2016 IEEE 28th International Conference on Tools with Artificial Intelligence

A Shapelet Learning Method for Time Series Classification

Yi Yang[1], Qilin Deng[1], Furao Shen[1,*], Jinxi Zhao[1] and Chaomin Luo[2]
[1]National Key Laboratory for Novel Software Technology, and Department
of Computer Science and Technology, Nanjing University, China
[2]Department of Electrical and Computer Engineering, University of Detroit Mercy
Email: yangyi868@gmail.com, qldeng@qq.com, frshen@nju.edu.cn, jxzhao@nju.edu.cn, luoch@udmercy.edu

***Abstract*—Time series classification (TSC) problem is important due to the pervasiveness of time series data. Shapelet provides a mechanism for the problem by its ability to measure local shape similarity. However, shapelets need to be searched from massive sub-sequences. To address this problem, this paper proposes a novel shapelet learning method for time series classification. The proposed method uses a self-organizing incremental neural network to learn shapelet candidates. The learned candidates reduce greatly in quantity and improve much in quality. After that, an exponential function is proposed to transform the time series data. Besides, all shapelets are selected at the same time by using an alternative attribute selection technique. Experimental results demonstrate statistically significant improvement in terms of accuracies and running speeds against 10 baselines over 28 time series datasets.**

***Index Terms*—Time Series, Classification, Shapelet, Data Mining**

I. Introduction

Time series classification (TSC) problem is important due to the pervasiveness of time series analysis, including statistics, signal processing, mathematical finance, earthquake prediction, electroencephalography, etc [1]. As a consequence, in the last decade vast amount of methods have been proposed to solve this problem. Among these methods, a recently proposed method named shapelet [4] provides a mechanism for the problem by its ability to measure local shape similarity. Shapelets are time series sub-sequences which are in some sense maximally representative of a class [6]. Shapelets provide not only interpretable results, but also incredibly high prediction accuracies on some data sets [5].

The original shapelet algorithm [4] constructs a decision tree classifier by recursively searching for the most discriminatory sub-sequence via an information gain measure. Because of the high time complexity, speed-up techniques such as early abandoning of distance computations and entropy pruning of the information gain metric [4], [9] are typically used with the brute force algorithm. Other speed up techniques such as SAX representation [7], infrequent shapelet [8] and using GPU [11] have also been proposed to accelerate the discovering of shapelets. Moreover, in [6], shapelets are directly learned via optimization of a classification objective function, instead of searching from the sub-sequences. For the purpose of improving the accuracy rate, ensemble methods are also used with shapelet algorithm [12].

In order to promote the accuracy, Lines J et al propose a shapelet transform algorithm [5]. They find the best k shapelets via an alternative quality metric, such as F- statistics or Kruskall-Wallis. And then, time series data are transformed into shapelet feature space in which each feature is the distance between a time series and a shapelet. After that, the transformed data are used to train a classifier, such as Bayesian Network or SVM. Their experimental results show that the transformed data, in conjunction with complex classifiers, provide better accuracies than the embedded shapelet tree. However, since the algorithm separates the procedure of finding shapelets from training of classifier, it actually gets the best k shapelets but not the best shapelets combination. In other words, it does not consider interactions among the shapelets.

Until now, except the learning shapelets algorithm [6], all other shapelet-based algorithm need to search shapelets from massive sub-sequences of the time series in a data set. But the best shapelet may not be among the sub-sequences. For example, the average of two sub-sequences may provides a better shapelet than either one. On the other hand, the quantity of sub-sequences is very large, which makes the time complexity of searching shapelets pretty high. Although speed-up techniques are used, the discovery of shapelets is still very slow.

In this paper, we propose a shapelet learning method for time series classification. Instead of using sub-sequences as candidates directly, we introduce a very fast algorithm to learn candidates from sub-sequences. The quantity of the learned candidates is significantly reduced compared to that of bruth force algorithm. After that, we propose a new data transform method rather than using the distance as the transformed attribute value directly. Same with the original shapelet tree algorithm, we embed the procedure of shapelets discovery in training of classifier for the sake of getting better shapelets combination. The experimental results on 28 data sets show that the proposed algorithm is about thousands times faster than the transform algorithm in [5] and provides competitive results. Our contributions can be summarised as follows:

1) A very fast algorithm is introduced to learn candidates from sub-sequences.

Corresponding author is Furao Shen.

2375-0197/16 $31.00 © 2016 IEEE
DOI 10.1109/ICTAI.2016.68

2) An exponential function is used to transform the data instead of using the distance as the transformed attribute value directly.
3) We show that alternative attribute selection techniques can be also applied to shapelet selection.
4) All shapelets are learned at the same time instead of one by one, which speeds up the searching time greatly.

The paper is structured as follows. Section II provides definitions and backgrounds of shapelet-based time series classification algorithm. In Section III, we introduce our proposed shapelet learning method. In Section IV, we provide an analysis of the proposed method. In Section V, we describe our experimental design and evaluate the proposed method. Finally, in Section VI, we conclude the paper and discuss future work.

