Time Series Shapelets: Training Time Improvement Based on Particle Swarm Optimization

Ivan S. Mitzev and Nickolas H. Younan

***Abstract*—Time series classification (TSC) has become a popular research topic in recent years. The TSC works with any real value sequences such as regular discrete time signals as well with time series obtained by conversion from shapes in an image. Recently, time series shapelets classification methods have been used for data mining and time series classification. These methods are proven to have high accuracy and outperform state-of-the-art methods such as the nearest neighbor in cases where the signals are noisy. Furthermore these methods work well with local regions of the signal instead with the whole signal, a technique that in some cases gives better results as the global features are more susceptible to distortions. However, one of the main drawbacks of the time series shapelets classification methods is its slow training time. In this paper, we propose a method that is based on particle swarm optimization (PSO) to improve not only the training time but also performance accuracy when applied to benchmark datasets.**

***Index Terms*—Time series shapelets, particle swarm optimization (PSO), classification.**

I. Introduction

Matching and distinguishing shapes in images are important practical problems. One of the approaches to solve this problem is to convert a shape into time series. This conversion can be done using radial scanning and linear scanning [1], [2], or a different approach. A time series shapelets classification method was introduced by Ye and Keogh [3] as a new data mining method, that works successfully with time series. The method can be used in variety of applications. Some benchmark datasets used in [3] include: recognition of historical artifacts such as shields and pointing arrows, human gait analysis, among others. Beside the wide specter of applications, this method has another advantage- the fast classification time. The classification time is in the range of $O(ml)$, where m is the length of the time series to be classified and l is the length of the shapelet. That makes it comparable with some state-of-the-art methods such as nearest neighbor. In some cases, time series shapelets methods perform even better than other methods in terms of accuracy [4]. That is because it uses local features from the signal, which are less prone to classification errors introduced from noises.

Aside from all these assets, the time series shapelets method has one big disadvantage- its training time is very slow. The brute force training algorithm proposed in [3] investigates $O(m^2k)$ candidate shapelets and the single candidate checks requires $O(mk)$ steps, where m is the average length of training time series and k is the number of time series. Thus, the overall training time is in the range of $O(m^3k^2)$ steps. In this work a new training method is proposed that is based on particle swarm optimizations (PSO). It harnesses N particles, where N is the number of possible shapelets lengths. All the particles represent candidate shapelets. At every iteration of the PSO candidate shapelets are slightly changed and the best local (represent best solution of certain particle) and the best overall candidate is chosen. They form the next change of the particle coordinates. After applying the proposed algorithm, for example in "*Gun/No Gun*" dataset from [5] the average training time dropped from 117.71 to 1.12 seconds, maintaining an accuracy of 81.9%.

The rest of this paper is organized as follows. In Section II some time series shapelets definitions are described along with different approaches for shapelets training. Section III describes the proposed method in details. Section IV compares the results from the brute force method and the proposed method and describes the technical details of the proposed method. Finally, Section V summarizes the pros and cons of the proposed method and gives hint for future investigations.

II. Time Series Shapelets

To have a better understanding of the time series shapelets classification method some definitions are presented below:

Time series: The time series is any consequent set of discrete samples distanced from each other in equal intervals. The time series notation used in this work is $T = \{t_1, t_2, ... t_n\}$.

Distance: To define the distance between two time series with the same length $S = \{s_0, s_1, ..., s_n\}$ and $T = \{t_0, t_1, ..., t_m\}$ the Euclidian distance has been used:

$$d(S,T)= \sqrt{(s_0 - t_0)^2 + \cdots + (s_n - t_n)^2} \quad (1)$$

In the case of $n < m$, the distance is calculated for all possible sub-sequences from T with length n then the smallest distance is selected.

Entropy: Let's assume there are two classes A and B present in a dataset D. Let $p(A)$ represent the probability that D belongs to A, and $p(B)$ is the probability that D belongs to B. The entropy between A and B is:

$$I(D) = -p(A)log(p(A)) - p(B)log(p(B)) \quad (2)$$

If D is partitioned into two non-overlapping parts D_1 with fraction $f(D_1)$ and D_2 with fraction $f(D_2)$, then the total entropy of D is:

Manuscript received October 31, 2014; revised February 25, 2015..

