# Short Papers

## A Bag-of-Features Framework to Classify Time Series

Mustafa Gokce Baydogan, George Runger, and Eugene Tuv

**Abstract**—Time series classification is an important task with many challenging applications. A nearest neighbor (NN) classifier with dynamic time warping (DTW) distance is a strong solution in this context. On the other hand, feature-based approaches have been proposed as both classifiers and to provide insight into the series, but these approaches have problems handling translations and dilations in local patterns. Considering these shortcomings, we present a framework to classify time series based on a bag-of-features representation (TSBF). Multiple subsequences selected from random locations and of random lengths are partitioned into shorter intervals to capture the local information. Consequently, features computed from these subsequences measure properties at different locations and dilations when viewed from the original series. This provides a feature-based approach that can handle warping (although differently from DTW). Moreover, a supervised learner (that handles mixed data types, different units, etc.) integrates location information into a compact codebook through class probability estimates. Additionally, relevant global features can easily supplement the codebook. TSBF is compared to NN classifiers and other alternatives (bag-of-words strategies, sparse spatial sample kernels, shapelets). Our experimental results show that TSBF provides better results than competitive methods on benchmark datasets from the UCR time series database.

**Index Terms**—Supervised learning, feature extraction, codebook

## 1 Introduction

Classification of time series is an important task with many challenging applications such as signature verification, speech recognition, or financial analysis. Algorithms can be divided into instance-based and feature-based methods. Instance-based classifiers predict a test instance based on its similarity to the training instances. For time series, one nearest neighbor (NN) classifiers with euclidean (NNEuclidean), or a dynamic time warping distance (NNDTW) have been widely and successfully used [1], [2], [3], [4], [5]. Although euclidean distance is time and space efficient, it is often weak in terms of classification accuracy [3]. DTW [6] allows a measure of the similarity invariant to certain nonlinear variations in the time dimension and is considered as a strong solution for time series problems [7]. DTW attempts to compensate for possible time translations/dilations between patterns. However, previous authors argued that it is more appropriate to measure similarity from higher level structures in long time series, rather than point-to-point, local comparisons [8]. DTW also provides limited insight into the classifier. Instances can be partitioned based on DTW distance [9], but distances are still not easily interpreted. As an alternative, [10], [11], [12] proposed to identify subsequences (called shapelets) that are representative of a class using euclidean distance. Furthermore, Kuksa and Pavlovic [13] similarly generated subsequences from the time series and defined spatial similarity kernels based on the subsequences (similarity-based approach), with classification from a support vector machine (SVM) [14].

Feature-based approaches are generally faster (depending on feature extraction and classification algorithms). In this direction, Geurts [15] used the discontinuity points from a piecewise-linear approximation of the time series to detect patterns and classify the series. A genetic algorithm-based feature extraction was proposed by Eads et al. [16] and SVM was trained on the extracted features. Also, Nanopoulos et al. [17] proposed a multilayer perceptron neural network fed by statistical features (e.g., means and standard deviations) calculated from the series. Moreover, Rodríguez et al. [18] used intervals of time series to extract features on which an SVM was trained. Standard classification algorithms can be built on global features easily, but they may omit important local characteristics. Local features can supplement global information with useful patterns, but the set of local features may vary in cardinality and lack a meaningful ordering that can cause problems for algorithms that require feature vectors with a fixed dimension. Furthermore, methods based on features of intervals (such as [19], [20]) assume that patterns exist in the same time interval over the instances, but a pattern that defines a certain class may exist anywhere in time [15], as well as be dilated in time.

Our work is based on the bag-of-features (BoF) approach in which complex objects are characterized by feature vectors of subobjects. BoF representations are popular, mostly in computer vision as content-based image retrieval [21], [22], [23], natural scene classification [24], and object detection and recognition [25], [26], [27], [28], [29] because of their simplicity and good performance [30]. A BoF is also referred to as a bag-of-words (BoW) [31] in document classification, a bag-of-instances in multiple instance learning [32], [33], and a bag-of-frames in audio and speech recognition [34], [35]. Local image descriptors are sampled (e.g., random, interest point detector [36]) and characterized by their feature vectors (e.g., distribution of the pixel values). A visual dictionary is learned using the vectors of visual descriptors (e.g., clustering to assign discrete labels to descriptors). The resulting distribution of descriptors is quantized through the codebook (e.g., a histogram of the cluster assignments for the sampled descriptors of each instance) as the summary of the image. Similarly, time series segments may contain rich local information about the time series. A BoF representation allows one to integrate local information from segments of the time series in an efficient way. Moreover, assumptions on the cardinality of the local feature set and patterns in the same time interval can be relaxed by this framework.

Studies on BoF representations for time series data are limited, with few studies in audio and speech recognition literature [34], [35], [37], [38], [39]. Time series similarity based on a BoW representation was considered by Lin and Li [8]. Also, time series were discretized by symbolic aggregate approximation (SAX) [40] and time series were represented as words using the symbols generated by SAX. Similarities of the time series were then computed using the representations from document classification approaches. Our study also works on the interval features, but we consider fixed- and variable-length intervals, and we also include shape-based features such as the slope and variance. Moreover, we learn the bag-of-features representation by training a classifier on

- *M.G. Baydogan is with Security and Defense System Initiative, 781 E. Terrace Rd., Tempe, AZ 85287. E-mail: mbaydoga@asu.edu.*
- *G. Runger is with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, 699 S. Mill Avenue, Brickyard Engineering (BYENG) 553, Tempe, AZ 85281. E-mail: runger@asu.edu.*
- *E. Tuv is with Intel, Logic Technology Development, 5000 W Chandler Blvd, CH5295, Chandler, AZ 85226.*

*Manuscript received 6 Apr. 2012; revised 6 Oct. 2012; accepted 19 Mar. 2013; published online 11 Apr. 2013.*
*Recommended for acceptance by F. de la Torre.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-04-0254.*
*Digital Object Identifier no. 10.1109/TPAMI.2013.72.*

the interval features, and this provides a substantially different representation than those from document classification literature. In [38], the speech signals were represented as images through preprocessing (simulation, strobe detection, temporal integration) and patches were segmented from the images. Using vector quantization, segments were represented by sparse codes and they were aggregated through histograms to generate features at the bag level. Also, Fu et al. [35] used a clustering approach to summarize the local information to a bag level.

