CONCATENATED DECISION PATHS CLASSIFICATION FOR TIME SERIES SHAPELETS

Ivan S. Mitzev, Nickolas H. Younan

Mississippi State University, Mississippi State, MS 39762

ABSTRACT

Time-series classification is widely used approach for classification. Recent development known as time-series shapelets, based on local patterns from the time-series, shows potential as highly predictive and accurate method for data mining. On the other hand, the slow training time remains an acute problem of this method. In recent years there was a significant improvement of training time performance, reducing the training time in several orders of magnitude. Reducing the training time degrade the accuracy in general. This work applies combined classifiers to achieve high accuracies, maintaining low training times- in the range from several second to several minutes- for datasets from the popular UCR database. The goal is achieved by training small 2,3-nodes decision trees and combining their decisions in pattern that uniquely identifies incoming time-series.

KEYWORDS

Data mining, Time-series shapelets, Combining classifiers

1. INTRODUCTION

The time-series shapelets classification method was introduced by Ye and Keogh [1] as a new type of data mining method, that uses the local features of time-series instead of their global. That makes it less sensitive to obstructive noise [1]. This method is successfully applied to a variety of application areas benefiting from its short classification time and high accuracy. Despite its advantages it has a significant disadvantage- a very slow training time. Current research mostly focuses on searching shapelets from all possible combination of time-series derived from a dataset [1, 2], keeping the training process relatively slow. A variety of proposals have been introduced to reduce the candidate shapelets [1, 2, 4, 5], but training time is still in the range of hours for some datasets. A newly introduced method, named *scalable discovery* (SD) method [7] shrinks significantly the training time, making the training process to last from portion of a second to several seconds for investigated 45 datasets from UCR collection [9]. It is based on the idea of pruning similar shapelets in the Euclidian space. Although, this is the fastest up to date training method as of our knowledge, it also maintains high accuracies in compare with other state-of-arts methods. In comparison with other popular methods such as *Logical Shapelets* (LS) and *Fast Shapelets* (FS) methods, the SD method produces better accuracies in 21 out of 45 datasets from UCR database, according to [7]. In this paper we introduce a new method that reaches higher accuracies compared with the method from [7], keeping the training time in observable limits. We tested with 24 datasets from [9], focusing on the datasets with number of classes higher than five. It was found that proposed method outperforms in terms of accuracy the

DOI : 10.5121/ijics.2016.6102

method from [7] for most datasets. The achieved training time is kept low, varying from several seconds to several minutes, depending on a dataset. High accuracy and relatively short training time makes the proposed method very competitive to present state-of-arts methods, which lack either accuracy or have huge training time.

The rest of this paper is organized as follows. In section 2 related work is presented. Section 3 describes the proposed method and gives technical details of its implementation. Section 4 discusses achieved results. Finally, section 5 summarizes the proposed method and gives ideas for further work.

2. RELATED WORK

Shapelet by definition is a sequence of samples that originate from one of the time-series from a dataset and maximally represent certain class. The classical method of shapelets discovery, known as b*rute force algorithm* [1], employs all possible sub-sequences from all time-series from the train dataset and treat them as candidate shapelets. To test a candidate shapelet how well separates two classes A and B, all distances between the candidate shapelet and time-series from A and B are formed. These distances are ordered into a histogram and the histogram is consecutively split into two parts until the best information gain is achieved. The split point is named *optimal split distance* and distances below it considered to belong to class A, but above it to class B. If any other candidate shapelet achieves higher information gain, it is selected as shapelet. The process continue until all the candidate shapelets are processed. The method requires vast amount of calculation time. First improvements include *subsequence distance early abandon* of calculated distances and *admissive entropy pruning* based on predicted information gain [1]. These improvements reduced the total required time for training, but the reduction was not that significant [6]. Another idea based on the *infrequent shapelets*, prunes the non-frequent candidate shapelets [4]. More improvements [2] suggests using of so called *logical shapelets*, that reuse the computation and optimize the search space. Recent approach is based on synthesizing shapelets from random sequences, using *particle swarm optimization* techniques [6]. In an effort to improve the classification accuracy for time series, Bagnall et al. [15] proposes to transform the time series classification problem into another data space for which discriminative features are easily discovered. The transformation include spectral approaches, autocorrelation function, principal components among others. The classifiers considered into [15] include NN type classifiers and ensemble classifiers. In ensemble classifiers the authors train classifiers for the transformed datasets and then combine the classifiers decisions through weighted voting. A new development in the area [7], vastly improves the training time of the shapelets by pruning candidate segments, which shows similarity in Euclidian distance space. This approach [7] is the fastest up to date as of our knowledge and in terms of accuracy is competitive with the current state-of-arts methods. We selected this method as a reference to proposed method, aiming to achieve similar or better accuracies, maintaining relatively low training time.

