ABSTRACT

A variety of real world applications fit into the broad definition of time series classification. Using traditional machine learning approaches such as treating the time series sequences as high dimensional vectors have faced the well known "curse of dimensionality" problem. Recently, the field of time series classification has seen success by using preprocessing steps that discretize the time series using a Symbolic Aggregate Approximation technique (SAX) and using recurring subsequences ("motifs") as features.

In this paper we explore a feature construction algorithm based on genetic programming that uses SAX-generated motifs as the building blocks for the construction of more complex features. The research shows that the constructed complex features improve the classification accuracy in a statistically significant manner for many applications.

Track: Evolutionary Machine Learning

Categories and Subject Descriptors
I.2.6 [Computing Methodologies]: Artificial Intelligence—Learning [Knowledge acquisition]

General Terms
Algorithms

Keywords
machine learning; time-series classification; genetic programming; evolutionary computation

1. INTRODUCTION

Many real-world applications such as motion detection in robotics, climate change detection based on anthropogenic measurements, and financial market predictions based on stock price variations fit into a broad definition of time series classification. In time series classification, individual Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2662-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2576768.2598321.

instances are sequences of numeric values associated with labels. Using traditional machine learning approaches such as treating the time series sequences as high-dimensional vectors have faced the well known "curse of dimensionality" problem [25]. Most data mining techniques typically involve some form of transformation on the existing real-valued time-ordered data. The complex structural characteristics of real-world time series data such as high-dimensionality, feature correlation, and measurement-induced noise render classic data mining algorithms ineffective and inefficient for these representations. In contrast, symbolic transformations such as Symbolic Aggregate approxXimation (SAX) have been shown to be very effective for a large number of time series applications [31]. SAX, in addition to performing mapping from time series to discrete symbols, also performs dimensionality reduction.

Unlike other data types, there are no "explicit" features or signals in discretized sequence data that can help traditional machine learning algorithms learn and predict from the data. In sequence-based classification, the immediate goal becomes the discovery of signals or features in the sequence data that correlate with the desired property, as well as discrimination between sequences that contain such property and those that do not. Sequence data exhibit inter-relationships in the elements that are important in understanding and predicting future sequences. However, finding these relationships is proven to be an NP-hard problem [34]. When we use na¨ıve enumerations for defining these features they often result in poor predictions. Some algorithms that perform well in prediction lack transparency, i.e., the discriminating features generated by these methods are not easily identifiable. Recently, an evolutionary algorithm based feature generation (EFG) technique was introduced to generate discriminating features for DNA sequence classification [16, 17]. It was shown that the EFG-based approach was able to generate features in human-readable manner that biologists are interested in, while achieving good predictive performance [16, 17].

In this paper, the SAX and EFG algorithms are combined to produce an effective time series classification methodology. SAX performs the high level transformation of converting real-valued time series to discretized alphabet-based sequences. EFG uses these transformed sequences to find complex patterns as discriminating features that can be used with any generic classifiers. We first provide a brief introduction to time series representations, SAX, and EFG in section 2. We then describe how EFG and SAX can be combined to-
gather in sections 3 and 4. In section 5 we describe the evaluation experiments that were performed on publicly available real-world datasets. Finally, we conclude with some overall observations and directions for future work.

2. BACKGROUND

Time series data have been analyzed in various data mining tasks such as classification, clustering, indexing, and summarization. In this paper we focus only on the classification task; however, the proposed approach is general and can be applied to other tasks as well.

For the purposes of this research a time series is assumed to be an ordered set of real-valued variables, and a time series classification problem is a set of labeled time series instances to be used as training data to construct a predictive model for classifying unseen time series instances.