Histogram-based approaches for image classification problems do not take the spatial location of the local patches into account in codebook generation. Analogously, BoF models in time series ignore the temporal ordering inherent in the signal and therefore may not identify a specific content or pattern [41]. Consequently, here we use a feature-based approach, but extract multiple subsequences from each time series. We consider uniform subsequences which are systematically segmented from the time series and subsequences selected from random locations and random lengths. Features computed from the random subsequences (e.g., mean, standard deviation) measure properties at different locations and dilations when viewed from the original time series. We form a matrix of these features, but the value in row $i$ and row $j$ of the same column may be calculated from subsequences that differ in location and/or length. We further partition subsequences into intervals to detect patterns represented by a series of values over shorter time segments. Moreover, location information is added and easily integrated via a supervised process to construct a codebook and a learner that handles mixed data types, different units, and so on.

Subsequences are labeled and these features are input to a tree-based (recursive partitioning) ensemble that partitions subsequences relevant to the class. In this manner, we provide a feature-based approach that can handle warping (but differently from DTW). The ensembles are trained to construct a compact codebook from simple class probability estimate (CPE) distributions. Additionally, relevant global features can easily supplement the codebook in our framework. Our supervised approach provides a fast, efficient representation, even with very basic features such as slopes, means, and variances from the subsequences. We demonstrate TSBF is efficient and accurate by comparing to alternative time series classifiers on a full set of benchmark datasets.

Random and uniform subsequence generation are compared. Also, we compare to classifiers based on an unsupervised codebook (from $K$-means clustering) and to classifiers that do not use codebooks (applied to the raw values or to extracted features). We also compare to sparse spatial sample kernels (SSSK) [13] and shapelets [12].

The remainder of this paper is organized as follows: We summarize the problem and describe the TSBF framework in Section 2. Section 3 demonstrates the TSBF by testing on a full set of benchmark datasets from UCR time series database [42]. Section 4 provides sensitivity results. Conclusions are drawn in Section 5.

## 2 Time Series Classification with a Bag of Features

A univariate time series, $x^n = (x_1^n, x_2^n, \ldots, x_T^n)$ is an ordered set of $T$ values. We assume time series are measured at equally spaced time points. We consider univariate time series for simplicity, although our method can be extended to multivariate time series (MTS) in a straightforward manner. Each time series is associated with a class label $y^n$, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, \ldots, C-1\}$. Given a set of unlabeled time series, the task of time series classification is to map each time series to one of the predefined classes.

Note that we first standardize each time series to zero mean and unit standard deviation. This adjusts for potentially different baselines or scales that are not considered to be relevant (or persistent) for a learner.

### 2.1 Subsequences and Feature Extraction

Time series classification approaches that are based on global properties of a time series can potentially be improved with local patterns that may define the class. Therefore, we represent each time series with feature vectors derived from subsequences. Here, a subsequence refers to values from the time series that are contiguous in time. To capture patterns along the time series, each subsequence $s$ is represented by the features of smaller segments called intervals.

We consider subsequences of fixed and random lengths. The random length subsequences can potentially detect patterns that appear with different lengths and be split across the time points [43]. Thus, we generate subsequences of random length $l_s$ and segment them using the same number of intervals to preserve the same number of intervals $d$ for each subsequence. This allows splits in tree-based models to be based on the features from different length intervals $w_s = l_s/d$. Therefore, the relationships of patterns with different lengths can be better captured.

We set a lower bound on the subsequence length $l_{(min)}$ as a proportion $z$ $(0 < z \leq 1)$ of the length of the time series. Thus, $l_s \geq l_{min} = z \times T$. We also set a minimum interval length $w_{min}$ so that extracted features are meaningful (that is, we avoid a slope computed from an interval with one point). Therefore, given $z$ and $w_{min}$ the number of intervals used to represent a subsequence is determined as $d = \lfloor \frac{z \times T}{w_{min}} \rfloor$. There are $r = \lfloor \frac{T}{w_{min}} \rfloor$ possible intervals in a time series if the time series is represented using the minimum interval length. For a subsequence with $d$ intervals, $r - d$ intervals are not covered. Therefore, we generate $r - d$ subsequences so that for every interval the expected number of subsequences that cover it is at least one.

Interval features $f_k(t_1, t_2)$, $(0 < t_1 \leq t_2 \leq T)$ for $k = 1, 2, \ldots, K$, are extracted and combined to represent a subsequence. For each interval, the slope of the fitted regression line, mean of the values, and variance of the values are extracted. These features provide information about the shape, level, and distribution of the values. In addition, the mean and variance of all the values in the subsequence, together with the start and end time points, are also included in the feature vector. Start and end points introduce potentially useful location information.

Subsequences of random lengths are motivated from scale-space theory, which is a framework for multiscale signal representation [44]. Alternatively, subsequences are generated in a fixed, uniform manner. In the uniform generation, the subsequence length is fixed as $z \times T$ and each subsequence is represented by $\lfloor \frac{z \times T}{w_{min}} \rfloor$ intervals. Starting from the first interval of the time series, a stride of one interval is used to generate all subsequences. Then the same number of subsequences as in the random generation is obtained.

### 2.2 Codebook and Learning

After the local feature extraction, a new dataset is generated where the features from each subsequence provide an instance, and each time series forms the bag. The class label defined for each instance is the class of the corresponding time series. A classifier that generates a CPE for each instance is used to score the strength of an assignment. Let $p_c^n(s)$ denote the CPE for class $c$ from subsequence $s$ of series $x^n$. The CPEs are discretized into $b$ bins and the distribution of $p_c^n(s)$ over the bag is summarized with a histogram for each $c$ (denoted by a vector $h_c^n$). The vectors are concatenated over $c$ to form the codebook, $h^n$, for time series $x^n$. Because the sum of CPEs for a subsequence is equal to one, the features for one class can be dropped in the codebook. We use equally spaced bins in our approach so that $(C-1) \times b$