3. PROPOSED METHOD

Our previous research [6] changes the traditional way of producing shapelets by synthesizing a shapelet from randomly generated sequences using particle swarm optimization (PSO), instead of extracting the shapelets from the original time-series. It finds the shapelet for every pair of classes presented in a dataset, then combines them in a decision tree and find the decision tree that achieves highest accuracy. Producing the most accurate decision tree requires all possible combinations of trees to be tested. That is a slow process for more than four classes and adds additional processing time to traditionally slow training time. For this purpose, only datasets with less than five classes were investigated in [6]. Datasets with more than five classes are processed with the method introduced in this paper. The proposed method utilizes groups of small (up to 4 classes) decision trees, instead of building one big decision tree that contains all classes. When a

time-series comes for classification every classification tree produces a decision. The *decision path* taken during classification is present as string of characters. The decision paths from all present trees are combined into *decision pattern*. Every class from the datasets appears to maintain unique decision pattern. The patterns from training datasets are kept and when time-series from the test dataset arrives for classification, its pattern is compared with the kept patterns. The incoming time-series is associated with the class, to which its decision pattern mostly match.

3.1. Training

3.1.1. Extracting subsets

The first step of the training process is to extract subsets of classes out of the original dataset, for which the decision trees will be defined. It is best to have uniform distribution of class indexes into the subsets, as it allows non dominant class indexes into the final solution. The maximum amount of subsets L is defined as:

$$L = K!/(K - n)!n! \quad (1)$$

where, K is the number of all classes in a dataset and n is the number of classes in a subset $n = 2,3,4$. Instead of taking all possible combinations we can operate with just limited amount of subsets, obeying the rule of uniform distribution of class indexes as shown on Fig.1. Taking limited amount of subsets will not always fully obey the uniform rule. For example, on Fig.1 most of classes are present *3* times, but class *21*, *29* and *30* are present just two times. Practically, it is enough all class indexes to be present into the subsets and the difference between the number of times class indexes are present to be no more than one.Fig. 1 Extracted subsets of 4-class combinations from a dataset with class indexes [1..37].

3.1.2. Training decision trees

[1..37] → [8, 27, 29, 32]
[8, 19, 26, 35]
[4, 6, 35, 37]
[2, 7, 11, 37]
[14, 20, 21, 24]
[21, 24, 29, 33]
[13, 22, 32, 37]
[1, 8, 19, 28]
[4, 10, 21, 25]
[1, 24, 26, 27]
[28, 31, 34, 35]
[4, 12, 30, 36]
[7, 9, 26, 28]
[1, 5, 13, 29]
[2, 5, 10, 25]
[18, 19, 22, 30]
[6, 13, 27, 34]
[3, 14, 25, 34]
[3, 15, 20, 33]
[2, 17, 31, 32]
[9, 14, 20, 22]
[9, 10, 31, 33]
[7, 11, 15, 23]
[3, 5, 12, 30]
[11, 16, 17, 18]
[6, 16, 18, 36]
[12, 15, 17, 23]

Next step of the training process is to create decision trees for extracted subsets. The training process in [6] is applied. It starts with random sequences with length from *3* (the smallest possible shapelet length) up to *N*, where *N* is the length of the time-series into the dataset. Every such sequence is considered candidate shapelet. On every iteration step of the training process the candidate shapelets are changed in a way to increase the information gain that measures the separation between two classes. The decision tree is built by taking into consideration all possible combination of trees produced by such two-class separation.

3.1.3. Decision patterns

The *decision path* is the path taken through the decision tree during decision process. When time-series comes for classification, the distance between shapelet and the time-series is calculated. If such distance is higher than the *optimal split distance* associated with the shapelet, the process takes the right branch of the tree, if not- the process takes the left branch. When right branch is taken, character "R" is added to the decision path, when the left branch is taken, character "L" is added to the decision path. An example of possible decision paths is shown on Fig.2. To form a *decision pattern* the decision paths from all decision trees are concatenated as shown on Fig. 3.