II. Definitions and Background

A. Time Series Classification

A time series is a sequence of data points that typically consists of successive measurements made over a fixed time interval. Given a time series of length m, $T =< t_1, t_2, ..., t_m >$, the task of time series classification is to assign T to one or more classes or categories.

B. Definitions

In this section, we give definitions about time series and shapelet.

Definition 1 (Time Series): A time series $T =< t_1, t_2, ..., t_m >$ is a sequence of m real-valued variables. The database D consists of n time series and each time series has a class label.

Definition 2 (Sub-sequence): Given a time series T of length m and a start point p, a sub-sequence S of T is a sampling of length $l \leq m$ of contiguous positions from T, that is $S_p^l =< t_p, ...t_{p+l-1} >$, for $1 \leq p \leq m - l + 1$. All possible sub-sequences of length l can be extracted by sliding a window of size l across T and considering each sub-sequence S_p^l of T. The set of all sub-sequences of length l extracted from T is defined as S_T^l, $S_T^l = \{S_p^l$ of T,for $1 \leq p \leq m - l + 1\}$. **Note that all sub-sequences are zero-mean normalized to solve the problem caused by different scales.**

Definition 3 (Distance between sub-sequences of the same length): Given two sub-sequences S_1 and S_2 , both of length l, the distance between them is shown in Equation 1.

$$dist(S_1, S_2) = \sum_{i=1}^{l} (S_{1,i} - S_{2,i})^2 \tag{1}$$

Definition 4 (Distance from the time series to the sub-sequences): The distance between a time series T and a sub-sequence S of length l is the distance between S and the sub-sequence whose distance to S is smallest among all sub-sequences of T of length l. It's defined as :

$$sdist(T, S) = \min_{1 \leq p \leq m-l+1} dist(S_p^l \; of \; T, S) \tag{2}$$

Definition 5 (Shapelet): The idea of shapelet is similar to bag-of-words which is often used in text classification[13] and image classification[14]. A shapelet is actually a time series word. Shapelet provides information about whether the time series contains a sub-sequence which is similar to the shapelet and how similar the sub-sequence is to the shapelet. A shapelet of length l is simply an ordered sequence of values from a data structure perspective. Nevertheless, shapelets semantically represent intelligence on how to discriminate the class or category of a time series. Distance between shapelets and time series is the same as distance between sub-sequences and time series.

Definition 6 (Shapelet feature space): Suppose we have k shapelets denoted as $< S_1, S_2, ..., S_k >$, given a time series T, we can transform T into shapelet feature space. In shapelet feature space, each feature represents the information about whether the time series contains a sub-sequence which is similar to the shapelet and how similar the sub-sequence is to the shapelet. Denote $Trans(T)$ as the transformed data of T in shapelet feature space. An easy way to do transformation is using the distance directly as the transformed attribute value. In this situation, $Trans(T) =< sdist(T, S_1), sdist(T, S_2), ..., sdist(T, S_k) >$.

C. Brute Force Algorithm

The most straightforward way for searching shapelets is using the brute force method. Firstly, we generate candidates which consist of all sub-sequences of all user-defined lengths and store them in an unordered list. And then, we evaluate each candidate by an user-defined criterion and choose the best k shapelets. The criterion can be information gain, F-statistics or other statistic informations. The original shapelet tree algorithm [4] constructs a decision tree at the same time of searching shapelets. Other classifiers such as SVMs, Bayesian Network, can also be applied to promote the accuracy after we transform the time series data into shapelet feature space.

III. Proposed Method

A. Basic Idea

The main reason why the brute force algorithm is time consuming is that the quantity of the sub-sequences is too large, and among the sub-sequences there are many sub-sequences which are similar to each other. This inspires us to use one sequence to represent several similar sub-sequences. We call this sequence a prototype. A prototype is the average of several similar sub-sequences. On the one hand, prototype provides possibilities for improving the quality of candidates. On the other hand, prototype reduces the quantity of candidates significantly. Instead of using sub-sequences as candidates directly, we learn prototypes using sub-sequences and consider each prototype a candidate.

Until now, because the quantity of candidates is too large, no shapelet-based algorithm transforms the time series data into shapelet feature space before shapelets selection. However, if we want to use common attribute selection techniques to select shapelets, transforming the data before shapelets selection is inevitable. Since the quantity of candidates is significantly reduced if we use prototypes as candidates, it is possible for us to transform the time series into shapelet feature space using the candidates directly. And then, stronger classifier and alternative attribute selection techniques can be applied on the transformed data. Using the selected attributes, we can find the corresponding candidates and consider them as shapelets. We name our shapelet learning algorithm LCTS since our method consists of three component: **L**earning **C**andidates, **T**ransforming the data into shapelet feature space, and selecting **S**hapelet via attribute selection techniques.