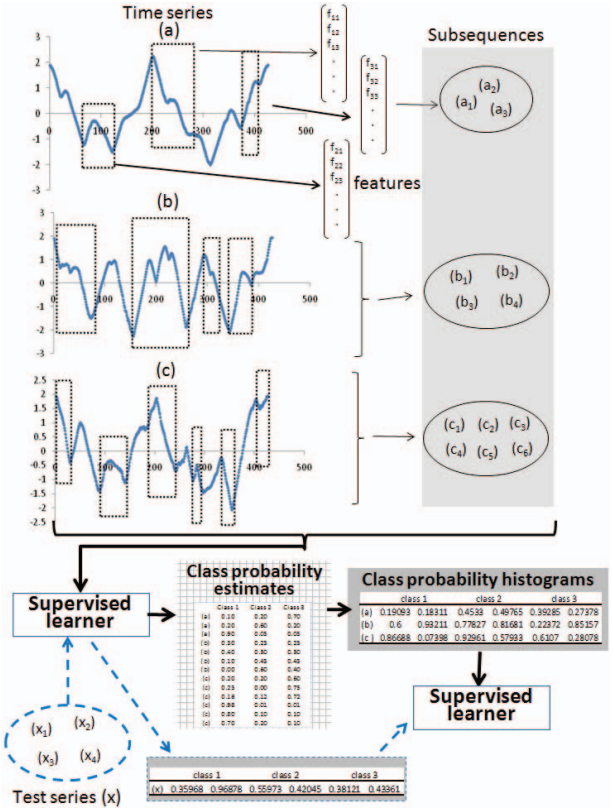


Fig. 1. Time series classification with a bag-of-features (TSBF) algorithm. Subsequences are sampled and partitioned into intervals for feature extraction. Each subsequence is labeled with the class of the time series, and a learner generates class probability estimates. Histograms of the class probability estimates are generated (and concatenated) to summarize the subsequence information. Global features are added. A final classifier is then trained on the new representation to assign each time series.

elements are in the codebook. We aim to capture the details of the similarity between subsequences with the histograms of CPEs. The relative frequencies of the predicted classes over each series are also concatenated in the codebook. Moreover, global features such as autocorrelation are easily introduced (but not used here).

A codebook is an effective way to simplify the information in the subsequences in terms of speed and discrimination [30], [45]. A random forest (RF) classifier [46] is used here to both generate CPEs for codebooks and to classify time series (although another learner that provides CPEs can be used in the framework). An RF is an ensemble of decision trees (each constructed from a different bootstrap sample). The instances not used in the construction of a single tree are called out-of-bag (OOB) instances and they can provide an adequate estimate of generalization error [46]. We denote the RF applied to the subsequence dataset as *RFsub*.

Given the codebook and the global features, an RF (denoted as *RFts*) is applied. RF is competitive with the widely used learners on high-dimensional problems [47]. Fast evaluation is also important and RF is fast for training and testing [46]. Moreover, RF's invariance to attribute scales (units) allows the global features to be easily integrated with the CPE histograms. Furthermore, it is inherently multiclass. Fig. 1 illustrates some of the steps of our approach.

## 3 Experiments and Results

TSBF is tested on a full set of time series data from [42]. The testbed provides diverse characteristics such as the lengths of the series, the number of classes, and so on (as shown in the first columns in Table 2), which enables a comprehensive evaluation. For reproducibility of the results, we built a Web page [48] that contains the code of TSBF and our detailed results.

TABLE 1
Evaluated Parameter Settings for SVM Training on Codebook

| Parameter | Levels |
|---|---|
| Penalty parameter | $\{2^0, 2^1, \ldots, 2^3, 2^4\}$ |
| Kernel | gaussian, polynomial |
| Gamma | $\{2^{-4}, 2^{-3}, \ldots, 2^3, 2^4\}$ |
| Degree (polynomial) | $\{1, 2, 3\}$ |

TSBF does not require the setting of many parameters and it is robust to the settings. RF is insensitive to both the number of trees and the number of candidate attributes scored to potentially split a node [46]. The number of features evaluated at each node of the tree is set to the default here (approximate square root of the number of features). For all RFs, the number of trees are set based on the progress of OOB error rates at discrete number of tree levels with a step size of 50 trees.

Subsequences are determined from two parameters. For each dataset in the experiments, we select the $z$ parameter from the set $\{0.1, 0.25, 0.5, 0.75\}$ to minimize the OOB error rate of *RFts* on the training data. A similar procedure is used to generate uniform subsequences. The same $z$ levels as in the random case are considered and parameters are again selected to minimize OOB error on the training data. This idea is similar to the searching for best warping window in NNDTWBest. Also, we set the minimum interval length $w_{min}$ as five time units to have meaningful features (such as slopes). Again, this setting can be adjusted based on the dataset characteristics (via OOB error) in favor of our algorithm, but we did not modify it here. The number of bins $b$ is set to 10 in our experiments. This parameter is expected to have a small effect on performance if it is set large enough because of the embedded feature selection in RFs.

Our supervised BoF approach is compared to an unsupervised BoW approach with a codebook derived from K-means clustering. In the unsupervised approach, the euclidean distance between subsequences is computed. The subsequences are generated for each level of $z$ as in TSBF with uniform subsequence extraction. Then, K-means clustering with $k$ selected from the set $\{25, 50, 100, 250, 500, 1{,}000\}$ is used to label subsequences for each $z$ setting. We use the histogram of the cluster assignments to generate the codebook.

Two classifiers, RF and SVM, are trained on the codebook for classification. For SVM, the $z$ and $k$ settings and the parameters of the SVM (kernel type, cost parameter) are determined based on 10-fold cross-validation (CV) on the training data. Evaluated parameter settings are provided in Table 1. For multiclass classification, SVMs are trained using the one-against-one approach, in which $\frac{C(C-1)}{2}$ binary classifiers are trained. The class is assigned by a majority voting approach. For RF, the $z$ and $k$ parameters are determined from the OOB error rate on the training data. Our detailed results (the selected settings for RF and SVM) are available in [48].

We also compare to an RF classifier applied directly to the times series values RF (Raw), and to an RF classifier applied to features extracted from fixed-uniform intervals RF (Feature). For the latter RF, an interval length of five units (the same as in our $w_{min}$ setting) is used, and the three features, mean, variance, slope, are extracted from each interval. Codebooks are not constructed for these classifiers.