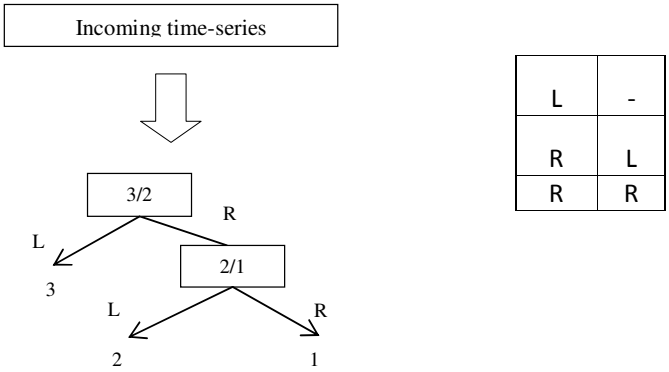

Fig.2 Possible decision paths of incoming time-series.

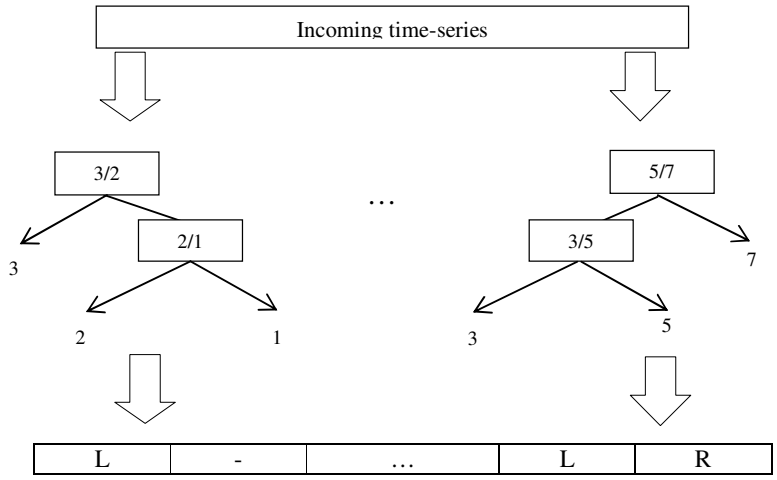

Fig.3 Decision pattern, obtained by combining the decision paths from all subsets' decision trees.

3.2. Classification

During the training process, decision pattern for every time-series from the training dataset is collected. These decision patterns are kept and used during the classification process. When time-series from test dataset comes for classification a decision pattern is created for that time-series. This pattern is compared with the patterns produced during the training process. Comparison process is very simple. The two decision patterns strings are compared character by character in place and the comparison coefficients is equal to the number of characters that coincide by place and value, divided by the length of the decision string as shown on Fig. 4. After all the comparison coefficients are collected we keep only those which are above certain threshold. The idea of this classification is that time-series from the same class will produce similar decision patterns, but decision patterns from different classes should differ significantly. The incoming time-series is identified as class to which it has closest decision pattern. In certain cases more than one class index show similar pattern to the investigates time-series. In such cases the classification process assign the incoming time-series to the class, that has majority of decision patterns closest to the incoming time-series decision pattern.

R	-	L	L	R	-	L	L	L	R
R	L	L	L	L	-	L	R	L	L

Fig. 4 Comparison between decision patterns of two time-series. Six out of ten characters coincide by place and value, therefore the comparison coefficients is 6/10 = 0.6.

4. EXPERIMENTAL RESULTS

The project implementation uses C# and .NET Framework 4.0. Time performance measurements were produced with a *System.Diagnostics.StopWatch* .NET class. In our experiments we used a PC with the following parameters: CPU: Intel Core i7, 2.4GHz; RAM: 8 GB; 64-bit Windows 7 OS. We selected datasets from the UCR collection [9] with a number of classes higher than five (Table 1) as for datasets with fewer classes applying proposed method is meaningless. Table 1 shows parameters of the used datasets. We used method from [7] as a reference method. We downloaded the Java implementation of the reference method from [10] and ran it on the same hardware as proposed method. Reference method requires to specify threshold *p* and aggregation ratio *r.* We kept these value the same as defined in [7] to maintain the highest accuracy.

