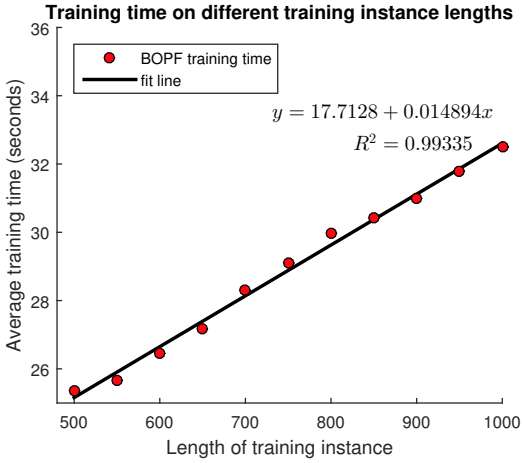

Fig. 6. Training time of BOPF on different training instance lengths.

time and training size) have a strong linear relationship.

The result of fixing the training instance size to 1000 and changing the time series length from 500 to 1000 is given in Figure 6. We start from the length of 500 as when the length is small, many of the subsequence lengths in wl are less than 10 and wl has many duplicate lengths according to the wl definition. These lengths are later removed from the wl, making the training time irregular.

In Figure 6 as in Figure 5, linear regression curve fitting is performed. The R^2 value is 0.99335 which is very close to 1, indicating the training time and training instance length have a strong linear relationship.

From Algorithm 1, at most 60 parameter combinations are used for testing. The testing time complexity is dominated by the BOP transformation process, which takes $O(m)$ time. So the total time complexity of Algorithm 1 is $O(nm)$.

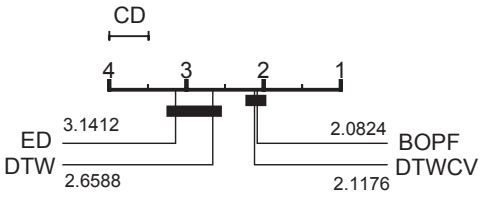

Fig. 7. Critical difference diagram of the comparison with benchmarks.

IV. Experimental Evaluation

A. Experimental Setup

To evaluate the proposed approach, we test it on all 85 datasets in the widely used UCR time series classification archive [5]. The C++ source code of BOPF is available in [26]. The experiments are conducted on a batch-processing cluster [27]. A single core of AMD Opteron Processor 6276 (2299 MHz) and 16 GB memory are used. Note that the algorithms under comparison are all in the same language (C++) and are run under the same hardware condition and operating system. The running time of the algorithms is recorded by using the clock() function in C++.

B. Comparison between BOPF and Benchmarks

The 1-nearest neighbor classifier with Euclidean Distance (ED), 1-nearest neighbor with full size warping window Dynamic Time Warping (DTW) and 1-nearest neighbor DTW with the best warping window constraint selected from Cross Validation (DTWCV) are used as benchmarks. These methods are widely used benchmarks in time series classification and they are introduced in Section II-B. The DTW methods are speeded up by the state-of-the-art UCR-Suite [7].

The four methods are run on 85 datasets and the respective classification error rates and running time are recorded. Table I and II show the classification error rates of the four methods on the UCR archive.

Figure 7 shows the critical difference diagram [28] ($alpha = 0.05$) for the testing error rates comparison. The values in the figure are the respective rank means (lower is better) and the approaches connected by a bold bar have no significant difference to each other. From the figure one can see that the overall classification performance of BOPF in classification accuracy is significantly better than ED and DTW, and slightly better than DTWCV. Performing the Wilcoxon signed rank test on the error rates data, the p-values between BOPF and ED, DTW, DTWCV are 1.5×10^{-6}, 1.8×10^{-3}, 2.3×10^{-1} respectively. The conclusion drawn from these statistics coincides with that from the critical difference diagram.

The time complexity of DTWCV, DTW and BOPF are $O(n^2m^2)$, $O(nm^2)$ and $O(nm)$ respectively. Their actual total running time on the 85 datasets are 1.70×10^6, 9.67×10^4, and 2.73×10^3 seconds respectively. So BOPF is three orders of magnitude faster than DTWCV and one order of magnitude faster than DTW in actual running time.