The authors are with the Mississippi State University, Mississippi State, MS 39762, USA (e-mail: ism6@msstate.edu, younan@ece.msstate.edu).

$$\hat{I}(D) = f(D_1)I(D_1) + f(D_2)I(D_2) \qquad (3)$$

Information gain: The information gain between D_1 and D_2 is defined as the difference between the entropies before and after partitioning.

$$Gain(D) = I(D) - (f(D1)I(D1) + f(D2)I(D2)) \qquad (4)$$

Optimal split distance: If dataset D is partitioned into two parts D_1 and D_2 and

$$d(S,p) < d, p \in D1, d(S,q) \geq d, q \in D2 \qquad (5)$$

then the optimal split distance d for subsequence S is given by:

$$Gain(S,d) \geq Gain(S,d') \text{ for all possible } d' \qquad (6)$$

Shapelet: Shapelet S is a subsequence in a dataset D for which

$$Gain\,(S,d) \geq Gain(S',d) \qquad (7)$$

where S' is any subsequence in the dataset and d is an optimal split point. In order to find a representative shapelet of a certain length, a brute force algorithm is applied in [3].

A. Brute Force Algorithm (BFA)

Let's assume that a training set D consists of two classes A and B. In the brute force algorithm defined in [3] a sub-sequence S_i from every training time series $T_j \in D$ is extracted through a moving window. Its length vary from *MIN_LENGTH* to *MAX_LENGTH*. Usually *MIN_LENGTH* is assigned to 3, the smallest practically meaningful shapelet length and *MAX_LENGTH* is defined as the length of the shortest time series in the training dataset. Every such sub-sequence is considered a potential shapelet. Then the distance between a candidate shapelet S_i and a time series T_j is calculated: $d_{i,j} = Dist(S_i,T_j)$. So found distances for S_i are ordered into a histogram. Then every adjacent point d between two consecutive distances from the histogram is labeled a split point. The fraction f_1 is assumed to be on the left side of the split point d, but the fraction f_2 is on its right side. Then the entropy for d is calculated according to (3) and the information gain according to (4). The optimal split point for S_i will be the one that produces maximum information gain. In a global aspect, the subsequence that produces the highest information gain is appointed as a shapelet. This algorithm generates $O(m^2k)$ candidate shapelets, where m is the average time series length and k is the number of training time series. On the other hand $O(mk)$ steps are required to check the candidate shapelet. In general the brute force algorithm requires $O(m^3n^2)$ steps, which is very time inefficient. The method showed significant calculation time in our experiments. For example, in the case of "*Shields*" dataset [5], where the time series length is 1080 samples, calculation time was 77127.7 *seconds*, which is more than 21 *hours*.

B. Subsequence Distance Early Abandon

As mentioned earlier to calculate $d_{i,j} = Dist(S_i,T_j)$, the smallest distance between S_i and the sub-series from T_j is required. The early abandon algorithm proposed in [3] stops calculating the Euclidian distance if it starts to exceed the current minimum distance. The time improvement in this case is relatively good, but not significant. For the "*Gun/No Gun*" dataset from [5] we obtained a training time of *708.1 seconds* for the brute force algorithm and the training time after applying early abandon had improved to *663.1 seconds*.

C. Admissible Entropy Pruning

The most expensive operation in the brute force algorithm is the calculation of the distance $d_{i,j} = Dist(S_i,T_j)$. For every subsequent candidate S_i, the algorithm can partially calculate $d_{i,j}$, only considering some of the time series T_j, but for the rest of them makes an optimistic prediction which distance to which part of the histogram belongs as proposed in [3]. In the case of the "*Gun/ No Gun*" database, we assumed the number of minimum calculated distances to be 5. The training set consists of 50 time series. For the remaining *45* time series, we set the distances to be either 0.0 or $\max(d_{i,j})+1$. Then the information gain in both cases is calculated and if it does not exceed the current best gain, then the calculation for S_i stops. The training time after applying entropy pruning and early abandon for the "*Gun/ No Gun*" dataset dropped significantly to 128.3 seconds.

D. Infrequent Shapelets

Another improvement of the original brute force algorithm is proposed by [6]. It uses the idea that the subsequences that uniquely identify certain class are much more rare than the rest of the subsequences in the time series. Also, the probability that an infrequent candidate shapelet can become a shapelet is much higher than for the rest of the candidates. The complexity of this algorithm is in the range of the brute force algorithm "in the worst case", although in practice it behaves well as it prunes non-frequent candidates according to [6]. Though, the reported training time improvement is significant (7-25 times) [6], the method is tested with limited amount of databases, as well it is not certain when (what dataset, what infrequent threshold, etc.) the worst case scenario may occur.