4.1. Accuracy assessment

Table 2 shows the results of both methods in terms of training time and accuracies they produce. In *18* out of *24* cases the proposed method outperforms the reference method in terms of accuracy, where the improvements vary from *2%* up to *23%*, where in six of these cases the improvement is above *10%*. In the rest, *3* cases differ less than *1.0%* and we consider that both method perform equally for these datasets. Only in *3* cases the reference method outperform the proposed method in terms of accuracy, but the difference is less than *2%*. Although the reference method shows better training times, the proposed method maintains an observable training time-varying from several seconds up to several minutes (~15 min. for Non.FatalECG.1) for datasets that have long time-series and higher number of time-series in a train datasets (uWave.X, uWave.Y, uWave.Z).

4.2. Decision pattern length assessment

The length of the decision pattern may very as shown on Table 2. For datasets, such as "Beef", which consist of *5* classes, the number of subsets is limited to *10* when constructed of *3* class indexes or to *5* when constructed of *4* class indexes. In this case to achieve better accuracy, combination of all possible trees up to 4 indexes are taken. On the other hand, datasets with more class indexes have more varieties to choose from. In the case of "*50Word*" dataset, which contain *50* class indexes, the total amount for combinations for two-classes decision tree is *1225*. We selected *497* of them based on the principle from *3.1.1* and the total length of the decision pattern become *994* characters. Rising the number of characters in the decision pattern in all investigated cases increased the accuracy in general. Although, it appears that there is certain limit of characters above which the accuracy does not increase and even may decrease as shown on Fig.5.

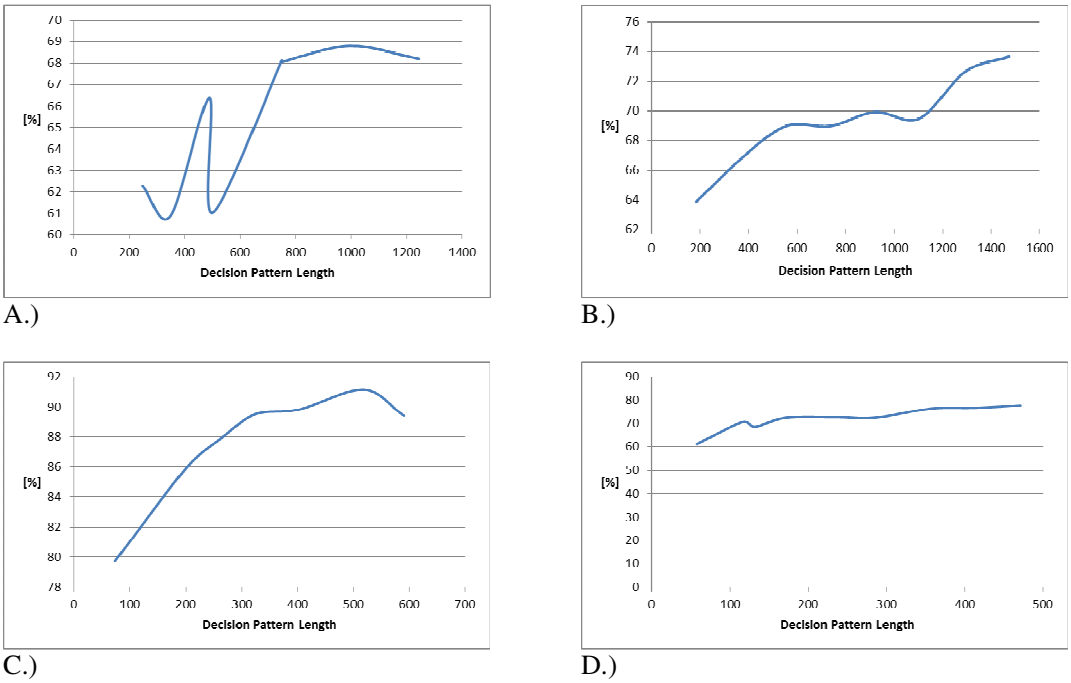

Fig. 5. The influence of the decision pattern length on accuracy. Datasets: A. "50words" (50 class indexes); B. "Adiac" (37 class indexes); C. "Swidish Leaf" (15 class indexes); D. "Cricket X" (12 class indexes).