### 3.1 Classification Accuracy

TSBF, with random and uniform subsequences, is compared to the alternatives mentioned previously and to NNs classifiers with DTW and SSSKs. Two versions of DTW are considered: NNDTWBest [3]

TABLE 2
Error Rates of TSBF (with Random and Uniform Features) over 10 Replicates, NN Classifiers with Dynamic Time Warping Distance (NNDTWBest and NNDTWNoWin), an NN Classifier with Sparse Spatial Sample Kernels (NNSSSK), BoW Approach with RF and SVM, RF (Raw) Applied Directly to Time Series Values, and RF (Feature) Applied Directly to Interval Features

| | # of | # of series | | | TSBF | | NNDTW | | NN | BoW | | RF | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | class | Train | Test | Length | Rand | Unif | Best | NoWin | SSSK | RF | SVM | Raw | Feature |
| 50Words | 50 | 450 | 455 | 270 | 0.209 | 0.211 | 0.242 | 0.310 | 0.488 | 0.347 | 0.316 | 0.348 | 0.333 |
| Adiac | 37 | 390 | 391 | 176 | 0.245 | 0.295 | 0.391 | 0.396 | 0.575 | 0.322 | 0.325 | 0.361 | 0.249 |
| Beef | 5 | 30 | 30 | 470 | 0.287 | 0.460 | 0.467 | 0.500 | 0.633 | 0.267 | 0.267 | 0.300 | 0.257 |
| CBF | 3 | 30 | 900 | 128 | 0.009 | 0.004 | 0.004 | 0.003 | 0.090 | 0.030 | 0.048 | 0.112 | 0.076 |
| Coffee | 2 | 28 | 28 | 286 | 0.004 | 0.007 | 0.179 | 0.179 | 0.071 | 0.000 | 0.036 | 0.007 | 0.004 |
| ECG | 2 | 100 | 100 | 96 | 0.145 | 0.207 | 0.120 | 0.230 | 0.220 | 0.150 | 0.110 | 0.184 | 0.158 |
| Face (all) | 14 | 560 | 1,690 | 131 | 0.234 | 0.196 | 0.192 | 0.192 | 0.369 | 0.278 | 0.238 | 0.190 | 0.231 |
| Face (four) | 4 | 24 | 88 | 350 | 0.051 | 0.048 | 0.114 | 0.170 | 0.102 | 0.125 | 0.102 | 0.211 | 0.172 |
| Fish | 7 | 175 | 175 | 463 | 0.080 | 0.056 | 0.160 | 0.167 | 0.177 | 0.034 | 0.029 | 0.221 | 0.175 |
| Gun-Point | 2 | 50 | 150 | 150 | 0.011 | 0.015 | 0.087 | 0.093 | 0.133 | 0.013 | 0.407 | 0.073 | 0.010 |
| Lighting-2 | 2 | 60 | 61 | 637 | 0.257 | 0.334 | 0.131 | 0.131 | 0.393 | 0.230 | 0.328 | 0.244 | 0.252 |
| Lighting-7 | 7 | 70 | 73 | 319 | 0.262 | 0.370 | 0.288 | 0.274 | 0.438 | 0.301 | 0.370 | 0.263 | 0.295 |
| OliveOil | 4 | 30 | 30 | 570 | 0.090 | 0.167 | 0.167 | 0.133 | 0.300 | 0.267 | 0.233 | 0.107 | 0.093 |
| OSU Leaf | 6 | 200 | 242 | 427 | 0.329 | 0.155 | 0.384 | 0.409 | 0.326 | 0.240 | 0.153 | 0.518 | 0.443 |
| Swedish Leaf | 15 | 500 | 625 | 128 | 0.075 | 0.088 | 0.157 | 0.210 | 0.339 | 0.149 | 0.125 | 0.126 | 0.088 |
| Synt. Control | 6 | 300 | 300 | 60 | 0.008 | 0.009 | 0.017 | 0.007 | 0.067 | 0.017 | 0.017 | 0.046 | 0.017 |
| Trace | 4 | 100 | 100 | 275 | 0.020 | 0.020 | 0.010 | 0.000 | 0.300 | 0.010 | 0.000 | 0.165 | 0.071 |
| Two Patterns | 4 | 1,000 | 4,000 | 128 | 0.001 | 0.004 | 0.002 | 0.000 | 0.087 | 0.034 | 0.010 | 0.158 | 0.190 |
| Wafer | 2 | 1,000 | 6,174 | 152 | 0.004 | 0.003 | 0.005 | 0.020 | 0.029 | 0.011 | 0.010 | 0.012 | 0.002 |
| Yoga | 2 | 300 | 3000 | 426 | 0.149 | 0.156 | 0.155 | 0.164 | 0.172 | 0.159 | 0.145 | 0.191 | 0.188 |
| ChlorineConc. | 3 | 467 | 3,840 | 166 | 0.336 | 0.346 | 0.350 | 0.352 | 0.428 | 0.384 | 0.405 | 0.291 | 0.272 |
| CinC_ECG_torso | 4 | 40 | 1,380 | 1,639 | 0.262 | 0.221 | 0.070 | 0.349 | 0.438 | 0.167 | 0.164 | 0.250 | 0.088 |
| Cricket_X | 12 | 390 | 390 | 300 | 0.278 | 0.256 | 0.236 | 0.223 | 0.585 | 0.346 | 0.305 | 0.427 | 0.362 |
| Cricket_Y | 12 | 390 | 390 | 300 | 0.259 | 0.260 | 0.197 | 0.208 | 0.654 | 0.300 | 0.313 | 0.396 | 0.330 |
| Cricket_Z | 12 | 390 | 390 | 300 | 0.263 | 0.244 | 0.180 | 0.208 | 0.574 | 0.297 | 0.295 | 0.406 | 0.380 |
| DiatomSize | 4 | 16 | 306 | 345 | 0.126 | 0.098 | 0.065 | 0.033 | 0.173 | 0.114 | 0.111 | 0.093 | 0.123 |
| ECGFiveDays | 2 | 23 | 861 | 136 | 0.183 | 0.239 | 0.203 | 0.232 | 0.360 | 0.334 | 0.164 | 0.210 | 0.062 |
| FacesUCR | 14 | 200 | 2,050 | 131 | 0.090 | 0.107 | 0.088 | 0.095 | 0.356 | 0.158 | 0.137 | 0.215 | 0.192 |
| Haptics | 5 | 155 | 308 | 1,092 | 0.488 | 0.478 | 0.588 | 0.623 | 0.591 | 0.562 | 0.630 | 0.551 | 0.548 |
| InlineSkate | 7 | 100 | 550 | 1,882 | 0.603 | 0.604 | 0.613 | 0.616 | 0.729 | 0.638 | 0.629 | 0.665 | 0.716 |
| ItalyPower | 2 | 67 | 1,029 | 24 | 0.096 | 0.107 | 0.045 | 0.050 | 0.101 | 0.058 | 0.044 | 0.033 | 0.040 |
| MALLAT | 8 | 55 | 2,345 | 1,024 | 0.037 | 0.036 | 0.086 | 0.066 | 0.153 | 0.042 | 0.098 | 0.082 | 0.094 |
| MedicalImages | 10 | 381 | 760 | 99 | 0.269 | 0.279 | 0.253 | 0.263 | 0.463 | 0.379 | 0.401 | 0.277 | 0.304 |
| MoteStrain | 2 | 20 | 1,252 | 84 | 0.135 | 0.102 | 0.134 | 0.165 | 0.166 | 0.158 | 0.177 | 0.119 | 0.103 |
| SonyRobot | 2 | 20 | 601 | 70 | 0.175 | 0.225 | 0.305 | 0.275 | 0.376 | 0.398 | 0.409 | 0.321 | 0.280 |
| SonyRobotII | 2 | 27 | 953 | 65 | 0.196 | 0.222 | 0.141 | 0.169 | 0.339 | 0.205 | 0.154 | 0.197 | 0.201 |
| StarLightCurves | 3 | 1,000 | 8,236 | 1,024 | 0.022 | 0.025 | 0.095 | 0.093 | 0.135 | 0.023 | 0.021 | 0.052 | 0.036 |
| Symbols | 6 | 25 | 995 | 398 | 0.034 | 0.025 | 0.062 | 0.050 | 0.184 | 0.077 | 0.088 | 0.148 | 0.138 |
| TwoLeadECG | 2 | 23 | 1,139 | 82 | 0.046 | 0.030 | 0.132 | 0.096 | 0.257 | 0.112 | 0.248 | 0.268 | 0.119 |
| uWaveGesture_X | 8 | 896 | 3,582 | 315 | 0.164 | 0.160 | 0.227 | 0.273 | 0.358 | 0.260 | 0.242 | 0.245 | 0.210 |
| uWaveGesture_Y | 8 | 896 | 3,582 | 315 | 0.249 | 0.239 | 0.301 | 0.366 | 0.493 | 0.354 | 0.352 | 0.314 | 0.290 |
| uWaveGesture_Z | 8 | 896 | 3,582 | 315 | 0.217 | 0.213 | 0.322 | 0.342 | 0.439 | 0.343 | 0.325 | 0.290 | 0.282 |
| WordsSynonyms | 25 | 267 | 638 | 270 | 0.302 | 0.293 | 0.252 | 0.351 | 0.553 | 0.390 | 0.371 | 0.439 | 0.445 |
| Thorax1 | 42 | 1,800 | 1,965 | 750 | 0.138 | 0.158 | 0.185 | 0.209 | 0.362 | 0.488 | 0.489 | 0.123 | 0.112 |
| Thorax2 | 42 | 1,800 | 1,965 | 750 | 0.130 | 0.116 | 0.129 | 0.135 | 0.315 | 0.184 | 0.220 | 0.090 | 0.079 |
| | | | | win/loss | | 24/20 | 28/17 | 32/13 | 44/1 | 36/9 | 33/12 | 36/9 | 31/13 |
| | | | | p-value | | 0.255 | 0.021 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 |