B. Learning Candidates

In LCTS, prototypes are learned using a self-organizing incremental neural networks(SOINN) [15]. For the reason of lacking space, we just explain what is SOINN and what SOINN can do here. The readers are recommended to refer [15] for the details of SOINN. SOINN is an on-line unsupervised learning mechanism for unlabeled data. SOINN can learn the topological structure of unsupervised on-line data, report the reasonable number of clusters, and give typical prototype patterns of every cluster without prior conditions [15]. From data compression perspective, SOINN learns representative data from a large quantity of data. From clustering perspective, SOINN is analogous to k-means. Given a dataset, suppose that SOINN learns k prototypes, if we use the k prototypes as the initial values of k-means, the result of k-means is similar to SOINN in this situation.

Input a data set, SOINN generates a topology structure which consists of a node set and a connection set. In our method, we only use the node set. Each node is a prototype. Another concept of SOINN we use in our method is the threshold of the prototype. The threshold of prototype P is the distance between P and the prototype which is furthest to P among prototypes which have a connection with P. Fig.1 provides a sketch map of the input and output of SOINN. As an unsupervised topology learning methods, SOINN has several properties:

1) SOINN is an incremental algorithm, which means that it needs only one pass on the data.
2) SOINN can de-noise the data automatically.
3) Each prototype learned by SOINN is the average of several similar data.

These properties are exactly what we need for learning prototypes. Based on this, we use SOINN in the first place as the prototype learning methods.

Given a length l, we generate all sub-sequences of length l from the time series in the database D. **Note that we zero-mean normalize each sub-sequence to solve the problem caused by different scales.** After that, we use SOINN to learn prototypes using the sub-sequences. Each learned prototype is a candidate.

We learn candidates of all searching lengths. Most shapelet-based algorithm use an user-defined $maxLength$ and an user-defined $minLength$ to define searching lengths as $minLength \leq l \leq maxLength$. But this method generates too much searching lengths and the disparities between the searching lengths are too small which means it is difficult for the method to learn different scale shapelets. Instead, we use a simple idea to generate searching lengths. Given $maxLength$, $minLength$, we start from $L = maxLength$. We add L into searching lengths and divide L by 2, then repeat the procedure until $L < minLength$. This simple idea generates different scale searching lengths and reduces the number of searching lengths greatly.

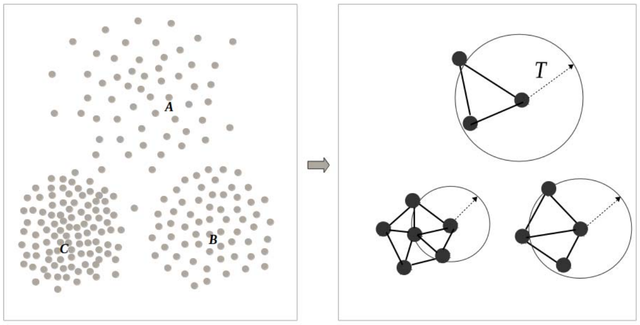

Fig. 1: Left is the input data. Right is the topology structure generated by SOINN. SOINN learns 11 representative prototype for the input data. Each prototype has a threshold T which is the distance between the prototype and its furthest neighbor prototype. Generally T is smaller in data-intensive areas than that of data-sparse areas.

C. Data Transformation

After learning candidates, we transform the time series into shapelet feature space. Since our final objective is to select shapelets from candidates, we hope the transformation could help us select shapelets. Given a time series T and a prototype P, denote $h(T, P)$ as the transformed value of T on P. Define S_P^T is the sub-sequence which is most similar to P among all sub-sequences of T. Note that the definition of S_P^T here is different from that of Section 2 and it is used in this whole subsection. We want that $h(T, P)$ is a value between 0 and 1 to represent how similar S_P^T is to P, while 0 represents a totally different and 1 represents exactly the same. There is a property that the smaller $dist(S_P^T, P)$ is, the more important a little change of $dist(S_p^T, P)$ is. Intuitively, the impact of $dist(S_P^T, P)$ increased by 1 on similarity between S_P^T and T when $dist(S_P^T, P) = 1$ is bigger than that of when $dist(S_P^T, P) = 100$. To express the property formally, denote $|x|$ as the absolute value of x and $f'(x)$ as the gradient of f on x, if we use a function f of the distance between the time series and the shapelet as the transform function, $|f'(x)|$ should decline with the increase of x. There are many functions which satisfy this condition. We choose the exponential function $f(x) = \exp(-\alpha \cdot x)$.

After that, we need to normalize the distance between the time series and the candidates. Denote T_P as the threshold of P. Generally T_P is smaller in data-intensive areas than that of data-sparse areas. This property of T_P can be easily seen in Fig.1. Denote D^P_{avg} as the average distance between P and its neighbor sub-sequences. It is obvious that D^P_{abg} is smaller in data-intensive areas that that of data-sparse areas. So T_P is positively correlated with D^P_{avg}. Suppose the distance between time series T and P is fixed, it is not hard to discover that the bigger D^P_{avp} is, the more similar S^T_P is to P. We would like to give an illustration of this property in the next paragraph. To conclude, the similarity of S^T_P and P is positively correlated with T_P. It is obvious that the similarity of S^T_P and P is negatively correlated with the distance between T and P. So we use $\frac{sdist(T,P)}{T_P}$ as the normalized distance between time series T and prototype P.

