SAX-EFG: An Evolutionary Feature Generation Framework for Time Series Classification

Uday Kamath
George Mason University
Fairfax, Virginia USA
ukamath@gmu.edu

Jessica Lin
George Mason University
Fairfax, Virginia USA
jessica@gmu.edu

Kenneth De Jong
George Mason University
Fairfax, Virgina USA
kdejong@gmu.edu

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 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 with 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 approXimation (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 interrelationships 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 naïve 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-

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2662-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2576768.2598321.

gether 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 distance or similarity between a pair of sequences. Given such a distance function, one can use existing classification methods such as k-nearest neighbor classifiers (KNNs) or support vector machines (SVMs) with local alignment kernels [42, 7]. Time series shapelets using subsequence patterns and logical combinations of these patterns have recently been proposed and are gaining popularity for classification [29, 45]. Many statistical representation based techniques such as position-specific scoring matrix (PSSM)—also known as the position-weight matrix (PWM)—method, which assumes symbols at all positions are drawn independently [39, 13], the weight array model (WAM) which relaxes assumptions of independence by additionally modeling dependencies on a previous position [40], higher-order Markov models, which model more dependencies and outperform PSSMs [43, 18], and even more complex models like Bayesian networks [4, 1] and Markov Random Fields (MRFs) [44, 2] have been successfully employed in sequence classification. String based kernels in SVM, the weighted position kernel (similar to positional features), and the spectrum kernel (similar to the spectrum features) [36, 37] have been widely used in bioinformatics sequence classification applications.

3. METHODOLOGY

3.1 Symbolic Aggregate approXimation

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 α, 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.

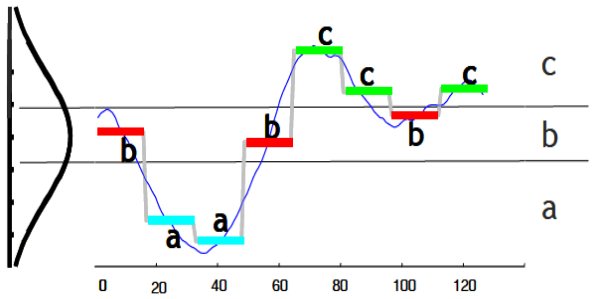

Figure 1: PAA approximations with predetermined breakpoints mapped to symbols. In the example above, with $n = 128$, $w = 8$ and $a = 3$, the time series is mapped to the word baabccbc [26]

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:

$$\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \qquad (1)$$

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.

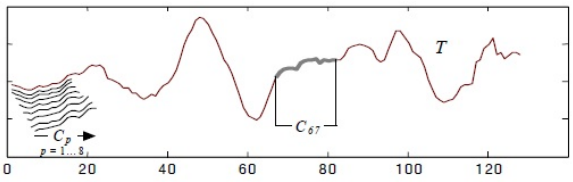

Figure 2: Timeseries of length 128, showing overlapping signals starting at different positions from p 1 to 8 [26].

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:

$S_{nr} = aac_1 \; abc_3 \; abb_4 \; bac_8 \; baa_9$

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 *hall 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).

Name	Args	Return Type
`and`	2 non-terminal boolean	boolean
`or`	2 non-terminal boolean	boolean
`not`	2 non-terminal boolean	boolean
`matches`	motif	boolean
`matchesAtPosition`	motif, position	boolean
`positionalShift`	motif, position,Shift	boolean
`correlational`	motif, motif, position, close	boolean
`motif-*`	ERC-chars	motif
`position`	ERC-int	Integer
`shift`	ERC-int	Integer
`close`	ERC-int	Integer
`region`	ERC-int,ERC-int	Integer
ERC-char	{Symbols}	Character
ERC-int	$\{1, \ldots, length\}$	Integer

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

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.