searches for the best warping window, based on the training data, then uses the learned window on the test data, while NNDTWNoWin has no warping window. Note that DTW is a strong solution for time series problems in a variety of domains [7].

Table 2 summarizes the average error rates from 10 replications of our algorithm on the test data. In some cases, different $z$ settings for TSBF generated the same OOB error rates on the training data. In these cases, the worst error rate on the testing data is selected. Features generated for the test data are computed from the same locations generated for training. The results for NN classifiers were obtained from [42]. Details for SSSK are provided in a later section.

Table 2 provides the significance levels for Wilcoxon matched-pairs signed-ranks tests for TSBF (with random subsequences) and the number of wins/losses against the algorithm on the column. We also use the same approach proposed by Ding et al. [49] to compare results. Scatter plots are used to conduct pairwise comparisons of error rates. Each axis represents a method and each dot represents the error rate for a particular dataset. The line $x = y$ is drawn to represent the region where both methods perform about the same. A dot above the line indicates that approach on the $x$-axis has better accuracy than the one on the $y$-axis for the corresponding dataset. Fig. 2 illustrates the performance of TSBF (average over 10 replicates) with random subsequence generation versus NN alternatives. The performance of TSBF is better for most of the datasets.

### 3.2 Computational Complexity

TSBF is implemented in both C and R Software and our experiments use an Ubuntu 12.04 system with 8-GB RAM, dual-core CPU (i7-3620M 2.7 GHz). We use R only for building the RFs and implemented the algorithms for subsequence and codebook generation in C because R is computationally inefficient in execution of the loops. Moreover, although the CPU can handle four threads in parallel, only a single thread is used. A parallel implementation of TSBF is also available from [48].

The overall computational complexity of our algorithm is mainly due to $RFsub$. The time complexity of building a single tree in $RFsub$ is $O(\sqrt{\nu}\eta \log \eta)$, where $\nu = K \times d + L$ is the number

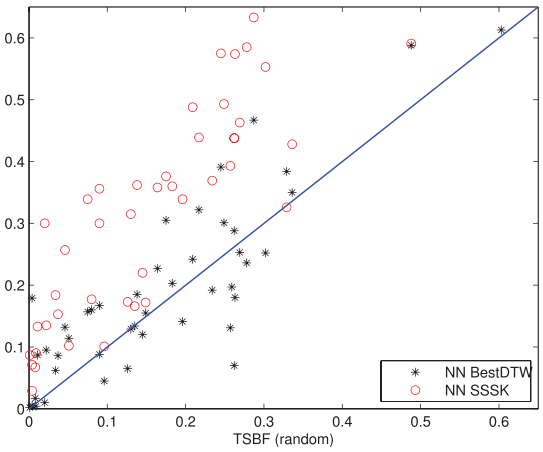


Fig. 2. Error rates for TSBF (random, average over 10 replications) versus NN classifiers.

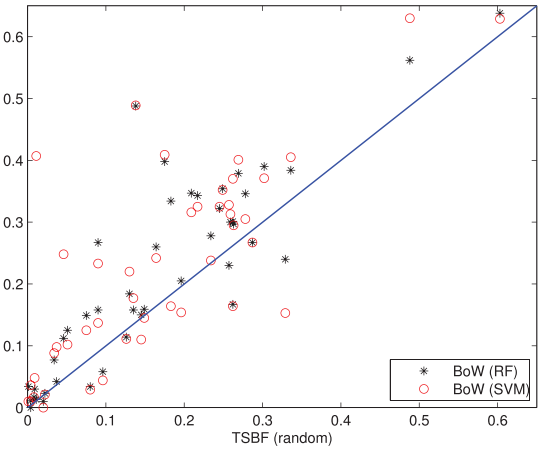


Fig. 4. Average error rates of TSBF (random, average over 10 replications) versus BoW approach with RF and SVM.

of features extracted from each subsequence and $\eta$ is the number of training instances for $RFsub$ (equals $N \times (r-d)$). The smaller $z$ is, the more subsequences are generated, but with fewer features for each.