TABLE I
TESTING ERROR RATES ON UCR ARCHIVE BENCHMARKS

Dataset	ED	DTW	DTWCV	BOPF
50words	0.369	0.31	0.235	0.308
Adiac	0.389	0.396	0.391	0.381
ArrowHead	0.2	0.297	0.2	0.24
Beef	0.333	0.367	0.333	0.3
BeetleFly	0.25	0.3	0.3	0.2
BirdChicken	0.45	0.25	0.3	0.05
Car	0.267	0.267	0.233	0.267
CBF	0.148	0.003	0.006	0
ChlorineConcentration	0.35	0.352	0.35	0.444
CinC_ECG_torso	0.103	0.349	0.07	0.314
Coffee	0	0	0	0.036
Computers	0.424	0.3	0.38	0.308
Cricket_X	0.423	0.246	0.228	0.295
Cricket_Y	0.433	0.256	0.238	0.315
Cricket_Z	0.413	0.246	0.254	0.272
DiatomSizeReduction	0.065	0.033	0.065	0.049
DistalPhalanxOutlineAgeGroup	0.218	0.208	0.228	0.145
DistalPhalanxOutlineCorrect	0.248	0.232	0.232	0.212
DistalPhalanxTW	0.273	0.29	0.273	0.205
Earthquakes	0.326	0.258	0.258	0.183
ECG200	0.12	0.23	0.12	0.18
ECG5000	0.075	0.076	0.075	0.061
ECGFiveDays	0.203	0.232	0.203	0.017
ElectricDevices	0.45	0.399	0.375	0.468
FaceAll	0.286	0.192	0.192	0.204
FaceFour	0.216	0.17	0.114	0
FacesUCR	0.231	0.095	0.088	0.083
FISH	0.217	0.177	0.154	0.069
FordA	0.341	0.438	0.341	0.103
FordB	0.442	0.406	0.414	0.211
Gun_Point	0.087	0.093	0.087	0.02
Ham	0.4	0.533	0.4	0.305
HandOutlines	0.199	0.202	0.197	0.139
Haptics	0.63	0.623	0.588	0.578
Herring	0.484	0.469	0.469	0.406
InlineSkate	0.658	0.616	0.613	0.676
InsectWingbeatSound	0.438	0.645	0.422	0.443
ItalyPowerDemand	0.045	0.05	0.045	0.05
LargeKitchenAppliances	0.507	0.205	0.205	0.243
Lighting2	0.246	0.131	0.131	0.295
Lighting7	0.425	0.274	0.288	0.342
MALLAT	0.086	0.066	0.086	0.057
Meat	0.067	0.067	0.067	0.167
MedicalImages	0.316	0.263	0.253	0.428
MiddlePhalanxOutlineAgeGroup	0.26	0.25	0.253	0.208
MiddlePhalanxOutlineCorrect	0.247	0.352	0.318	0.47
MiddlePhalanxTW	0.439	0.416	0.419	0.398
MoteStrain	0.121	0.165	0.134	0.082
NonInvasiveFatalECG_Thorax1	0.171	0.21	0.189	0.237
NonInvasiveFatalECG_Thorax2	0.12	0.135	0.12	0.176
OliveOil	0.133	0.167	0.133	0.133
OSULeaf	0.479	0.409	0.388	0.202

TABLE II
(CONTINUE) TESTING ERROR RATES ON UCR ARCHIVE BENCHMARKS

Dataset	ED	DTW	DTWCV	BOPF
PhalangesOutlinesCorrect	0.239	0.272	0.239	0.35
Phoneme	0.891	0.772	0.773	0.903
Plane	0.038	0	0	0.01
ProximalPhalanxOutlineAgeGroup	0.215	0.195	0.215	0.156
ProximalPhalanxOutlineCorrect	0.192	0.216	0.21	0.22
ProximalPhalanxTW	0.293	0.263	0.263	0.203
RefrigerationDevices	0.605	0.536	0.56	0.483
ScreenType	0.64	0.603	0.589	0.6
ShapeletSim	0.461	0.35	0.3	0
ShapesAll	0.248	0.232	0.198	0.218
SmallKitchenAppliances	0.659	0.357	0.328	0.195
SonyAIBORobotSurface	0.305	0.275	0.304	0.218
SonyAIBORobotSurfaceII	0.141	0.169	0.141	0.068
StarLightCurves	0.151	0.093	0.095	0.089
Strawberry	0.062	0.06	0.062	0.075
SwedishLeaf	0.211	0.208	0.154	0.163
Symbols	0.1	0.05	0.062	0.052
synthetic_control	0.12	0.007	0.017	0.01
ToeSegmentation1	0.32	0.228	0.25	0.057
ToeSegmentation2	0.192	0.162	0.092	0.085
Trace	0.24	0	0.01	0
TwoLeadECG	0.253	0.096	0.132	0
Two_Patterns	0.09	0	0.002	0.031
UWaveGestureLibraryAll	0.052	0.108	0.034	0.099
uWaveGestureLibrary_X	0.261	0.272	0.227	0.24
uWaveGestureLibrary_Y	0.338	0.366	0.301	0.34
uWaveGestureLibrary_Z	0.35	0.342	0.322	0.31
wafer	0.005	0.02	0.005	0.008
Wine	0.389	0.426	0.389	0.204
WordsSynonyms	0.382	0.351	0.252	0.382
Worms	0.635	0.536	0.586	0.481
WormsTwoClass	0.414	0.337	0.414	0.293
yoga	0.17	0.164	0.156	0.257
Rank mean	3.141	2.659	2.118	2.082