4.3. Training time assessment

Training the decision trees as mentioned in 3.1.2. is based on the Particle Swarm Optimization (PSO) technique. PSO optimizes certain solution by changing its coordinates into the search space after estimating it with a fitness function. PSO utilizes number of solutions- particles that form a swarm. The change of the particle's coordinates is dictated by the best so far particle's position and the best position in the swarm.

Working with *N-3* candidate shapelets (where *N* is the length of the time-series) as initially proposed in [6] makes the process of training relatively slow. To make it faster the training process initially starts with small amount of time-series extracted from the original datasets. That initial training reveal the typical length of the shapelets for that particular datasets, even though it

does not create a very accurate classification tree. Once the typical length of the shapelets for a datasets are found, only the *10 most typical* lengths are considered further. Thus, instead of using *N-3* candidate shapelets, the training process continues with only 10 candidate shapelets, but with all the time-series from the dataset. The achieved time reduction is significant.

In order to achieve even better training times we introduced data compression as suggested in [7]. According to [7, 16] averaging the neighboring values of the time-series will not harm the accuracy, but will reduce the training time because of the shortened time-series. The compression rate is kept as defined in [7] to make the proposed and reference method comparable.

The iteration process of the PSO as proposed in [6], finishes when certain number of iterations are achieved and when the information gain improvement from the previous step is considered not significant. If the iteration restriction condition is avoided, the training time increases significantly, but the accuracies does not deviate much from when both constrains were used, as shown on Table 3. In *14* out of *24* investigated datasets the accuracy is higher when both constrains are used. In *4* datasets the results are equivalent, but in in *6* cases avoiding iteration constrain produce even higher accuracies. These fluctuations in accuracies are mostly seen because of the randomized initialization of the candidate shapelets. It also depends on the dataset and the user preference for train time vs. accuracy trade off to include or avoid the iteration constrain for the PSO process.

Table 1. Used datasets from UCR database.

Dataset	**Number of Classes**	**Number of time-series in the train/test dataset**	**Time-series length**
Beef	5	30 / 30	470
Haptics	5	155 / 308	1092
OsuLeaf	6	200 / 242	427
Symbols	6	25 / 995	398
synthetic.	6	300 / 300	60
Fish	7	175 / 175	463
InlineSkate	7	100 / 550	1882
Lighting7	7	70 / 73	319
MALLAT	8	55 / 2345	1024
uWave.X	8	896 / 3582	315
uWave.Y	8	896 / 3582	315
uWave.Z	8	896 / 3582	315
MedicalImages	10	381 / 760	99
Cricket X	12	390 / 390	300
Cricket Y	12	390 / 390	300
Cricket Z	12	390 / 390	300
FaceAll	14	560 / 1690	131
FacesUCR	14	200 / 2050	131
SwedishLeaf	15	500 / 625	128
WordsS.	25	267 / 638	270
Adiac	37	390 / 391	176
Non.FatalECG.1	42	1800 / 1965	750
Non.FatalECG.2	42	1800 / 1965	750
50words	50	450 / 455	270

Table 2. Comparison between classification times and accuracies produced by proposed method and the reference method.

Dataset	Comp. Rate	Proposed method			Reference method	
		Pattern Length	Train Time, [sec]	Accuracy, [%]	Train Time, [sec]	Accuracy, [%]
Beef	0.125	70	4.15	**52.21**	0.05	48.89
Haptics	0.500	20	70.66	**39.39**	1.69	34.56
OsuLeaf	0.125	150	55.84	**76.99**	0.14	53.31
Symbols	0.250	150	7.87	**94.20**	0.05	82.48
synthetic.	0.250	150	125.58	98.88	0.06	98.44
Fish	0.250	287	102.51	**90.85**	0.15	75.05
InlineSkate	0.125	245	78.57	39.57	0.56	39.88
Lighting7	0.500	245	42.52	**75.79**	0.39	65.30
MALLAT	0.125	280	42.87	**92.85**	0.10	90.77
uWave.X	0.250	117	559.22	75.32	4.37	**76.45**
uWave.Y	0.250	168	594.66	65.12	3.33	**66.72**
uWave.Z	0.125	117	508.93	66.30	1.89	**67.48**
Med.Images	0.500	240	139.47	**71.27**	0.58	67.68
Cricket X	0.250	471	267.66	**77.78**	0.61	68.63
Cricket Y	0.250	408	198.58	**79.14**	0.50	64.01
Cricket Z	0.250	414	184.38	**75.29**	0.66	68.21
FaceAll	0.500	342	167.02	**75.42**	1.25	71.63
FacesUCR	0.500	330	36.97	**90.56**	0.32	84.61
SwedishLeaf	0.500	519	342.27	**91.14**	0.34	85.60
WordsS.	0.250	600	28.48	**65.46**	0.29	60.92
Adiac	0.500	1473	514.63	**73.65**	0.27	55.67
Non.FatalECG.1	0.250	836	878.18	**85.01**	6.90	80.93
Non.FatalECG.2	0.125	836	349.32	**89.19**	4.67	86.34
50words	0.250	994	58.15	68.79	0.35	68.06