Most time series classification approaches first perform some transformation that converts the real-valued ordered data to an approximate representation of the data. Many transformations like Discrete Fourier Transform (DFT) [12], Discrete Wavelet Transform (DWT) [5], Piecewise Linear, and Piecewise Constant models (PAA) [20], (APCA) [14, 19], Singular Value Decomposition (SVD), and symbolic representations [25] have been proposed. Each of these techniques can be considered to be approximating the signal using linear combinations of some basis functions. In similarity-based classification of time series, a distance function or a similarity function is defined to measure the dissimilarity-based classification of time series, a distance functional using linear combinations of some basis functions. In the symbolization step, each PAA coefficient is mapped to symbol based on a set of breakpoints. Ideally, we want the distribution of symbols to be equiprobable. Since normal short time series tend to have a Gaussian distribution, we can define the breakpoints as the boundaries that will produce equal-sized areas under the Gaussian curve [28, 26]. A symbolic transformation table could thus be created by defining breakpoints that would result in regions of equal-probability on the Gaussian distribution. These breakpoints (or the z-values) may be determined by looking them up in statistical tables. Once the breakpoints are obtained, the mapping of a PAA coefficient to symbol is straight-forward. Figure 1 summarizes the process.

3. METHODOLOGY

3.1 Symbolic Aggregate approxXimation

SAX is a transformation technique that allows a time series of arbitrary length \( n \) to be reduced to a string of user-defined length \( w \), \( w \ll n \). The alphabet size is also a user-defined integer \( \alpha \), where \( \alpha \geq 2 \). SAX performs the discretization using two distinct steps: 1) transform the data into the Piecewise Aggregate Approximation (PAA) representation, and 2) map the PAA representation into a discrete string representation. SAX can be performed repeatedly via a sliding window on subsequences from a longer time series, in which case a third, optional step may be employed: numerosity reduction of the data.

3.1.1 Piecewise Aggregate Approximation (PAA)

A time series \( T \) of length \( n \) can be represented in a \( w \)-dimensional space by a vector \( C \). The \( i^{th} \) element of \( C \) is calculated by the following equation:

\[
\bar{c}_i = \frac{w}{n} \sum_{j=\frac{i-1}{w}+1}^{\frac{i}{w}} c_j
\]

First, each time series is normalized to have a mean zero and standard deviation of one. The normalized time series data \( T \) is then divided into \( w \) equal sized “frames”. The mean value of the data falling within a frame is calculated and a vector of these values becomes the reduced representation. The representation can be visualized as an approximation of the original time series with a linear combination of box basis functions. PAA has been considered to be a simple yet effective technique compared to more sophisticated ones like DWT and DFT [21, 20].

3.1.2 Discrete Symbolization

In the symbolization step, each PAA coefficient is mapped to a symbol based on a set of breakpoints. Ideally, we want the distribution of symbols to be equiprobable. Since normalized short time series tend to have a Gaussian distribution, we can define the breakpoints as the boundaries that will produce equal-sized areas under the Gaussian curve [28, 26]. A symbolic transformation table could thus be created by defining breakpoints that would result in regions of equal-probability on the Gaussian distribution. These breakpoints (or the \( z \)-values) may be determined by looking them up in statistical tables. Once the breakpoints are obtained, the mapping of a PAA coefficient to symbol is straight-forward. Figure 1 summarizes the process.

3.1.3 Numerosity Reduction

Most time series data have a large number of values, and one common technique is to consider a sliding window of length \( n \) (user defined parameter) subjected to SAX. Each subsequence of length \( n \) is normalized with mean zero and unit standard deviation and converted to a SAX string.
Thus a set of SAX strings are obtained which correspond to the original time series. It was found that a SAX subsequence $S_i$ is likely to be very similar to its neighboring subsequences $S_{i+1}$ and $S_{i-1}$, especially when the sequence is in a smooth region, as depicted in Figure 2. Typically only the first of the repeating sequence of identical strings is recorded to avoid artificial over-representation of patterns. Such technique is called numerosity reduction.

As an example, suppose we have the following sequence of SAX strings with the sliding window technique:

$$S = aac aac abc abb abb abb abb bac baa$$

With the numerosity reduction option, we would get the following sequence instead:

$$S_{nr} = aac_1 aac_2 abc_3 abb_4 abb_5 bac_6 baa_7$$

The subscripts denote the offsets of the strings in the original sequence. In some datasets and applications, including these omitted subsequences may be useful, as they might carry important signals. But in most datasets, it was found that excluding these repeating subsequences resulted in a more accurate representation and classification accuracy [26].