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 structural elements that reduce the chance of forming larger trees without much improvement to the fitness. 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: $\text{Fitness}(f) = \frac{C_{+,f}}{C_+} * |\hat{C_{+,f}} - \hat{C_{-,f}}|$. 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_+ and C_- are the total number of positive training sequences. $\hat{C_{+,f}}$ and $\hat{C_{-,f}}$ 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 $1/(1+\text{Koza}(f))$ 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 α 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

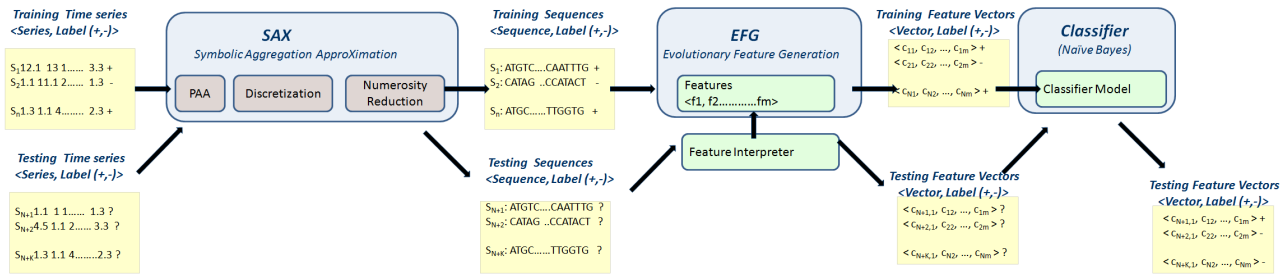

Figure 3: The EFG algorithm with the SAX algorithm

Dataset	Time Series Length	Train Size	Test Size
GunPoint	150	50	150
ECG200	96	100	100
Coffee	286	28	28
Lightning2	637	60	61
SonyAIBOSurface	70	20	601

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

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 1NN 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 $(1 - accuracy)$, 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 sightly 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

Dataset	Euclidean	DTW	Shapelet	Logical	SAX-BEST	SAX	EFG-SAX	Rank
GunPoint	0.09	0.08	0.11	0.14	0.18 (32,4,10)	0.20 (32,4,4)	**0.032** (32,4,4)	1
ECG200	0.15	0.22	0.15	0.14	0.12(32,8,4)	0.12 (32,8,4)	0.13(32,8,4)	2
Coffee	0.16	0.12	0.04	0.04	0.46(48,4,3)	0.18(48,4,4)	**0.01**(48,4,4)	1
Lightning2	0.29	**0.17**	-	-	0.313 (128,8,4)	0.313 (128,8,4)	0.22(128,8,4)	2
SonyAIBOSurface	0.30	0.30	0.15	**0.14**	0.38 (10,8,5)	0.38 (10,8,4)	0.32 (10,8,4)	3

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.

Methodology	Feature		Kernel	Statistical	
Datasets	K-Mer	EFG-SAX	WD-S	HMM	MSP
Gun	0.07	**0.032**	0.05	0.12	0.04
Sony	0.52	**0.32**	0.52	0.47	0.39

Table 4: Error Rate comparing EFG-SAX with feature, statistical, and kernel 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-mer, 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.

$$c(t) = (6 + \eta) * \chi[a, b](t) + \varepsilon(t) \quad (2)$$

$$b(t) = (6 + \eta) * \chi[a, b](t) * (t - a)/(b - a) + \varepsilon(t) \quad (3)$$

$$f(t) = (6 + \eta) * \chi[a, b](t) * (b - t)/(b - a) + \varepsilon(t) \quad (4)$$

$$\chi[a, b](t) = \left\{ \begin{array}{l} 0, t < a \\ 1, a < t < b \\ 0, t > b \end{array} \right\} \quad (5)$$

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 η, random noise as a result of ε 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