In a further effort to decrease the training time some parallel processing techniques utilizing *.NET Parallel.ForEach* was applied on candidate shapelets, during PSO iterations. The particle coordinates updates are independent from each other, thus parallel processing could be applied successfully on them. Parallel processing was also considered during calculating the distances between candidate shapelet and the time-series from the dataset. As this calculations are independent from each other it was considered as a good place for optimization. Finally, the results did not show any significant improvement of training time after applying these parallel techniques. That is because in both cases the number of parallel calculation was not significant enough. In case of PSO iteration we consider only 10 shapelet candidates, but in case of distance calculation the number of parallel calculation is equal to the number of time-series from the two classes to be distinguished. Applying parallelism on proposed places did not bring significant effects and possibly it will take effect only in case the number of training time-series is considerably high.

Table 3. Comparison between classification times and accuracies produced by proposed method with/without number of iteration constrains during PSO training.

Dataset	Comp. Rate	Proposed method with iteration constrains			Proposed method without iteration constrains	
		Pattern Length	Train Time, [sec]	Accuracy, [%]	Train Time, [sec]	Accuracy, [%]
Beef	0.125	70	4.15	**52.21**	2.65	46.67
Haptics	0.500	20	70.66	**39.39**	13.58	37.98
OsuLeaf	0.125	150	55.84	**76.99**	15.45	71.90
Symbols	0.250	150	7.87	**94.20**	3.31	92.16
synthetic.	0.250	150	125.58	98.88	21.67	98.67
Fish	0.250	287	102.51	90.85	23.39	90.86
InlineSkate	0.125	245	78.57	39.57	18.14	**42.36**
Lighting7	0.500	245	42.52	**75.79**	11.37	72.60
MALLAT	0.125	280	42.87	92.85	12.90	**94.96**
uWave.X	0.250	117	559.22	75.32	135.60	75.26
uWave.Y	0.250	168	594.66	65.12	152.84	65.57
uWave.Z	0.125	117	508.93	66.30	128.49	**68.51**
Med.Images	0.500	240	139.47	**71.27**	62.52	67.37
Cricket X	0.250	471	267.66	**77.78**	58.22	72.82
Cricket Y	0.250	408	198.58	**79.14**	39.25	76.67
Cricket Z	0.250	414	184.38	**75.29**	43.44	71.79
FaceAll	0.500	342	167.02	75.42	43.77	**76.33**
FacesUCR	0.500	330	36.97	**90.56**	12.51	89.95
SwedishLeaf	0.500	519	342.27	**91.14**	64.36	88.96
WordsS.	0.250	600	28.48	**65.46**	14.05	60.66
Adiac	0.500	1473	514.63	73.65	99.10	**75.70**
Non.FatalECG.1	0.250	836	878.18	**85.01**	111.06	84.32
Non.FatalECG.2	0.125	836	349.32	89.19	83.25	**90.48**
50words	0.250	994	58.15	**68.79**	18.29	66.81

5. CONCLUSION AND FUTURE WORK

This paper proposes a new method for time-series shapelets classification, which demonstrates higher accuracies than produced by fastest known state-of-arts method for most of the investigated datasets. It maintains an observable time for training, varying from several seconds to several minutes. The proposed method is easy to implement and can be possibly applied to another classification tasks. As future work we will focus on improving the classification time by utilizing parallel processing capabilities that employ all possible processor's cores on a certain machine. This technology could be successfully applied on the comparison between incoming time-series pattern and available decision patterns as they are processed independently. That will possibly decrease the classification time of the proposed method.