Figure 8 (Figure 9) provides the pairwise running time comparison between BOPF and DTW (DTWCV) on the 85 datasets. Each dot represents a dataset and the x-axis value is the running time (in log-scale and milliseconds) of BOPF on the dataset. The y-axis value in Figure 8 (Figure 9) is the running time of DTW (DTWCV). The speed-up boundary lines are marked in the figures. From the figures and the raw data in [26], one observes that the larger the dataset is, the more speed-up one can receive from using BOPF.

The above experimental results indicate that BOPF can provide competitive classification results using quite a short time. In the situations where a linear time complexity is required, e.g. when dealing with large size datasets and when in real-time analytics, BOPF is considered as a good solution.

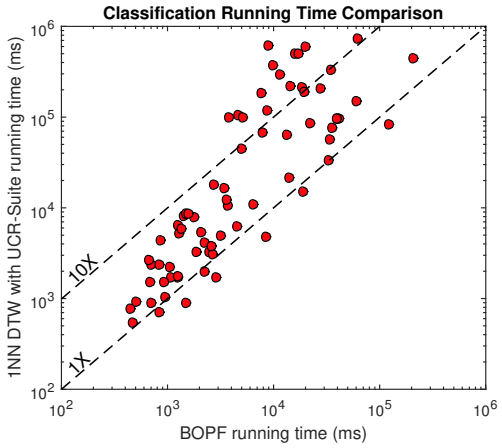

Fig. 8. Pairwise comparison of running time between BOPF and DTW on 85 datasets.

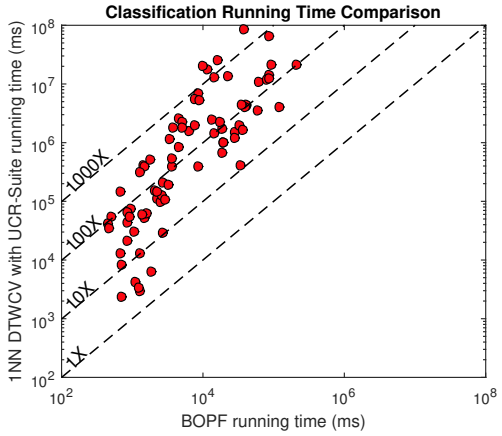

Fig. 9. Pairwise comparison of running time between BOPF and DTWCV on 85 datasets.

C. Effectiveness of Design Strategies

BOPF employs an incremental cross validation procedure to select a subset of features instead of using all of them. Also the top (30) parameter combinations are selected and used to predict the label with a majority vote. To verify the effectiveness of these strategies, 4 versions of BOPF are run on the 85 datasets and their classification performances are compared.

These versions are BOPF, BOPF using all the features (BOPF-all), BOPF using only the best parameter combination (BOPF-best) for prediction, BOPF with all the features and with only the best parameter combination for prediction (BOPF-all-best). Figure 10 presents the critical difference diagram for the comparison. The raw data is available in [26] and is not listed here due to the space limitation.

One can see from the figure that BOPF is significantly better than BOPF-all, which verifies the effectiveness of using a subset of features instead of using all of them. Comparing

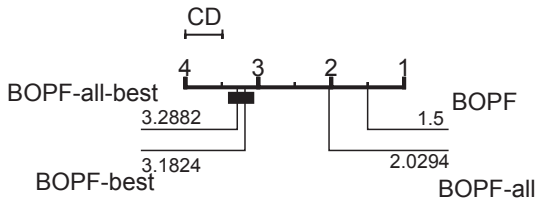

Fig. 10. Critical difference diagram of comparison between BOPF and BOPF-all, BOPF-best, BOPF-all-best.

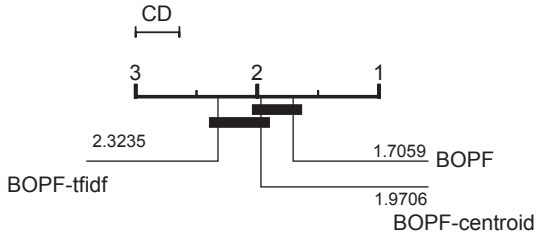

Fig. 11. Critical difference diagram of comparison between BOPF and BOPF-centroid, BOPF-tfidf.