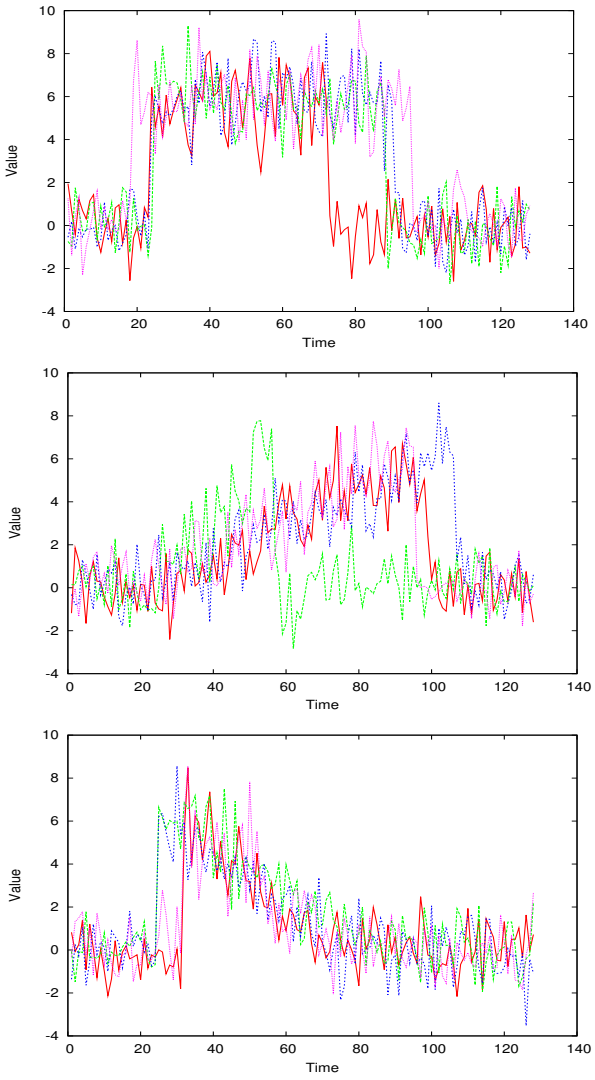

Figure 4: Showing Cylinder, Bell and Funnel as three classes from top to bottom as time series.

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.

Algorithm	Error
Euclidean	0.14
DTW	0.05
SAX-BEST (32,4,10)	0.10
SAX (32,4,4)	0.11
EFG-SAX (32,4,4)	**0.02**
Shapelet	0.10
Logical	0.33

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

7. REFERENCES

[1] I. Ben-Gal, A. Shani, A. Gohr, J. Grau, S. Arviv, A. Shmilovici, S. Posch, and I. Grosse. Identification of transcription factor binding sites with variable-order bayesian networks. *Bioinformatics*, 21(11):2657–2666, 2005.

[2] A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comp Biol*, 3(3):e54, 2003.

[3] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD Workshop*, pages 359–370, 1994.

[4] D. Cai, A. Delcher, B. Kao, and S. Kasif. Modeling splice sites with bayes networks. *Bioinformatics*, 16(2):152–158, 2000.

[5] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE, 1999.

[6] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*. Online, 2001.

[7] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 –27, january 1967.

[8] K. A. De Jong. *Evolutionary computation: a unified approach*. MIT Press, Cambridge, MA, 2001.

[9] J. J. R. Diez, C. A. GonzÃąlez, and H. BostrÃűm. Learning first order logic time series classifiers: Rules and boosting. In *PKDD*, page 299, 2000.

[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. *Proc. VLDB Endow.*, 1(2):1542–1552, Aug. 2008.

[11] D. Eads, D. Hillb, S. Davisa, S. Perkinsa, J. Maa, R. Portera, and J. Theilera. Genetic algorithms and support vector machines for time series classification. In *Proc. of SPIE Vol*, volume 4787, page 75, 2002.

[12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[13] N. I. Gershenzon, G. D. Stormo, and I. P. Ioshikhes. Computational technique for improvement of the position-weight matrices for the DNA/protein binding sites. *Nucl. Acids Res.*, 33(7):2290–2301, 2005.