### 3.2 Evolutionary Feature Generation (EFG)

EFG uses a Genetic Programming (GP) algorithm to explore a large, complex space of potentially useful features from the given training dataset [23]. Features are represented as standard GP trees, and a population of features is evolved over time using standard GP mechanisms of mutation and crossover. EFG uses a surrogate filter-based fitness function to estimate the usefulness of the GP-generated features, since the wrapper methodology to find effectiveness of the features is costly. A hal
t of fame mechanism incrementally collects the best estimated features for subsequent use with a classifier. Next, we present details of all the evolutionary elements and constructs that are used in the EFG algorithm.

#### 3.2.1 Feature Representation

Various researchers, as highlighted in the related work section, have individually discovered many building blocks that can be very effective in finding the patterns in sequence classification. The novelty of the EFG algorithm is that it not only defines many new building blocks, but it also gives a structure through strongly-typed GP evolution, combining various building blocks in an effective and human-understandable manner. This structured way of searching a vast feature space involves building a complex structure given the constraints defined from simpler ones. Strongly-typed GP plays the role of giving structure and guidance to the vast search space of features. Next, we highlight the building blocks from the simplest short subsequence known as a motif, which becomes the common building block to the complex higher-order signals that can be constructed through the algorithm. We have arranged the explanation at various levels of complexity starting from Level 1 (the simplest) to Level 3 (the most complex).

#### Level 1: Motif

The most common building block is the presence of a short subsequence of strings of a given length, which are constructed as parse trees from the given alphabets. These motifs are used as a building block in all the second level constructs.

#### Level 2: Pattern Matching Functions

Using the motifs as basic building blocks, EFG constructs features corresponding to patterns to be matched using a set of predefined functions, matches, matchesAtPosition, positionalShift and correlational as indicated in Table 1. The matches captures the simple compositional pattern. The matchesAtPosition allows constructing simple positional features from the motifs at a given position. The positionalShift allows constructing positional features that may be displaced in either direction by a small shift given as a parameter. The correlational operator captures the presence of positional features adjacent to each other, within a distance.

#### Level 3: Complex Higher Order Signals

Many statistical learning approaches, such as Bayesian networks and Markov-chain models, rely on higher order elements formed from lower order signals as the discriminative features [24]. The approach of having logical combinatorial operators like and, or, and not acts in a similar way to construct more complex features combining the simple Level 1 motifs, Level 2 elements, or even the Level 3 features to form any level of complex patterns from simpler conjuncts or disjuncts.

Table 1: A table of the non-terminals and terminal nodes employed by EFG.

<table>
<thead>
<tr>
<th>Name</th>
<th>Args</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>2 non-terminal boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>or</td>
<td>2 non-terminal boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>not</td>
<td>2 non-terminal boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>matches</td>
<td>motif</td>
<td>boolean</td>
</tr>
<tr>
<td>matchesAtPosition</td>
<td>motif, position</td>
<td>boolean</td>
</tr>
<tr>
<td>positionalShift</td>
<td>motif, position, Shift</td>
<td>boolean</td>
</tr>
<tr>
<td>correlational</td>
<td>motif, motif, position, close</td>
<td>boolean</td>
</tr>
<tr>
<td>motif-*</td>
<td>ERC-chars</td>
<td>motif</td>
</tr>
<tr>
<td>position</td>
<td>ERC-int</td>
<td>Integer</td>
</tr>
<tr>
<td>shift</td>
<td>ERC-int</td>
<td>Integer</td>
</tr>
<tr>
<td>close</td>
<td>ERC-int</td>
<td>Integer</td>
</tr>
<tr>
<td>region</td>
<td>ERC-int, ERC-int</td>
<td>Integer</td>
</tr>
<tr>
<td>ERC-char</td>
<td>(Symbols)</td>
<td>Character</td>
</tr>
<tr>
<td>ERC-int</td>
<td>{1, . . . , length}</td>
<td>Integer</td>
</tr>
</tbody>
</table>
3.2.2 Genetic Operators

As in most evolutionary algorithms, individuals have to undergo some modifications through genetic breeding operators to generate a new representation from the existing population individual(s). Studies have shown robust evolutionary algorithms incorporate both mutation and crossover as the breeding operators [38]. In this research, we explored forming problem-specific mutations as small, incremental operators. These mutation operators are motif mutation, positional mutation, shift mutation and adjacency mutation. In this work, the standard subtree crossover, one of the most common genetic recombination operators used in GP [23] is employed.