REFERENCES

[1] 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, 2009.

[2] 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, 2011.

[3] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," Proceedings of the 13th SIAM International Conference on Data Mining, 2013.

[4] He1 Q., Dong Z., Zhuang F., Shang T., Shi Z., "Fast Time Series Classification Based on Infrequent Shapelets", 2012 11th International Conference on Machine Learning and Applications, 2012

[5] J. Yuan, Z. Wang, H. Meng, „A discriminative Shapelets Transformation for Time Series Classification", International Journal for Pattern Recognition and Artificial Intelligence, Vol. 28, No. 6, 2014.

[6] I. Mitzev, N. Younan, (2015), "Time Series Shapelets: Training Time Improvement Based on Particle Swarm Optimization", 7th International Conference on Machine Learning and Computing, March 2015

[7] J. Grabocka, M. Wistuba, L. Schmidt-Thieme, "Scalable Discovery of Time-Series Shapelets", arXiv:1503.03238 [cs.LG], March 2015

[8] J. Lines, L. Davis, J. Hills, A. Bagnall, "A Shapelet Transform for Time Series Classification", Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012

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

[10] J. Grabocka, M. Wistuba, L. Schmidt-Thieme, Source Code and Executables for Scalable Discovery of Time-Series Shapelets algorithm, https://www.dropbox.com/sh/btiee2pyn6a989q/AACDfzkkpdYPmgw7pgTgUoeYa

[11] P.Senin, S.Malinchik, "SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space model", Data Mining (ICDM), 2013 IEEE 13th International Conference, 2013

[12] D. Gordon, D. Hendler, L. Rokach, "Fast Randomized Model Generation for Shapelet-Based Time Series Classification", arXiv:1209.5038 [cs.LG], 2012

[13] J. Grabocka, N. Schilling, M.Wistuba, L.Schmidt-Thieme, "Learning Time-Series Shapelets", KDD'14, August 24–27, 2014, NY, USA, 2014

[14] M. Wistuba. J. Grabocka, L. Schmidt-Thieme, "Ultra-Fast Shapelets for Time Series Classification", arXiv:1503.05018v1 [cs.LG] 17 Mar 2015

[15] A. Bagnall, L. Davis, J. Hills and J. Lines, Transformation based ensembles for time series classi¯cation, in Proc. 12th SIAM Int. Conf. Data Mining, California (2012), pp. 307–318.

[16] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," ACM Trans. Database Syst., vol. 27, no. 2, pp. 188–228, Jun. 2002.

Authors

Ivan S. Mitzev is currently PhD candidate of Electrical and Computer Engineering at Mississippi State University. He received his M.S. degree of Electrical Engineering from Mississippi State University in 2010. His research interests include software development, pattern recognition and bio-medical signal processing.

Nicolas H. Younan is currently the Department Head and James Worth Bagley Chair of Electrical and Computer Engineering at Mississippi State University. He received the B.S. and M.S. degrees from Mississippi State University, in 1982 and 1984, respectively, and the Ph.D. degree from Ohio University in 1988. Dr. Younan's research interests include signal processing and pattern recognition. He has been involved in the development of advanced signal processing and pattern recognition algorithms for data mining, data fusion, feature extraction and classification, and automatic target recognition/identification. Dr. Younan has published over 250 papers in refereed journals and conference proceedings, and book chapters. He has served as the General Chair and Editor for the 4th IASTED International Conference on Signal and Image Processing, Co-Editor for the 3rd International Workshop on the Analysis of Multi-Temporal Remote Sensing Images, Guest Editor, Pattern Recognition Letters, and JSTARS, and Co-Chair, Workshop on Pattern Recognition for Remote sensing (2008-2010). He is a senior member of IEEE and a member of the IEEE Geoscience and Remote Sensing society, serving on two technical committees: Image Analysis and Data Fusion, and Earth Science Informatics (previously Data Archive and Distribution). He also served as the Vice Chair of the International Association on Pattern Recognition (IAPR) Technical Committee 7 on Remote Sensing (2008-2010), and Executive Committee Member of the International Conference on High Voltage Engineering and Applications(2010-2014).