Fig.2 gives an illustration of the normalization method. A time series T from the FaceFour dataset and two prototypes $P1$ and $P2$ are depicted in Figure 2. The distance between T and $P1$ is smaller than the distance between T and $P2$. The volatility of $P2$ is higher than that of $P1$ which results in that the sparsity of data around $P2$ is larger than that of $P1$ and $T_{P2} > T_{P1}$. Intuitively, the similarity between S^T_{P2} and T is higher than that of S^T_{P1} and T because S^T_{P2} and $P2$ matches more waves than S^T_{P1} and $P1$. So the origin distance does not reflect the real similarity relationship. However, the normalized distance $\frac{sdist(T,P1)}{T_{P1}}$ is bigger than $\frac{sdist(T,P2)}{T_{P2}}$, which reflects the real similarity relationship.

Combine the exponential function and distance normalization method, now we have our final transform function as follows:

$$h(T,P) = \exp(-\alpha \cdot \frac{sdist(T,P)}{T_P}), \alpha > 0 \tag{3}$$

α is an user-defined parameter. It controls the scale of transformed attribute value. Given a time series T and k prototypes $< P_1, ..., P_k >$, the transformed data are $Trans(T) =< h(T,P_1), ..., h(T,P_k) >$.

D. Shapelets Selection

The shapelets tree [4] constructs a decision tree classifier by recursively searching for the most discriminatory sub-sequence via an information gain measure. It selects shapelets at the same time of training a classfier, but the decision tree is weak classifier which makes the accuracy of the original shapelets algorithm a bit low. To promote the accuracy, Lines J et al [5] select shapelets via some alternative quality measure metric and then apply stronger classifiers. Although they separate the procedure of selecting shapelets and training classifier, their results indicate that the support vector machine(SVM) is the best classifier on the transformed data [5]. That provides an inspiration for us to use SVM as the classifier. In order to embed the shapelet discovery algorithm in the classifier, we add a common used attribute selection technique named the L1-regularizer [10] to SVM. It's well known that L1-regularizer can avoid over-fitting and generate sparse solution. The sparse solution provides a way for us to select shapelets from candidates.

We use the least squares support vector machine [16] as the classifier. Denote w as the weight of SVM, x_i as the ith data in $transData$, y_i as the class label of ith data, C as the regularization parameter. For two-class problem, The loss function is defined as follows:

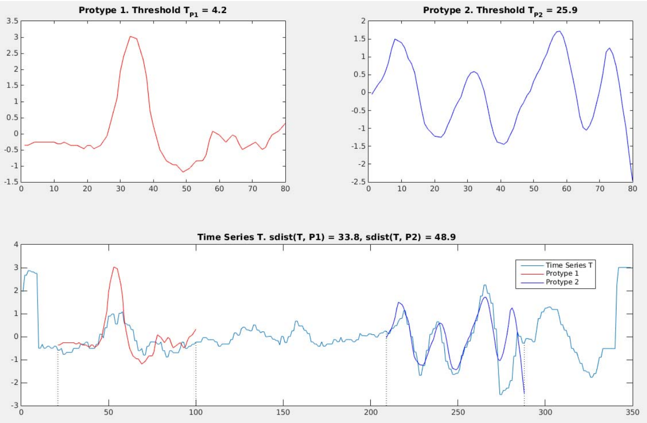

Fig. 2: Illustration of a time series T and two prototypes $P1$ and $P2$. Distance between T and $P1$ is smaller than that of T and $P2$. But the similarity between $P2$ and S^T_{P2} is higher than that of $P1$ and S^T_{P1} intuitively. The normalized distance $\frac{sdist(T,P1)}{T_{P1}}$ is smaller than $\frac{sdist(T,P2)}{T_{P2}}$, which reflects the real similarity relationship.

$$L(w) = \sum_{j=1}^{k} |w_j| + C \sum_{i=1}^{n} max(0, 1 - y_i w^T x_i)^2 \tag{4}$$

For multi-class problem, we implement one-vs-the rest multi-class strategy for classification.

After minimize the loss function, we get a sparse weight matrix. For each candidate, if all weights associated with it are zero, we remove it from the candidate set and remove the weights associated with it from the weight matrix. Finally, the rest candidates in the candidate set are selected shapelets.

IV. Analysis of the Method

A. Learn Better Candidates

Instead of considering sub-sequences as candidates directly, we use the self-organizing incremental neural networks (SOINN) to learn candidates from the sub-sequences. Each candidate represents the common character of several similar sub-sequences. Isolated sub-sequences are considered as noisy points in SOINN and are removed. Similar sub-sequences are combined to one candidate. Therefore, the learned candidates are better than sub-sequences themselves both quantity and quality.