E. Time Series Qlets

The proposal in [7] applies unsupervised method to improve the performance of the time series shapelets method. The algorithm reduces the dataset complexity by extra quantization and indexing, after which it builds an ngram trie out of the indexed data. Then it calculates the boundary entropy and frequency of the pattern of indexed data and convert back to quantized data those with higher entropy and frequency. As proposed in [7], the method claims the number of calculation to be $O(MN)$ (M-length of time series dataset, N- "the size of expected dataset"), but does not show any particular processing times as a result.

III. Proposed Algorithm

The algorithm proposed in this work is based on the particle swarm optimization technique. PSO is a computational algorithm that iteratively optimizes a given solutions by applying mathematical rules and after estimating the fitness of a current solutions changes their coordinates into the search space. PSO was originally introduced by Kennedy, Eberhard, and Shi [8], [9] as an optimization technique inspired by the social behavior of bird flocks and fish herds. PSO utilizes a certain number of solutions, called particles that form a swarm. Every such particle has position and velocity coordinates in the search space. The velocity represents the change of the particle position from iteration to iteration. The change of the particle's position is dictated by the best so far known particle's position as well from the best position in the overall swarm. In this project, the PSO technique is used in the following manner. Every particle in the swarm represents a candidate shapelet. The number of candidate shapelets is equal to MAX_LENGTH – MIN_LENGTH+1. Every particle has different length from MIN_LENGTH up to MAX_LENGTH. The initial values for the particle position and velocity are taken directly from the training set. The mathematical rule that checks the particle fitness is chosen to be very similar to the one that checks candidate shapelets in BFA. It considers the particle as equivalent to the S_i sub-sequence from BFA and the particle's position coordinates corresponds to S_i samples. After calculating $d_{i,j}$ distances and building the histogram of distances, the information gain is calculated. If that gain is bigger than the best so far information gain assigned to this particle, the current position becomes the particle's best position. The process is repeated for every particle in the swarm, then the best particle is chosen to be the one with the highest information gain. The position of the best particle in the swarm and the best so far position of certain particle define the particle's velocity, respectively the particle's next position. In the process of the calculation of a new particle's position, the randomness that is defined by two numbers *R1*, *R2* (randomly generated on every iteration step) plays an important role. The iteration process stops when a certain number of iteration is achieved or when the best information gain from the current iteration does not exceed much the best information gain from the previous iteration. After the iteration process stops the best particle in the swarm is selected to be a shapelet. The function *UpdatePSO* (Listing 1) finds the best solution in the swarm. The PSO process uses *CheckCandidate* (Listing 2) as a fitness function, which resembles the function used by the brute force algorithm. The proposed algorithm uses sub-sequences from the training set only for initialization. Then, these sequences are manipulated in order to obtain the solution that brings the highest information gain. For every iteration steps, $O(m^2)$ calculations are made to update the position of the particle and $O(mk)$ to check the fitness of the solution, where m is the average length of a particle and k is the number of training time series in a dataset. In other words the whole process takes approximately $O(m^2l)$ steps, where l is the number of iteration in the PSO algorithm. The number of iteration could be very small in practice. Our tests show that sufficient results could be obtained for less than *50* iterations.

To build the classification tree, we find the shapelets for all the possible pairs of classes in the class pool. Then, the algorithm builds all possible variations of classification trees out of these shapelets and select the tree with the highest accuracy. In the case of the "*Arrowhead*" dataset [5] for example, there are three classes indexed as 0, 1, and 2. Fig.1 shows classification tree variations for the "*Arrowhead*" dataset along with the accuracies they produce. The tree variations are built during the training phase of the algorithm thus shown accuracies are obtained after testing with the time series from the training dataset. The tree with the highest accuracy is selected as a classification tree for the class pool. Once the classification tree was selected, the actual testing phase was done with time series from the testing dataset.

```
UpdatePSO
{
  Do
    ForEach Particle in Swarm

      For j = 0 to ParticleLength
        Partcle.Velocity[j] = W * Partcle.Velocity[j] +
        C1*R1*Particle.BestPosition[j] - Particle.Position[j] +
        C2*R2*BestParticle.Position[j] - Particle.Position[j]
      EndFor

      For j = 0 to ParticleLength
        Partcle.Position[j] += Partcle.Velocity[j]
      EndFor

      CheckCandidate (Particle)

      If (Particle.BestInfoGain > BestParticle.BestInfoGain)
        BestParticle = Particle
      EndIf

    EndForEach

    OldBestGain = NewBestGain
    NewBestGain = GetSwarmBestInformationGain

  While ( (OldBestGain - NewBestGain) > EPSILON )

  BestShapelet = BestParticle
}
```