The training time of TSBF depends on the number of $z$ levels evaluated. We considered four levels of $z$ in this study. We analyzed the average training times over 10 replicates for the datasets in Table 2. The median training time is 16.3 seconds per dataset with a minimum of 1 second and a maximum of 1,949 seconds (for *ItalyPower* and *Thorax1*, respectively). The test time to classify one series (feature generation and classification through RFs) takes less than a millisecond after the models are built. TSBF is very fast and convenient for real-time classification of time series data.

## 4 Sensitivity and Additional Experiments

### 4.1 Benefits of Two-Stage Classification Approach

The performance of our two-stage classification approach is compared to RF (raw) and RF (feature). Introducing interval features improves the error rates when compared to training on time series values. However, TSBF provides significantly better results than both approaches, as illustrated in Fig. 3. This shows the benefit of our two-stage approach. A detailed discussion was also provided by Baydogan [50].

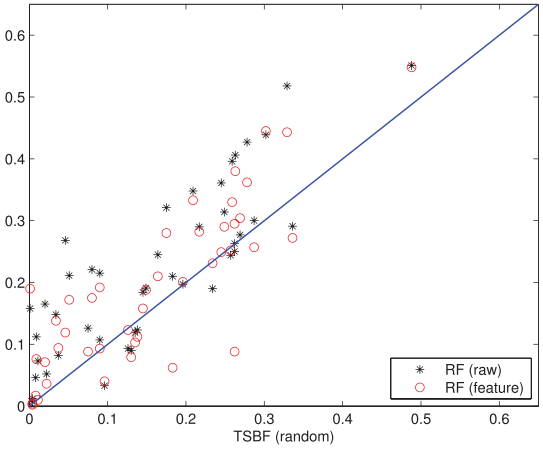


Fig. 3. Average error rates of TSBF (random, average over 10 replications) versus RF (raw) applied directly to time series values and RF (feature) applied directly to features extracted from uniform subsequences. Results show improved performance from the use of a codebook.

Fig. 4 compares TSBF (random, average over 10 replications) to BoW strategies based on $K$-means clustering of subsequences generated in a uniform manner. TSBF performs better than BoW approaches. A BoW representation has the potential to be affected by the curse of dimensionality (specifically the distance calculation) as the number of features extracted from subsequences increases. This is illustrated on a sample problem by Baydogan [50]. On the other hand, our supervised approach uses only the relevant features from the supervised learning and generates the CPEs (codebook) accordingly. Also, the performance difference between SVM and RF on the BoW representation is not significant.

### 4.2 Effect of the Number of Subsequences

We choose six datasets to illustrate the properties of TSBF (random, averaged over 10 replications) versus the number of subsequences in the local feature extraction. These datasets provide reasonable number of training and test time series. Test error rates for TSBF versus the number of subsequences are shown in Fig. 5 for $z = 0.5$. For each dataset, we randomly generated $\delta \in \{0.5, 1, 1.5, 2, 2.5\}$ proportion of the default number of subsequences setting (i.e., $r-d$). Marginal improvements are observed as the number of subsequences increases. Similar results are obtained for other parameter settings (not shown). For all $\delta$ settings, 500 trees are used for $RFsub$ but increasing the number of trees for a greater number of subsequences may provide better results.

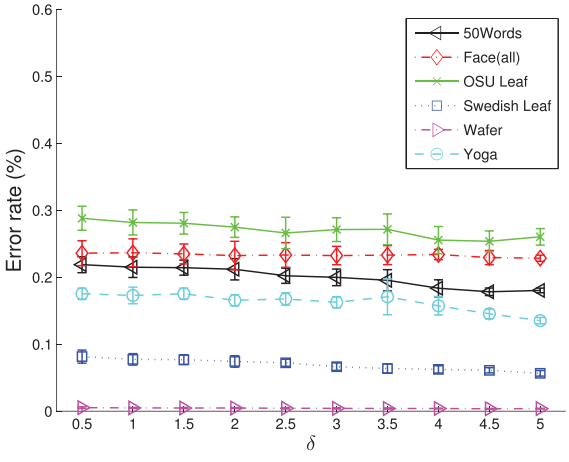


Fig. 5. Error rate of TSBF versus the number of subsequences as a multiplier $\delta$ of the default. Sensitivity to $\delta$ is slight.

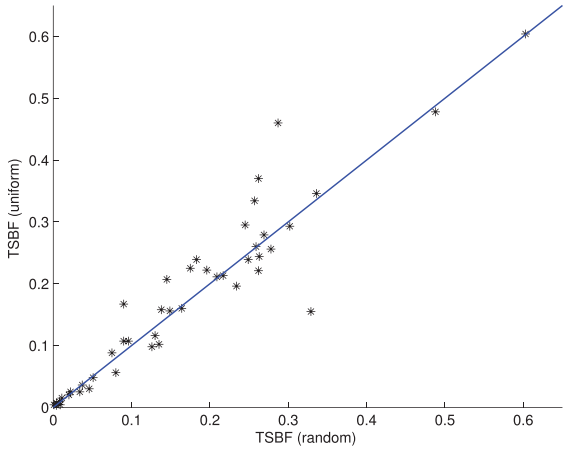


Fig. 6. Plot of average error rates of TSBF with random and uniform subsequence generation.

### 4.3 Random and Uniform Subsequences

The average test error rates over 10 replications of TSBF with random and uniform subsequence generation are illustrated in Fig. 6. Random generation provides slightly better results, but the difference is not significant. Some replicates of the random sequences yield noticeably better performance as provided in our detailed results [48]. Presumably the warping may be handled better with a certain set of randomly generated subsequences. On the other hand, uniform extraction provides more stable results across the replications. Moreover, the number of trees trained for *RFsub* and *RFts* is approximately the same for both approaches.

### 4.4 Sparse Spatial Sample Kernels and TSBF

Sparse spatial sample kernels embed time series into a high-dimensional feature space and use a multiscale representation of strings to improve sequence matching [13]. The series are first discretized to generate a symbolic representation. Then, the similarity between time series is computed over subsequences that do not need to be contiguous. With $k = 1$, similarity is computed over the symbols. That is, contiguous symbols are not concatenated to generate words (as in bag-of-words). We consider double and triple kernels as proposed by Kuksa and Pavlovic [13]. There are a number of hyperparameters that have to be chosen carefully, such as the kernel parameter $d$, alphabet size, discretization scheme (uniform binning, VQ, k-means, etc.), and related parameters (e.g., number of bins $b$).