3.2.3 Bloat Control

One common problem with evolving variable-length or tree-structured individuals in EAs is that as the generation progresses, the individuals become complex in structure or length without any changes to fitness, commonly referred to as “bloat”. One of the ways to control bloat is to have length without any changes to fitness, commonly referred to as “bloat”. One of the ways to control bloat is to have a maximum tree depth. By making specific building blocks, such as the correlational feature, rather than leaving it for evolution to form complex trees with two positional features capturing adjacent positional information is one such example. Another method that is used in EFG to combat increase in length and complexity is to employ a lexicographic tournament selection: if multiple individuals have the same fitness, the selection chooses the individual with the smaller tree depth [23].

3.2.4 Population and Generation Mechanism

The EFG algorithm creates individuals in generation 0 consisting of \( N \) randomly generated features using the well-known ramped half-and-half generative method [23]. The population size for GP is generally large, and we have employed a size of 10000. Instead of keeping the population size fixed for every generation, we employed the well known strategy of population implosion to reduce the size of population by 10% in every generation [27].

3.2.5 Fitness Function

A surrogate fitness function, or a “filter” approach, which is considered to be fast and effective way for feature evaluation [22] is employed in EFG. Since most sequence classification data are imbalanced and have very few positives and a large number of negatives, the goal is to improve precision while managing the discriminating power of features. We formulate the fitness function: 

\[
\text{Fitness}(f) = \frac{C_{+\_f} \ast |C_{+\_f} - C_{-\_f}|}{C_{+\_f} + C_{-\_f}}. 
\]

In this equation, \( f \) refers to a feature, \( C_{+\_f} \) and \( C_{-\_f} \) are the number of positive and negative training sequences that contain the feature \( f \), respectively. \( C_{+\_f} \) and \( C_{-\_f} \) are the total number of positive training sequences. \( C_{+\_f} \) and \( C_{-\_f} \) are the normalized count of positive and negative sequences. This fitness function tracks the occurrence of a feature in positive sequences, as negative sequences may not have any common features or signals. The fitness function additionally penalizes non-discriminating features; that is, features that are equally found in positive and negative training sequences.

The goal is to maximize the fitness function, but the standard fitness formulated for GP aims for minimization [23]. So, the standard fitness of a feature is defined by \( f \) as 

\[
\text{Standard}(f) = 1/(\text{Fitness}(f)). 
\]

EFG then converts the standard fitness back into the GP-adjusted fitness \( 1/(1 + Koza(f)) \) to select fit individuals. Note that the GP-adjusted fitness takes values in \([0, 1]\).

3.2.6 Hall of Fame

Since GP is a generational EA, i.e., the parents die after producing the offspring, there can be a “genetic drift” and convergence to a local optimum [8]. This can result in the loss of some of the best individuals, which can be useful discriminating features for classification. Introducing elitism, i.e., the ability to keep some of the best individuals in the population helps to overcome this at the expense of introducing strong selection pressure. To maintain this balance of not losing the best individuals in every generation and not introducing elitism for strong selection pressure, external storage has been found to be the ideal design decision [8]. The EFG algorithm will use this external storage of features known as hall of fame, and at the end of the EA run, these highly fit feature sets chosen from each generation become the feature set that constitutes the solution.

4. SAX-EFG FRAMEWORK

The overall framework for employing EFG along with SAX for time series is shown in Figure 4. SAX performs the preprocessing to convert the time series data to a sequence of SAX strings, and EFG does the feature generation task while any discriminating classifier like Naïve Bayes can be used to learn models from these features.

The mapping of time series to SAX strings requires various user-defined parameters like the sliding-window length \( n \), PAA frame reduction size \( w \), and alphabet size \( \alpha \) for discretizing. In the experiments in this paper an alphabet of size 4 was used for all the experiments. The sliding window size and PAA size were used from standard SAX based runs from previous research and is mentioned along with results in the table.