Listing 1: Pseudo code for iterations calculation (proposed PSO algorithm).

```
CheckCandidate(Particle)
{
  Distances ← Initialize

  ForEach TimeSeries in TrainDataSet_ClassA_And_ClassB

    Distance = MinDistance(Particle.Position, TimeSeries)
    Distances ← Add(Distance)

  EndForEach

  Histogram = OrderDistances(Distances)

  InforGain = CalculateInformationGain(Histogram)

  If (InforGain > Particle.BestInfoGain)
    Particle.BestInfoGain = InfoGain
    Particle.BestPosition = Particle.Position
  EndIf
}
```

Listing 2: Pseudo code for particle validation (proposed PSO algorithm).

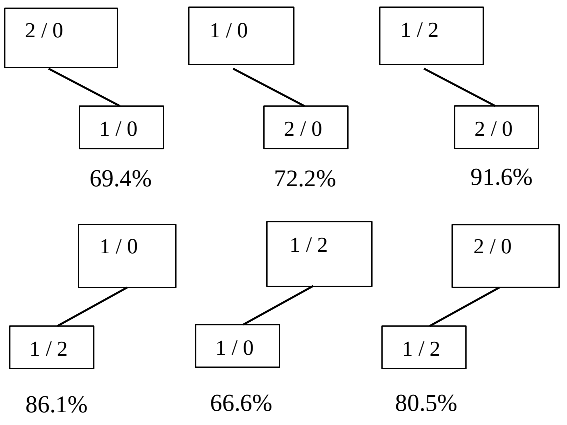

Fig. 1. Possible classification tree variations for the "*Arrowhead*" dataset [5] along with accuracies they produce.

IV. Experimental Results

The algorithms were tested on a PC with the following parameters: CPU: Intel Core i7, 2.4GHz; RAM: 8 MB; 64-bit Windows 7 OS. The algorithms were implemented in C# and .NET Framework 4.0. Time performance measurements were done through *System. Diagnostics Stop Watch*. NET class. MIN_LENGTH was assigned to 3 and MAX_LENGTH to length close to the minimum length of the training time series as shown in Table I. The constants used in the PSO algorithm were selected as [10]: W= 0.729, C_1= 1.49445, and C_2 = 1.49445. Table II shows the results from the brute force algorithm with early abandon and entropy pruning switched "on". Table III presents the results obtained from the proposed PSO algorithm on the same datasets and with the same MIN_LENGTH and MAX_LENGTH as these used in BFA. The results show that the accuracy of the proposed PSO algorithm to be similar to that of the BFA algorithm, but the training time dropped significantly and is 105 to 675 times smaller than in BFA (early abandon and entropy pruning switched "on"). The tests were done with time series from the test dataset and shown accuracies correspond to the ratio of the number of correctly recognized over all test time series. The training time for both algorithms, the BFA and the proposed PSO algorithm depends on the training dataset length. For the proposed PSO algorithm this dependency is much less, thus the training time improvement is more significant for longer time series. For example, the time series length in the "*Shields*" dataset [5] varies from 1086 to 1224 samples and the time improvement is ~675 times, but for the "*Gun/NoGun*" dataset where the time series length is just 150 sample, this improvement is much less ~105 times. Another interesting fact is that the proposed PSO algorithm outperforms the BFA algorithm in terms of accuracy in datasets where the time series have different lengths ("*Arrowhead*", "*Shields*", "*Toe segm.* 1"). This may come from the fact that the selected MAX_LENGTH may not work well for the BFA algorithm, but gives good results for the proposed PSO algorithm. Our experiments confirmed that varying MAX_LENGTH influences the accuracy of both the BFA and proposed PSO algorithm.