We discretize the time series using SAX [40]. We consider five levels for the alphabet size and interval lengths of $\{4, 8, 12, 16, 20\}$. The kernel parameter $d$ is selected from the $\{5, 10, \ldots, \min(50, \text{representation length}/2)\}$. The representation length is basically the ratio of time series length to interval length. The evaluated settings are different for *ItalyPower* because the length is only 24 for this particular dataset. To set the parameters, we perform leave-one-out cross-validation on the training data. The parameter combination providing the best CV error rate is used for testing. The results are provided in Table 2. Fig. 2 illustrates that TSBF provides substantially better results in these experiments.

### 4.5 Shapelets and TSBF

Time series subsequences which are likely to represent a class are referred to as shapelets [10], and multiple shapelets were extended with rules [12]. Shapelet methods are distance-based methods, while ours is feature-based, but both use local patterns related to the classes. Instead of rules, we use the summarized version of this information in the form of CPEs and efficient representation through the BoF. We compare TSBF to logical shapelets [12] in Table 3. To be fair, the parameters of the logical shapelet algorithm are set to search for all possible shapelets.

TABLE 3
Error Rates of Logical-Shapelets and TSBF on Eight Datasets

| | TSBF | Logical | NNDTW | |
|---|---|---|---|---|
| | Random | Shapelets | Best | NoWin |
| Beef | 0.287 | 0.600 | 0.467 | 0.500 |
| CBF | 0.001 | 0.336 | 0.004 | 0.003 |
| Coffee | 0.000 | 0.071 | 0.179 | 0.179 |
| ECG | 0.145 | 0.140 | 0.120 | 0.230 |
| Trace | 0.011 | 0.530 | 0.010 | 0.000 |
| Sony AIBO Robot | 0.175 | 0.041 | 0.305 | 0.275 |
| Cricket | 0.018 | 0.010 | 0.051 | 0.010 |
| Passgraphs | 0.274 | 0.298 | 0.260 | 0.282 |

*TSBF has better or comparable performance on the datasets except for Sony AIBO Robot.*

However, because of the computational requirements, we could not achieve this for certain datasets.

Furthermore, we do not tune the parameters of TSBF for the new datasets. We use the same settings as previously. The algorithms are not compared in terms of computation time because it depends to a large extent on parameter settings. An additional two datasets (Cricket and Passgraphs) from [12] are also added for comparison. TSBF has better or comparable performance on all datasets except for Sony AIBO Robot.

### 4.6 Classification of Multivariate Time Series and Time Series of Different Lengths

A multivariate time series is an $M$-variable time series. TSBF can be modified in several ways for MTS classification. In the first alternative, each univariate time series of MTS can be handled separately and TSBF can be conducted $M$ times to generate a codebook for each of them. Then, the codebooks can be concatenated to represent the MTS (late fusion). Another option is to segment the subsequences from the univariate time series, combine them in a single dataset, and train *RFsub* on this set (early fusion). The codebook can be generated for each univariate time series using the CPEs and then the codebooks can be concatenated as in the first approach.

When the time series are of different lengths, the number of subsequences needs to be modified based on the length of the time series. More (less) subsequences should be segmented for long (short) time series and the CPE histograms are generated accordingly.

## 5 Conclusions

A framework is presented to learn a bag-of-features representation for time series classification. Subsequences extracted from random locations and of random lengths provides a method to handle the time warping of patterns in a feature-based approach. Furthermore, the partition into intervals allows one to detect patterns represented by a series of measurements over shorter time segments. The supervised codebook enables the integration of additional information (such as subsequence locations) through a fast, efficient learner that handles mixed data types, different units, and so on. TSBF provides a comprehensive representation that handles both global and local features. The flexible bag-of-features representation allows for the use of any supervised learner for classification. Our experimental results show that TSBF gives better results than competitive methods on the benchmark datasets from the UCR time series database [42]. Although our focus in this study is on the classification of the time series, the bag-of-features approach can be adjusted to other applications such as similarity analysis, clustering, and so forth.

## Acknowledgments

This research was partially supported by US Office of Naval Research grant no. N00014-09-1-0656.

## REFERENCES


[1] 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.

[2] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery,* vol. 7, no. 4, pp. 349-371, 2003.

[3] C. Ratanamahatana and E. Keogh, "Making Time-Series Classification More Accurate Using Learned Constraints," *Proc. SIAM Int'l Conf. Data Mining,* pp. 11-22, 2004,

[4] K. Ueno, X. Xi, E. Keogh, and D. Lee, "Anytime Classification Using the Nearest Neighbor Algorithm with Applications to Stream Mining," *Proc. IEEE Int'l Conf. Data Mining,* pp. 623-632, 2007.

[5] Z. Xing, J. Pei, and P.S. Yu, "Early Prediction on Time Series: A Nearest Neighbor Approach," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 1297-1302, 2009.

[6] H. Sakoe, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 26, no. 1, pp. 43-49, Feb. 1978.

[7] C. Ratanamahatana and E. Keogh, "Three Myths about Dynamic Time Warping Data Mining," *Proc. SIAM Int'l Conf. Data Mining,* vol. 21, pp. 506-510, 2005.

[8] J. Lin and Y. Li, "Finding Structural Similarity in Time Series Data Using Bag-of-Patterns Representation," *Proc. Int'l Conf. Scientific and Statistical Database Management,* pp. 461-477, 2009.

[9] Y. Yamada, H. Yokoi, and K. Takabayashi, "Decision-Tree Induction from Time-Series Data Based on Standard-Example Split Test," *Proc. Int'l Conf. Machine Learning,* pp. 840-847, 2003.

[10] L. Ye and E. Keogh, "Time Series Shapelets: A New Primitive for Data Mining," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 947-956, 2009.

[11] L. Ye and E. Keogh, "Time Series Shapelets: A Novel Technique That Allows Accurate, Interpretable and Fast Classification," *Data Mining and Knowledge Discovery,* vol. 22, pp. 149-182, 2011.

[12] A. Mueen, E.J. Keogh, and N. Young, "Logical-Shapelets: An Expressive Primitive for Time Series Classification," *Proc. 17th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 1154-1162, 2011.

[13] P. Kuksa and V. Pavlovic, "Spatial Representation for Efficient Sequence Classification," *Proc. 20th Int'l Conf. Pattern Recognition,* pp. 3320-3323, Aug. 2010.