The EFG algorithm used a default motif length range of 1 to 8 characters. Ideally, a motif should be restricted to contain only whole SAX strings. That is, in theory a motif should not be allowed to break up a string (pattern). In our experiments, however, we find that enforcing such restriction via the use of \( N \) (don’t care or gap symbol) between SAX strings results in very little difference in accuracy. This may be due to the sparsity of the patterns, as well as the fact that a pattern needs to be both overrepresented and discriminative in order to be included in the feature set. The shift parameter for positional matching with error is set to 3 and closeness for capturing correlation between motifs is set to 4. The mutation and crossover parameters are also set to default of 0.1 each and 0.7 respectively. The hall of fame capture of features per generation is set to 250. The EFG algorithm is run for 30 generations.

4.1 Datasets

The datasets chosen for this task are some standard time series datasets with binary class labels from the real-world time series data. GunPoint dataset comes from the video surveillance domain and has two classes, containing 50 training and 150 testing examples with time series length of 150 [33]. The electrocardiogram (ECG) dataset contains measurements of cardiac electrical activity as recorded from electrodes at various locations on the body; each instance in
Figure 3: The EFG algorithm with the SAX algorithm

Table 2: Dataset characteristics giving time series length, training and test size.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time Series Length</th>
<th>Train Size</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>GunPoint</td>
<td>150</td>
<td>50</td>
<td>150</td>
</tr>
<tr>
<td>ECG200</td>
<td>96</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Coffee</td>
<td>286</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Lightning2</td>
<td>637</td>
<td>60</td>
<td>61</td>
</tr>
<tr>
<td>SonyAIBOSurface</td>
<td>70</td>
<td>20</td>
<td>601</td>
</tr>
</tbody>
</table>

The ECG database contains the measurements recorded by one electrode during one heartbeat [30]. The Coffee dataset comes from food spectrograms domain [32]. The Lightning2 dataset comes from the geographic and satellite-based monitoring domain [11]. The SonyAIBOSurface dataset comes from the robotics and surface detection domain [41]. Table 2 gives the details of each dataset in terms of the time series length, training data size and the testing data size.

5. EXPERIMENTS

5.1 Comparing Time Series Algorithms

For our method, the SAX parameters like the sliding window length $n$ and the PAA reduction length $w$ are set based on previous studies done on SAX and outlined in Table 3. The choice of an alphabet of size 4 was a self-imposed restriction to allow some additional comparisons with our DNA sequence research. In future work, this can be changed and explored with higher numbers.

It has been shown that for time series classification, 1NN on the raw time series using standard distance measures like Euclidean Distance ($L_2$) or Dynamic Time Warping (DTW) [3] outperforms many more sophisticated classifiers [10]. Therefore, to follow literature standard, we compare our approach to 1NN as the classifier, with the following variations: (1) Euclidean Distance as the distance measure on the raw data, (2) DTW as the distance measure on the raw data, (3) SAX as the representation using the same parameters as EFG-SAX (the column labeled SAX), and (4) trained SAX as the representation where the best parameters are learned from the training set for each dataset (the column labeled SAX-BEST). In addition to the INN methods, we also compare with the shapelet and the logical shapelet algorithms.

During the testing phase, the same parameters of SAX are applied to the time series for SAX conversion and the trained model is used for predicting. The error rate measured as $(1 - \text{accuracy})$, the standard metric used in all time series studies, is used as the comparison metric. Accuracy is defined as the percentage of correct predictions.

The implementations using Euclidean Distance ($L_2$), DTW, and SAX were obtained from the authors [26, 3]. The shapelet and logical shapelet were obtained from the work [29, 45]. EFG implementation was also obtained from the authors [16]. The SAX-BEST, SAX and EFG have parameters of sliding window size, segment size and the alphabet size for each experiment in the Table 3 and both the shapelet methods used the best parameters as noted in [29, 45]. All the experiments are run 30 times, comparisons made using paired-t tests and the statistically significant results with 95% confidence are reported in bold-faced and underlined in Table 3.

It can be observed that EFG-SAX gives performance comparable to many time series based algorithms for most datasets. For the Coffee and the GunPoint datasets EFG-SAX gives the statistically significantly better results. More importantly, it can be seen that in 4 out of 5 datasets SAX with EFG performs better than plain SAX. We could not run the Lightning2 dataset with either shapelet or logical shapelet implementations due to memory related issues.