B. Time Complexity

Recall that the number of the time series objects in the dataset is n, the length of each time series is m, the number of learned candidates or prototypes is k. The baseline method which exhaustively tries candidates from sub-sequences requires $O(n^2 m^3)$ running time for discovering the best shapelet of a particular length [4]. LCTS consists of three components.

Firstly, the time complexity of learning prototypes using SOINN is $O(knm^2)$. And then, data transformation needs one-pass of the database and the time complexity is also $O(knm^2)$. Lastly, in the third component, we train a L1-regularized SVM. If we denote i as the number of iterations, q as the number of time series classes, the time complexity of this component is $O(iknq)$. To sum up, the time complexity of LCTS is $O(2knm^2 + iknq)$. Given that usually $k \ll nm$ and $iq \ll nm^2$, LCTS runs in a faster time. Experiment results show that usually k is about hundredths of nm.

C. Select Shapelets Combination

Lines J et al select shapelets via some quality measure metric, search each shapelet independently and get the best k shapelets which score highest. However, such an approach does not take into account interactions among the shapelets. In other words, two shapelets can be individually sub-optimal, but when combined together they can improve the results. This problem is also explained in [6]. It's impossible for the baseline method to consider interactions among the shapelets because the quantity of candidates combination is too large.

Since LCTS selects shapelets through L1-regularizer, shapelets are selected by the classifier itself and all shapelets are selected at the same time. It makes it possible for the algorithm to consider the interactions among the shapelets and select better shapelets combination.

TABLE I: Dataset informations

Dataset	Train/Test	Length	Cls.
Adiac	390/391	176	37
Beef	30/30	470	5
Beetle/Fly	20/20	512	2
Bird/Chicken	20/20	512	2
Chlorine.	467/3840	166	3
Coffee	28/28	286	2
Diatom.	16/306	345	4
DP_Little	400/645	250	3
DP_Middle	400/645	250	3
DP_Thumb	400/645	250	3
ECGFiveDays	23/861	136	2
FaceFour	24/88	250	4
GunPoint	50/150	150	2
ItalyPower.	67/1029	24	2
Lighting7	70/73	319	7
MedicalImages	381/760	99	10
MoteStrain	20/1252	84	2
MP_Little	400/645	250	3
MP_Middel	400/645	250	3
Otoliths	64/64	512	2
PP_Little	400/645	250	3
PP_Middle	400/645	250	3
PP_Thumb	400/645	250	3
SonyAIBO.	20/601	70	2
Symbols	25/995	398	6
SyntheticControl	300/300	60	6
Trace	100/100	275	4
TwoLeadECG	23/1139	82	2

D. Weak Aspect of LCTS

There are still a lot of similar candidates although we learn candidates using SOINN, which causes that the L1-regularizer remains some similar candidates as selected shapelets. But it's not serious, experimental results show that the number of shapelets selected is reasonable.

TABLE II: Parameters Setup and Number of Learned Shapelets

Dataset	$MaxLen/MinLen$	α	C	Num.
Adiac	104/13	2	1	478
Beef	200/12	0.5	10	109
Beetle/Fly	96/24	0.5	10	17
Bird/Chicken	80/10	0.5	10	23
Chlorine.	96/12	5	1	152
Coffee	32/16	5	10	14
Diatom.	150/75	0.5	10	35
DP_Little	120/15	2	1	324
DP_Middle	60/6	2	1	347
DP_Thumb	120/15	2	1	378
ECGFiveDays	40/20	5	10	13
FaceFour	80/10	1	10	63
GunPoint	60/15	1	10	30
ItalyPower.	24/24	1	10	16
Lighting7	319/40	0.5	10	237
MedicalImages	95/5	0.3	1	489
MoteStrain	80/5	1.2	10	20
MP_Little	120/15	1	1	350
MP_Middel	100/12	1	1	350
Otoliths	200/12	0.2	10	284
PP_Little	100/12	0.7	1	291
PP_Middle	110/25	0.7	1	245
PP_Thumb	120/15	0.7	1	337
SonyAIBO.	25/25	1	10	16
Symbols	100/50	0.07	10	97
SyntheticControl	60/7	0.5	10	135
Trace	36/36	1	10	23
TwoLeadECG	20/20	1.2	10	13

V. Experimental Results

A. Datasets

For the sake of equivalent comparison, we use exactly the same set of datasets as the baselines [5], [6]. The set contains 28 datasets whose details are shown in Table I. All datasets can be downloaded from the UCR [2] and UEA2 [3] websites.

B. Baselines

The baselines that we compare with contain three categories:

1) The original Shapelet Tree [4] constructed by recursively searching for the most discriminatory sub-sequence via an information gain measure.
2) Classifiers learned over shapelet transformed data proposed by [5], such as C4.5 tress(C4.5), Naive Bayes (NB), Nearest Neighbors (1NN), Bayesian Networks (BN), Random Forest (RAF), Rotation Forest (ROF) and Support Vector Machines (SVM).
3) Other shapelet-based classifiers :The Fast Shapelets (FSH) [7] speeds up Shapelet Tree method by using SAX representation of time series. The Learning Shapelets (LTS) [6] learns shapelets directly by minimizing a classification loss function.