TABLE I: Lengths of the Time Series in Used Training Datasets [5] and the Value of the Corresponding MAX_LENGTH

Dataset	Time series length	MAX_LENGTH	Number of classes
Gun / NoGun	151	150	2
Arrowhead	476 - 624	120	3
Shields	1086 - 1224	1080	3
Coffe	287	245	2
Mallet (First 5 classes)	257	200	5
Toe segm. 1	484 - 578	250	2
Toe segm. 2	345	340	2
Toe segm. 3	280 - 490	277	2

TABLE II: Performance of the Brute Force Algorithm with Early Abandon and Entropy Pruning Switched "on"

Database	Training Time [s]	Accuracy [%]
Gun / NoGun	117.71	69.63
Arrowhead	1555.16	58.09
Shields	77127.79	62.68
Coffe	239.37	84.62
Mallet (First 5 classes)	10622.41	55.38
Gait 1	2529.03	89.07
Gait 2	824.42	84.24
Gait 3	769.29	73.59

TABLE III: Performance of the Proposed PSO Algorithm

Database	Training Time [s], Energy tree	Accuracy [%], Energy tree
Gun / NoGun	0.88	78.62
Arrowhead	14.26	53.81
Shields	229.89	76.11
Coffe	2.6	73.08
Mallet (First 5 classes)	65.06	73.62
Gait 1	9.16	75.69
Gait 2	4.34	87.07
Gait 3	5.56	84.55

V. Conclusion and Future Work

We have proposed a new training method for the time series shapelets algorithm that improves the training time significantly keeping relatively high accuracies. It is based on particle swarm optimization and have the best performance to any similar methods as of our knowledge. A variety of benchmark datasets were used with a variety of time series lengths and number of classes to prove the consistency of the proposed method. The method shows good performance in terms of training time and accuracy for all tested databases.

Future works include testing of the proposed PSO algorithm with more datasets as well with variety of MAX_LENGTHs. As the base particle swarm optimization algorithm is very suitable for parallel processing [11], [12], it is highly desirable to check the improvement of the training time of the proposed PSO algorithm on a GPU processors. In addition, the method used to find the best classification tree out of all possible combination of classification trees is a bit extensive especially for a higher number of classes (more than (5). Thus, more work to improve this techniques should be done.

REFERENCES

[1] C. Guo, H. Li, and D. Pan, "An improved piecewise aggregate approximation based on statistical features for time series mining," in *Proc. KSEM*, 2010, pp. 234-244.

[2] E. Keogh, L. Wei, X. Xi, S. Lee, and M. Vlachos, "LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," *VLDB*, September 2006.

[3] L. Ye and and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proc. the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pp. 947-956, 2009.

[4] 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," in *Proc. the VLDB Endowment*, vol. 1, no. 2, pp. 1542-1552, Aug. 2008.

[5] L. Ye. (2009). The time series shapelet Webpage. [Online]. Available: http://www.cs.ucr.edu/~lexiangy/shapelet.html

[6] Q. He1, Z. Dong, F. Zhuang, T. Shang, Z. Shi, "Fast time series classification based on infrequent shapelets," in *Proc. the 2012 11th International Conference on Machine Learning and Applications*, 2012.

[7] A. Anand and V. Padmanabhan, "Time series qlet: Invriant approach for data mining," in *Proc. the 2013 Sixth International Conference, Contemporary Computing (IC3)*, August 2013.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. the IEEE International Conference on Neural Networks*, vol. IV. pp. 1942-1948, 1995.

[9] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. the IEEE International Conference on Evolutionary Computation*, pp. 69-73, 1998.

[10] J. McCaffrey. (2006). Particle swarm optimization. MSDN Magazine. [Online]. Available: http://msdn.microsoft.com/en-us/magazine/hh335067.aspx

[11] I. Liera, M. Liera, and M. Castro, "Parallel particle swarm optimization using GPGPU," *CIE*, 2011.

[12] V. Roberge and M. Tarbouchi, "Parallel particle swarm optimization on graphical unit for pose estimation," *WSEAS Transactions on Computers*, 2012.

[13] D. J. Berndt and J. Clifford, "Using dynamic timewarping to find patterns in time series," in *Proc. AAAI-94 Workshop on Knowledge Discovery in Databases*, Seattle, Washington, 31 July 1994.

[14] X. Xi, E. Keogh, C. Shelton, and L. Wei, "Ratanamahatana, fast time series classification using numerosity reduction," in *Proc. the 23rd International Conference on Machine Learning, Pittsburgh, Pennsylvania, ICML '06,* ACM, New York, NY, vol. 148, pp. 1033-1040, June 25-29, 2006.

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 an executive committee member of the international conference on high voltage engineering and applications(2010-2014).