BOPF and BOPF-best, BOPF is significantly better. This indicates that using the top combinations to make a majority vote is superior to using only the best combination. BOPF-all-best is the worst version in the experiment, which coincides with the above discussion.

BOPF adopts a combination of feature centroids and tf-idf weights. To verify this design, 3 versions of BOPF are compared: BOPF, BOPF with only centroids (BOPF-centriod), and BOPF with only tf-idf (BOPF-tfidf). Figure 11 gives the critical difference diagram of the comparison and the raw data is in [26]. The overall performance of BOPF is superior to those of BOPF-centroid and BOPF-tfidf. This result shows that using the combination of centroids and tf-idf weights can generate better performance than just using a single method.

V. Case Study

Xeno-canto [8] is a website of sharing bird sounds around the world. People across the world record the bird sounds in wild and upload the data to the website. We download 15 gigabytes bird sounds from the website and the birds are from two different orders, i.e. Charadriiformes and Piciformes respectively.

The Mel-Frequency Cepstral Coefficients (MFCCs) [9] is adopted to transform each audio file into a 13-dimension time series. We only use the second dimension data in this study and only the first 1000 points of the transformed time series is used. This results in a time series dataset of 24471 instances and 1000 length. This dataset is much larger than any of the 85 datasets in the UCR time series classification archive and is one of the largest datasets for time series classification to date. The labels of the time series are the orders of the birds that produce the sounds. One third of the instances in the

dataset are randomly selected to form a training set and the rest instances form a testing set. The dataset is provided in [26].

BOPF is run on this dataset to test its performance on large size data. Since the dataset is quite large, it is time-consuming to run super-linear time complexity algorithm on this dataset. So we just run the linear time complexity benchmark 1-nearest neighbor with Euclidean Distance (ED) for comparison. The same experimental settings described in Section IV is used.

The ED does not have training phase and it takes 428 seconds for the testing phase on the bird sounds dataset. Its testing error rate is 0.474. On the other hand BOPF spends 525 seconds on the training phase and 171 seconds on the testing phase, receiving a testing error rate of 0.384. So compared with ED, BOPF can achieve a much better classification accuracy result using comparable total running time on this large dataset. If only considering the testing time, which is in the situations where the model has been trained before, BOPF is even much faster than ED. The above results and analysis show the utility of BOPF when dealing with large size data.

VI. Conclusion

The era of big data calls for low time complexity data mining methods to cope with the huge-volume and fast-changing data. This paper propose a linear time complexity time series classification method Bag-Of-Pattern-Features (BOPF). The new method is parameter-free and the experiments on all the 85 datasets in the UCR time series classification archive show that the method can provide competitive results in a quite short time. Further experiments verify the effectiveness of the designs in BOPF. A case study on the bird sounds data shows the utility of the method on real-world large size data.

In the case study we apply MFCC to transform the audio files into 13-dimension time series and only 1 dimension of the data is used in our current study. A future direction is to generalize the method to deal with multivariate time series data and to see if using all the dimensions can improve the classification performance.

Acknowledgment

The experiments were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA. (URL: http://orc.gmu.edu)

References

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, pp. 1–55, 2016.

[2] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[3] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *Journal of Intelligent Information Systems*, vol. 39, no. 2, pp. 287–315, 2012.

[4] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[5] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[6] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[7] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 262–270.

[8] (2017) The xeno-canto website. [Online]. Available: http://www.xeno-canto.org/

[9] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[10] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[11] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

[12] P.-F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 306–318, 2009.

[13] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM (JACM)*, vol. 24, no. 4, pp. 664–675, 1977.

[14] L. Ye and E. Keogh, "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, 2009, pp. 947–956.

[15] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: an expressive primitive for time series classification," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1154–1162.

[16] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[17] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 392–401.

[18] L. Hou, J. T. Kwok, and J. M. Zurada, "Efficient learning of timeseries shapelets," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[19] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[20] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," *arXiv preprint arXiv:1701.07681*, 2017.

[21] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proceedings of the thirteenth SIAM conference on data mining (SDM)*. SIAM, 2013, pp. 668–676.

[22] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[23] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 1175–1180.

[24] P. Schäfer, "Scalable time series classification," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1273–1298, 2016.

[25] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.

[26] (2017) Supporting web page for this paper. [Online]. Available: http://mason.gmu.edu/~xli22/BOPF

[27] (2017) The argo cluster. [Online]. Available: http://wiki.orc.gmu.edu/index.php/About_ARGO

[28] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.