TABLE III: Accuracies of Shapelet Tree, C4.5 and LCTT on 28 datasets

Dataset	Shapelet Tree	C4.5	LCTT
Adiac	0.299	0.243	**0.568**
Beef	0.500	**0.600**	0.467
Beetle/Fly	0.775	0.750	**0.900**
Bird/Chicken	0.850	**0.900**	0.550
Chlorine.	0.588	0.565	**0.598**
Coffee	**0.964**	0.857	0.929
Diatom.	0.722	0.752	**0.821**
DP_Little	0.654	**0.659**	0.546
DP_Middle	0.705	**0.712**	0.597
DP_Thumb	**0.581**	0.580	0.547
ECGFiveDays	0.775	**0.962**	0.929
FaceFour	**0.841**	0.761	0.818
GunPoint	0.893	0.907	**0.980**
ItalyPower.	0.892	0.910	**0.944**
Lighting7	0.493	0.534	**0.562**
MedicalImages	0.488	0.449	**0.555**
MoteStrain	0.825	**0.844**	0.790
MP_Little	**0.664**	0.634	0.611
MP_Middel	0.710	**0.733**	0.592
Otoliths	**0.672**	0.656	0.563
PP_Little	**0.596**	0.574	0.512
PP_Middle	0.614	**0.625**	0.581
PP_Thumb	**0.608**	0.595	0.522
SonyAIBO.	0.845	0.845	**0.952**
Symbols	**0.780**	0.471	0.565
SyntheticControl	**0.943**	0.903	**0.943**
Trace	0.980	0.980	**1.00**
TwoLeadECG	0.851	0.853	**0.961**

C. *Parameters Setup and Reproducibility*

$MaxLength$ and $MinLength$ are chosen according to the length of the time series and the difficulty of distinguishing time series of different categories. α controls the scale of transformed data, we choose α from those values that makes the average transformed attribute value between 0.3 and 0.7. C is the L1-regularizer parameter. In our experiments, we set $C = 10$ firstly, and if this value generates too much shapelets (exceeds the length of the time series), we adjust it to 1. The detail parameters setup is described in Table II. **In order to promote reproducibility, the source code and datasets are made publicly available**[1]**.**

D. *Validity of Learned Candidates and Transformed Data*

Since we use a novel idea to learn candidates and transform the time series, we need to validate the validity of the idea. We train a C.5 tree denoted as LCTT (derived from **L**earning **C**andidates, **T**ransforming the data and decision **T**ree) on the data transformed by our method described in Section III. We compare the result of LCTT with Shapelet Tree method and C4.5 trained on transformed data proposed by [5]. Since the three methods are all decision tree classifier, the comparison is meaningful. The results of the three method on 28 datasets are show in Table III.The results show that the three methods are neck and neck on the 28 datasets. But note that the searching lengths we used in our method are far less than that of the other two method. And since the candidates in our method are results of SOINN, the quantity of candidates in our method is far smaller than that of the other two methods. Therefore, the results show that our method of learning candidates and transforming data is validity.

TABLE IV: Running Time of Our Method and 2 baselines

Dataset	F-Stat (Sec)	LTS (Sec)	LCTS (Sec)
Adiac	4509.91	3017.23	**13.722**
Beef	1251.21	293.68	**6.498**
Beetle/Fly	21496.51	131.015	**1.323**
Bird/Chicken	20465.63	81.405	**2.168**
Chlorine.	15681.39	558.51	**6.86**
Coffee	258.15	90.96	**0.197**
Diatom.	53.91	173.1	**1.858**
DP_Little	78005.7	1525.595	**20.378**
DP_Middle	91208.51	910.33	**13.972**
DP_Thumb	123766.49	963.765	**24.376**
ECGFiveDays	149.1	29.365	**0.064**
FaceFour	4556.41	386.45	**2.525**
GunPoint	569.42	46.69	**0.543**
ItalyPower.	1.75	10.285	**0.005**
Lighting7	14912.74	394.44	**11.444**
MedicalImages	7742.97	406.725	**17.207**
MoteStrain	10.76	16.875	**0.179**
MP_Little	88071.5	965.27	**55.039**
MP_Middel	134731.54	940.555	**33.849**
Otoliths	55874.19	407.835	**82.292**
PP_Little	79993.31	890.925	**21.752**
PP_Middle	57874.19	407.835	**17.793**
PP_Thumb	91401.49	1449.36	**30.359**
SonyAIBO.	6.73	11.415	**0.02**
Symbols	8901.28	308.99	**2.068**
SyntheticControl	984.36	219.97	**1.165**
Trace	54128.53	275.375	**0.279**
TwoLeadECG	3.12	15.415	**0.032**

E. *Accuracy and Running Time*

We compare the accuracy of LCTS on the 28 datasets with the 10 baselines. The accuracies are showed in Table V. The best method per dataset is highlighted in bold.