[14] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support Vector Machines," *IEEE Intelligent Systems and Their Applications,* vol. 13, no. 4, pp. 18-28, 1998.

[15] P. Geurts, "Pattern Extraction for Time Series Classification," *Proc. Fifth European Conf. Principles of Data Mining and Knowledge Discovery,* vol. 2168, pp. 115-127, 2001.

[16] D. Eads, K. Glocer, S. Perkins, and J. Theiler, "Grammar-Guided Feature Extraction for Time Series Classification," *Proc. Conf. Neural Information Processing Systems,* 2005.

[17] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-Based Classification of Time-Series Data," *Int'l J. Computer Research,* vol. 10, pp. 49-61, 2001.

[18] J. Rodríguez, C. Alonso, and J. Maestro, "Support Vector Machines of Interval-Based Features for Time Series Classification," *Knowledge-Based Systems,* vol. 18, nos. 4/5, pp. 171-178, 2005.

[19] J. Rodríguez, C. Alonso, and H. Boström, "Boosting Interval Based Literals," *Intelligent Data Analysis,* vol. 5, no. 3, pp. 245-262, 2001.

[20] J.J. Rodríguez and C.J. Alonso, "Interval and Dynamic Time Warping-Based Decision Trees," *Proc. ACM Symp. Applied Computing,* pp. 548-552, 2004.

[21] R. Rahmani and S.A. Goldman, "MISSL: Multiple-Instance Semi-Supervised Learning," *Proc. Int'l Conf. Machine Learning,* pp. 705-712, 2006.

[22] C. Zhang, X. Chen, M. Chen, S.-C. Chen, and M.-L. Shyu, "A Multiple Instance Learning Approach for Content Based Image Retrieval Using One-Class Support Vector Machine," *Proc. IEEE Int'l Conf. Multimedia and Expo,* pp. 1142-1145, 2005.

[23] Q. Zhang, S.A. Goldman, W. Yu, and J.E. Fritts, "Content-Based Image Retrieval Using Multiple-Instance Learning," *Proc. Int'l Conf. Machine Learning,* pp. 682-689, 2002.

[24] O. Maron and A.L. Ratan, "Multiple-Instance Learning for Natural Scene Classification," *Proc. Int'l Conf. Machine Learning),* pp. 341-349, 1998.

[25] B. Babenko, M.-H. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 33, no. 8, pp. 1619-1632, Aug. 2011.

[26] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu, "Multiple Component Learning for Object Detection," *Proc. European Conf. Computer Vision,* pp. 211-224, 2008.

[27] R. Fergus, P. Perona, and A. Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 264-271, 2003.

[28] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 4, pp. 349-361, Apr. 2001.

[29] V.C. Raykar, B. Krishnapuram, J. Bi, M. Dundar, and R.B. Rao, "Bayesian Multiple Instance Learning: Automatic Feature Selection and Inductive Transfer," *Proc. Int'l Conf. Machine Learning,* pp. 808-815, 2008.

[30] E. Nowak, F. Jurie, and B. Triggs, "Sampling Strategies for Bag-of-Features Image Classification," *Proc. European Conf. Computer Vision,* pp. 490-503, 2006.

[31] D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," *Proc. European Conf. Machine Learning,* pp. 4-15, 1998.

[32] Y. Chen, J. Bi, and J.Z. Wang, "Miles: Multiple-Instance Learning via Embedded Instance Selection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, pp. 1931-1947, Dec. 2006.

[33] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez, "Solving the Multiple Instance Problem with Axis-Parallel Rectangles," *Artificial Intelligence,* vol. 89, nos. 1/2, pp. 31-71, 1997.

[34] F. Briggs, R. Raich, and X.Z. Fern, "Audio Classification of Bird Species: A Statistical Manifold Approach," *Proc. IEEE CS Int'l Conf. Data Mining,* pp. 51-60, 2009.

[35] Z. Fu, G. Lu, K.M. Ting, and D. Zhang, "Music Classification via the Bag-of-Features Approach," *Pattern Recognition Letters,* vol. 32, no. 14, pp. 1768-1777, 2011.

[36] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf.,* pp. 147-151, 1988.

[37] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The Bag-of-Frames Approach to Audio Pattern Recognition: A Sufficient Model for Urban Soundscapes but Not for Polyphonic Music," *The J. Acoustical Soc. of Am.,* vol. 122, no. 2, pp. 881-891, 2007.

[38] R.F. Lyon, M. Rehn, S. Bengio, T.C. Walters, and G. Chechik, "Sound Retrieval and Ranking Using Sparse Auditory Representations," *Neural Computation,* vol. 22, pp. 2390-2416, 2010.

[39] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic Annotation and Retrieval of Music and Sound Effects," *IEEE Trans. Audio, Speech, and Language Processing,* vol. 16, no. 2, pp. 467-476, Feb. 2008.

[40] 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, pp. 107-144, 2007.

[41] M. Casey, C. Rhodes, and M. Slaney, "Analysis of Minimum Distances in High-Dimensional Musical Spaces," *IEEE Trans. Audio, Speech, and Language Processing,* vol. 16, no. 5, pp. 1015-1028, July 2008.

[42] E. Keogh, Q. Zhu, B. Hu, H. Y., X. Xi, L. Wei, and C.A. Ratanamahatana, "The UCR Time Series Classification/Clustering Homepage," http://www.cs.ucr.edu/eamonn/time_series_data/, 2011.

[43] T.-c. Fu, "A Review on Time Series Data Mining," *Eng. Applications of Artificial Intelligence,* vol. 24, pp. 164-181, 2011.

[44] T. Lindeberg, "Scale-Space Theory: A Basic Tool for Analyzing Structures at Different Scales," *J. Applied Statistics,* vol. 21, nos. 1/2, pp. 225-270, 1994.

[45] F. Moosmann, E. Nowak, and F. Jurie, "Randomized Clustering Forests for Image Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 9, pp. 1632-1646, Sept. 2008.

[46] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[47] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An Empirical Evaluation of Supervised Learning in High Dimensions," *Proc. Int'l Conf. Machine Learning,* pp. 96-103, 2008.

[48] M.G. Baydogan, "A Bag-of-Features Framework to Classify Time Series Homepage," www.mustafabaydogan.com/a-bag-of-features-framework-to-classify-time-series-tsbf.html, 2012.

[49] 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," *Proc. VLDB Endowment,* vol. 1, pp. 1542-1552, Aug. 2008.

[50] M.G. Baydogan, "Modeling Time Series Data for Supervised Learning," PhD thesis, Arizona State Univ., Dec. 2012.