5.2 Comparing Sequence Classification Algorithms

If SAX is used for discretizing the time series classification datasets, can EFG be comparable to other techniques known for handling discrete sequence-based data? A comparison of EFG with feature-based, statistical and kernel methods on some subset of datasets with the same preprocessing from time series to symbolic discrete set using SAX will be the next research goal. Two datasets, GunPoint and SonyAIBOSurface from above, which were statistically best and slightly worse respectively were considered for this comparison. We used an SVM with WeightedDegreePosition Kernel as the string kernel implementation, K-mer for feature based, and Homogeneous HMM and MSP as statistical discriminative and generative techniques.

The kernel parameters such as motif length or order were kept same as that of EFG at 8, while other parameters for the SVM and the HMM were chosen using cross-validation using a default ranges in the grid search. The implementation of the statistical methods employed for comparison is in Java, based on the publicly-available Jstacs package [15]. The kernel-based methods are implemented using the publicly-available Shogun toolkit [35] with the standard SVM implementation provided in the publicly-available LibSVM package [6]. The feature-based methods employed for a baseline validation are implemented in Java. The methods
that have randomness are run 30 times, and the mean error is noted and the significance is calculated using paired-t tests with 95% confidence intervals and shown in bold-faced and underlined in Table 4.

Interestingly, it can be seen that SAX based discretized symbolic representation works well with EFG as compared to other techniques. The dataset SonyAIBOSurface, for which SAX-EFG achieved a bit worse performance as compared to using other state-of-the-art, shows even worse performance using all other techniques like WD-S kernel, K-meer, HMM and MSP.

5.3 Multi-class Time Series Classification

Finally, some of the time series applications are multi-class in nature. EFG in general is a binary-classification-based framework as it tries to find features which are discriminating to one class as compared to the other, evident from the fitness function. There are two broadly different ways to address the multi-class datasets: 1) change the EFG algorithm or the fitness function to accommodate the multi-class discrimination using something like entropy, 2) without changing the EFG algorithm, use machine learning strategies to adapt the binary-class problems to multi-class problems. There have been many machine learning strategies to adapt the binary-class problems to multi-class problems as one-vs.-one or one-vs.-rest problems, creating many binary classification problems.

In this work, we use the second approach and adapt the one-vs.-one strategy in the methodology to create many binary classification problems. We create binary classification models using EFG features for these and combine the models using simple vote mechanism in an ensemble.

The dataset adopted for experimentation is the cylinder-bell-funnel (CBF) time series data [9]. The time series for CBF is defined from the following equations, where \( c(t), b(t) \) and \( f(t) \) define cylinder, bell, and funnel respectively.

\[
  c(t) = (6 + \eta) \cdot \chi[a,b](t) + \varepsilon(t) 
\]

\[
  b(t) = (6 + \eta) \cdot \chi[a,b](t) \cdot (t - a)/(b - a) + \varepsilon(t) 
\]

\[
  f(t) = (6 + \eta) \cdot \chi[a,b](t) \cdot (b - t)/(b - a) + \varepsilon(t) 
\]

\[
  \chi[a,b](t) = \begin{cases} 
  0, t < a \\ 
  1, a < t < b \\ 
  0, t > b 
\end{cases} 
\]

Figure 4 illustrates instances of CBF, showing the cylinder class having a plateau from \( a \) to \( b \), the bell class having a gradual increase from \( a \) to \( b \) and the funnel class having sudden increase at \( a \) and gradual decrease to \( b \). The time series is of length 128 and is considered a model characterizing the properties of temporal domains. Various characteristics of CBF such as random amplitude variation as a result of \( \eta \), random noise as a result of \( \varepsilon \) and large variations at start and end make it really suitable as a model complex classification problem [9].

The implementation of EFG-SAX was trained on three models for cylinder-bell, cylinder-funnel, and bell-funnel using EFG-SAX as before with a Naïve Bayes classifier. The voting ensemble using average function for the probability estimate was chosen to combine the outputs for each models and give classification and confidence to the unseen test data.