Our method LCTS gives best accuracy on 14 datasets among the 28 datasets. Among the 10 baselines, only LTS shows a little better accuracy and it has best accuracy on 16 datasets. In order to evaluate these method better, we compare the average accuracy rank. The average rank of LCTS is 2.607, a litter worse than LST, but much better than other 9 baselines. LST's average rank is 1.964 and the third best method's average rank is 5.107. The disparity between LCTS and LST is far less than that of LCTS and the third best method. The results of 1-1 Wins show the similar classification performance. Therefore, when it refers to accuracy, our method LCTS is much better than other 9 baselines and only a little worse than LTS. But the running time of LCTS is far less than LTS as shown in the following paragraph.

Papers about shapelets often compare the running time of searching for one best shapelet. Since LCTS finds all shapelets at the same time, we compare the whole running time of LCTS with the running time of the F-Stat metric (which is the quickest metric for selecting shapelet by quality metric) and LTS for finding one best shapelet. It's obvious unfair for us, but the results still show that LCTS is much faster than

[1] https://github.com/yyawesome/LearningShapelets

TABLE V: Accuracies of the 10 Baselines on 28 Time Series Datasets.

Dateset	Baselines										LCTS
	Shapelet Tree	C4.5	1NN	NB	BN	RAF	ROF	SVM	FSH	LTS	
Adiac	0.299	0.243	0.253	0.281	0.251	0.304	0.307	0.238	0.558	0.542	**0.785**
Beef	0.500	0.600	0.833	0.733	**0.900**	0.600	0.700	0.867	0.517	0.800	0.767
Beetle/Fly	0.775	0.750	**1.000**	0.925	0.975	0.900	0.950	0.975	0.742	0.950	0.900
Bird/Chicken	0.850	0.900	0.975	0.875	0.950	0.950	0.925	0.950	0.843	**1.000**	**1.000**
ChlorineCocentration	0.588	0.565	0.569	0.460	0.571	0.576	0.635	0.562	0.578	**0.743**	0.636
Coffee	0.964	0.857	**1.000**	0.929	0.964	**1.000**	0.893	**1.000**	0.925	**1.000**	**1.000**
DiatomSizeReduction	0.722	0.752	0.935	0.788	0.902	0.804	0.830	0.922	0.869	0.951	**0.977**
DP_Little	0.654	0.659	0.728	0.735	0.729	0.730	0.747	**0.752**	0.566	0.727	0.705
DP_Middle	0.705	0.712	0.737	0.740	0.747	0.755	0.768	**0.796**	0.581	0.758	0.718
DP_Thumb	0.581	0.580	0.607	0.630	0.639	0.641	0.671	0.698	0.580	**0.740**	0.668
ECGFiveDays	0.775	0.962	0.984	0.964	0.995	0.933	0.986	0.990	0.996	**1.000**	**1.000**
FaceFour	0.841	0.761	**1.000**	0.977	**1.000**	0.875	0.989	0.977	0.909	**1.000**	**1.000**
GunPoint	0.893	0.907	0.980	0.920	0.993	0.960	0.987	**1.000**	0.932	**1.000**	**1.000**
ItalyPowerDemand	0.892	0.910	0.921	0.925	0.924	0.930	0.920	0.921	0.921	0.962	**0.971**
Lighting7	0.493	0.534	0.493	0.575	0.658	0.644	0.658	0.699	0.601	**0.877**	0.753
MedicalImages	0.488	0.449	0.457	0.174	0.282	0.508	0.515	0.525	0.608	**0.734**	0.712
MoteStrain	0.825	0.844	0.903	0.888	0.891	0.846	0.870	0.887	0.785	0.913	**0.927**
MP_Little	0.664	0.634	0.685	0.688	0.695	0.714	0.752	0.750	0.565	**0.758**	0.729
MP_Middel	0.710	0.733	0.709	0.720	0.711	0.752	0.747	0.769	0.605	**0.780**	0.743
Otoliths	0.672	0.656	0.719	0.688	0.641	0.656	0.594	0.641	0.560	**0.766**	0.703
PP_Little	0.596	0.574	0.672	0.692	0.701	0.666	0.698	**0.721**	0.549	0.710	0.707
PP_Middel	0.614	0.625	0.685	0.698	0.714	0.705	0.754	0.759	0.580	0.767	**0.772**
PP_Thumb	0.608	0.595	0.677	0.694	0.695	0.678	**0.728**	0.755	0.536	0.715	0.713
SonyAIBORobotSurface	0.845	0.845	0.840	0.790	0.897	0.852	0.890	0.867	0.698	0.952	**0.970**
Symbols	0.780	0.471	0.856	0.780	0.923	0.846	0.844	0.846	0.930	**0.959**	**0.959**
SyntheticControl	0.943	0.903	0.930	0.780	0.767	0.890	0.920	0.873	0.917	**1.000**	0.993
Trace	0.980	0.980	0.980	0.980	**1.000**	0.980	0.980	0.980	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.851	0.853	0.995	0.991	0.988	0.961	0.980	0.993	0.922	**1.000**	**1.000**
LCTS 1-1 Wins	28	28	21	25	22	23	21	17	27	6	-
LCTS 1-1 Draws	0	0	2	0	2	2	0	2	1	8	-
LCTS 1-1 Losses	0	0	5	3	4	3	7	9	0	14	-
Average Rank	8.643	9.000	5.643	7.107	5.107	6.036	5.214	4.143	8.143	1.857	2.607