[14] P. Geurts. Pattern extraction for time series classification. In *Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

[15] J. Grau, J. Keilwagen, A. Gohr, B. Haldemann, S. Posch, and I. Grosse. A java framework for statistical analysis and classification of biological sequences. *J. Mach. Learn. Res.*, 13:1967–1971, 2012.

[16] U. Kamath, J. Compton, R. Islamaj Dogan, K. De Jong, and A. Shehu. An evolutionary algorithm approach for feature generation from sequence data and its application to dna splice site prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(5):1387–1398, 2012.

[17] U. Kamath, K. A. De Jong, and A. Shehu. An evolutionary-based approach for feature generation: Eukaryotic promoter recognition. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 277–284. IEEE, 2011.

[18] J. Keilwagen, J. Grau, I. A. Paponov, S. Posch, M. Strickert, and I. Grosse. De-novo discovery of differentially abundant transcription factor binding sites including their positional preference. *PLoS Comp Biol*, 7(2):e1001070, 2011.

[19] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[20] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *ACM SIGMOD Record*, volume 30, pages 151–162. ACM, 2001.

[21] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[22] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[23] J. Koza. *On the Programming of Computers by Means of Natural Selection*. MIT Press, Boston, MA, 1992.

[24] J. Lasserre and C. M. Bishop. Generative or Discriminative? Getting the Best of Both Worlds. *BAYESIAN STATISTICS*, 8:3–24, 2007.

[25] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[26] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: A novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, Oct. 2007.

[27] S. Luke, G. C. Balan, and L. Panait. Population implosion in genetic programming. in genetic and evolutionary computation. In *Genetic and Evolutionary Computation Conference*, 2003.

[28] M. L. Marx and R. J. Larsen. *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall, 2006.

[29] 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*, KDD '11, pages 1154–1162, 2011.

[30] R. T. Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, DTIC Document, 2001.

[31] P. Patel, E. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *In Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 370–377, 2002.

[32] C. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In *Advances in Knowledge Discovery and Data Mining*, pages 771–777. Springer, 2005.

[33] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of SIAM international conference on data mining*, pages 11–22. Lake Buena Vista, Florida, 2004.

[34] R. Riviere, D. Barth, C. J., and A. Denise. Shuffling biological sequences with motif constraints. *J. Discrete Algo.*, 6(2):192–204, 2007.

[35] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, and F. de Bona. The SHOGUN machine learning toolbox. *J. Mach. Learn. Res.*, 11:1799–1802, 2010.

[36] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(10):S7, 2007.

[37] S. Sonnenburg, A. Zien, P. Philips, and G. Rätsch. Poims: positional oligomer importance matrices - understanding support vector machine-based signal detectors. In *ISMB*, pages 6–14, 2008.

[38] W. M. Spears. Crossover or mutation. In *FOGA*, pages 221–237, 1992.

[39] R. Staden. Methods to locate signals in nucleic acid sequences. *Nucl. Acids Res.*, 12(1):505–519, 1984.

[40] L. Taher, P. Meinicke, and B. Morgensten. On splice site prediction using weight array models: a comparison of smoothing techniques. *J. of Physics: Conference Series*, 90(1):012004, 2007.

[41] D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 235. ACM, 2007.

[42] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[43] E. P. Xing, M. I. Jordan, R. M. Karp, and S. Russell. A hierarchical Bayesian Markovian model for motifs in biopolymer sequences. In S. Becker, S. Thrun, and O. K., editors, *Advances in Neural Information Processing Systems*, pages 200–207, 2002.

[44] O. Yakhnenko, A. Silvescu, and V. Honavar. Discriminatively trained Markov model for sequence classification. In *IEEE Intl Conf on Data Mining (ICDM)*, pages 1–8, November 2005.

[45] 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*, KDD '09, pages 947–956, 2009.