We compare EFG-SAX with the following methods: 1NN with Euclidean Distance on the raw data, 1NN with DTW on the raw data, 1NN with SAX using the same parameters as EFG-SAX, 1NN with SAX using trained alphabet size, and two shapelet methods. The results are shown in Table 5. The methods that have randomness are run 30 times, and the mean error is noted and the significance is calculated using paired-t tests with 95% confidence intervals and shown in bold-faced underlined. EFG-SAX outperforms all the traditional time series classification algorithms in a significant way, recording the lowest error rate.

6. CONCLUSION

In this paper, the SAX-EFG framework for time series classification was implemented and studied. It is clear that for binary classification applications, the EFG-SAX framework performs better or close to the best algorithms for time series classification.

It was also interesting to note that EFG-SAX outperforms the traditional feature, kernel, and statistical methods with

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Euclidean</th>
<th>DTW</th>
<th>Shapelet</th>
<th>Logical</th>
<th>SAX-BEST</th>
<th>SAX</th>
<th>EFG-SAX</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>GunPoint</td>
<td>0.09</td>
<td>0.08</td>
<td>0.11</td>
<td>0.14</td>
<td>0.18</td>
<td>0.20</td>
<td><strong>0.032</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td>ECG2000</td>
<td>0.15</td>
<td>0.22</td>
<td>0.15</td>
<td>0.14</td>
<td>0.12(32,8,4)</td>
<td>0.12(32,8,4)</td>
<td>0.13(32,8,4)</td>
<td><strong>2</strong></td>
</tr>
<tr>
<td>Coffee</td>
<td>0.16</td>
<td>0.12</td>
<td>0.04</td>
<td>0.04</td>
<td>0.46(48,4,3)</td>
<td>0.12(32,8,4)</td>
<td>0.18(48,4,4)</td>
<td><strong>1</strong></td>
</tr>
<tr>
<td>Lightning2</td>
<td>0.29</td>
<td><strong>0.17</strong></td>
<td>-</td>
<td>-</td>
<td>0.313(128,8,4)</td>
<td>0.313(128,8,4)</td>
<td>0.22(128,8,4)</td>
<td><strong>2</strong></td>
</tr>
<tr>
<td>SonyAIBOSurface</td>
<td>0.30</td>
<td>0.30</td>
<td>0.15</td>
<td><strong>0.14</strong></td>
<td>0.38(10,8,5)</td>
<td>0.38(10,8,4)</td>
<td>0.32(10,8,4)</td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>

Table 3: Error Rate comparing EFG-SAX with the state-of-the-art algorithms and the last column giving an overall rank in the comparison.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Feature</th>
<th>Kernel</th>
<th>Statistical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datasets</td>
<td>K-Mer</td>
<td>EFG-SAX</td>
<td>WD-S</td>
</tr>
<tr>
<td>Gun</td>
<td>0.07</td>
<td><strong>0.032</strong></td>
<td>0.05</td>
</tr>
<tr>
<td>Sony</td>
<td>0.52</td>
<td><strong>0.32</strong></td>
<td>0.52</td>
</tr>
</tbody>
</table>

Table 4: Error Rate comparing EFG-SAX with feature, statistical, and kernel methods.
same SAX based discretization confirming the generic nature of EFG to find more complex patterns in sequences. Finally, applying EFG for multi-class algorithms using ensemble methodology with one-vs.-one shows relative strength of EFG in finding discriminative features and enhancing the ensemble classification accuracy. In future, we also want to study in-depth the correlation between the discriminative features found in the discretized sequences to the real-valued patterns in the time series.

7. REFERENCES


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>0.14</td>
</tr>
<tr>
<td>DTW</td>
<td>0.05</td>
</tr>
<tr>
<td>SAX-BEST (32,4,10)</td>
<td>0.10</td>
</tr>
<tr>
<td>SAX (32,4,4)</td>
<td>0.11</td>
</tr>
<tr>
<td>EFG-SAX (32,4,4)</td>
<td><strong>0.02</strong></td>
</tr>
<tr>
<td>Shapelet</td>
<td>0.10</td>
</tr>
<tr>
<td>Logical</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 5: Comparison of EFG-SAX with different algorithms on the CBF dataset.