them. The results are shown in Table IV. The results show that LCTS is 9895.95 times faster than F-stats and 61.28 times faster than LST in average on the 28 datasets.

F. Speed for Predicting

The running time for predicting a time series is proportional to the number of learned shapelets. Here we give discussion on the number of shapelets learned by LCTS. Table II gives the number of learned shapelets on the 28 datasets. We define $\gamma = \frac{Shapelets\quad Num.}{Length\quad of\quad Time\quad Series}$. The mean value of γ is 0.921. The standard deviation of γ is 1.104. Among the 28 datasets, there are 18 datasets on which the number of learned shapelets is less than the length of the time series and 10 datasets on which the number of learned shapelets exceeds the length of the time series. Overall, the number of shapelets learned by LCTS is of the same order of magnitude as the length of time series and it is within reasonable bounds.

VI. Conclusion and Future Work

In this paper, we propose a new perspective for learning time series shapelets. Instead of searching top shapelets from time series sub-sequences, we learn prototypes using a self-organizing incremental neural network (SOINN) and consider the prototypes as shapelet candidates. After learning prototypes, we propose a transform function to transform the time series data into shapelet feature space. The transformed value can unbiased reflect the similarity relationships between time series and prototypes. And then an alternative attribute selection method name L1-regularizer is used to select shapelets. Experimental results have verified the validity of the prototype learning method and the transform function, and show that our method provides remarkable improvement on accuracies and running time. Our future work will focus on reducing the number of learned shapelets while not decreasing the accuracy.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 61375064, 61373001 and 61321491), Foundation of Jiangsu NSF (Grant No. BK20131279).

References

[1] Hamilton J D. Time series analysis. Princeton: Princeton university press, 1994.

[2] Keogh E, Xi X, Wei L, et al. The UCR time series classification/clustering homepage. http://www. cs. ucr. edu/eamonn/time_series_data, 2006.

[3] Hills, J, Lines, J, Baranauskas, E, Mapp, J, and Bagnall. A Time Series Classification with Shapelets. Journal of Data Mining and Knowledge Discovery. http://www.uea.ac.uk/computing/machine-learning/shapelets.

[4] Ye L, Keogh E. Time series shapelets: a new primitive for data mining. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009: 947-956.

[5] Lines J, Davis L M, Hills J, et al. A shapelet transform for time series classification. Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012: 289-297.

[6] Grabocka J, Schilling N, Wistuba M, et al. Learning time-series shapelets. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014: 392-401.

[7] Rakthanmanon T, Keogh E. Fast shapelets: A scalable algorithm for discovering time series shapelets. Proceedings of the thirteenth SIAM conference on data mining (SDM). SIAM, 2013: 668-676.

[8] He Q, Dong Z, Zhuang F, et al. Fast time series classification based on infrequent shapelets. Machine Learning and Applications (ICMLA), 2012 11th International Conference on. IEEE, 2012, 1: 215-219.

[9] Mueen A, Keogh E, Young N. Logical-shapelets: an expressive primitive for time series classification. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011: 1154-1162.

[10] Xu Z B, Zhang H, Wang Y, et al. L1/2 regularization. Science China Information Sciences, 2010, 53(6): 1159-1169.

[11] K.-W. Chang, B. Deka, W. mei W. Hwu, and D. Roth. Efficient pattern-based time series classification on gpu. In M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu, editors, ICDM, pages 131140. IEEE Computer Society, 2012

[12] Cetin M S, Mueen A, Calhoun V D. Shapelet ensemble for multi-dimensional time series. Quebec: SIAM SDM, 2015.

[13] Wallach H M. Topic modeling: beyond bag-of-words. Proceedings of the 23rd international conference on Machine learning. ACM, 2006: 977-984.

[14] de Carvalho Soares R, da Silva I R, Guliato D. Spatial locality weighting of features using saliency map with a bag-of-visual-words approach. IEEE 24th International Conference on Tools with Artificial Intelligence. IEEE, 2012, 1: 1070-1075.

[15] Furao S, Ogura T, Hasegawa O. An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks, 2007, 20(8): 893-903.

[16] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers. Neural processing letters, 1999, 9(3): 293-300.

[17] Ng A Y. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. Proceedings of the twenty-first international conference on Machine learning. ACM, 2004: 78.