Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Efficient Learning of Timeseries Shapelets

Lu Hou James T. Kwok
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong

Jacek M. Zurada
Department of Electrical and Computer Engineering
University of Louisville, Louisville, KY, 40292, USA
Information Technology Institute
University of Social Science, Lodz 90-113, Poland

Abstract

In timeseries classification, shapelets are subsequences of timeseries with high discriminative power. Existing methods perform a combinatorial search for shapelet discovery. Even with speedup heuristics such as pruning, clustering, and dimensionality reduction, the search remains computationally expensive. In this paper, we take an entirely different approach and reformulate the shapelet discovery task as a numerical optimization problem. In particular, the shapelet positions are learned by combining the generalized eigenvector method and fused lasso regularizer to encourage a sparse and blocky solution. Extensive experimental results show that the proposed method is orders of magnitudes faster than the state-of-the-art shapelet-based methods, while achieving comparable or even better classification accuracy.

Introduction

Timeseries is the object of interest in many real-world applications such as finance, medical diagnosis, and weather forecasting (Hamilton 1994). In recent years, a new primitive called *shapelets* has attracted a lot of attention in timeseries classification (Ye and Keogh 2009; Rakthanmanon and Keogh 2013; Hills et al. 2014; Grabocka et al. 2014). Shapelets are subsequences of a timeseries with high discriminative power. The idea is that different classes of timeseries can often be distinguished by their local variations instead of the global structure. Besides, these local shapelets allow easier visualization and interpretation of the data.

Shapelet-based timeseries classification is first proposed in (Ye and Keogh 2009). The best discriminative shapelets are searched over all possible subsequences, and a decision tree is constructed based on the standard information gain criterion. Instead of relying only on decision trees, Hills et al. (2014) proposed to first transform the data by measuring distances between the original timeseries data and discovered shapelets. An off-the-shelf classifier (such as the support vector machine) is then used on the shapelet-transformed data. Empirically, this improves prediction accuracy while still maintaining the explanatory power of shapelets. Instead of restricting the shapelets to subsequences in the original timeseries, Grabocka et al. (2014) proposed to directly learn the shapelets jointly with the classifier. The resultant optimization problem is highly nonlinear, but the classification accuracy can be further improved.

Despite such promising performance, the search for shapelets is computationally expensive. In recent years, numerous solutions have been proposed to improve the efficiency. Rakthanmanon and Keogh (2013) pruned the space of shapelet candidates using the triangle inequality. However, given N length-Q timeseries, the search still takes $O(N^2Q^3)$ time. Instead of using the information gain as quality measure for shapelets, Hills et al. (2014) used other measures such as the Kruskal-Wallis statistic (Kruskal and others 1952) and F-statistic (Wright 1949). While these measures are easier to compute than the information gain, the speed improvement is minor. Chang et al. (2012) proposed to parallelize the distance computations of different shapelet segments using graphics processing units (GPUs). However, this requires significant modifications to the distance computation, and also considerable effort to deal with memory access issues and limited hardware capabilities of the GPU. Rakthanmanon and Keogh (2013) proposed to make the search process more efficient by projecting the timeseries to a reduced-dimensional space based on the SAX representation (Lin et al. 2007). However, the shapelets obtained are of the same length as SAX words, which reduces the prediction accuracy.

Very recently, Wistuba, Grabocka, and Schmidt-Thieme (2015) proposed an ultra-fast shapelet algorithm that randomly selects shapelets. This is based on the assumption that discriminative shapelets should appear frequently in the timeseries data, so random sampling will have a high chance of discovering them. However, since these shapelets are random, ensuring a high accuracy requires using a large number of shapelets, which in turn slows prediction. Moreover, random shapelets are more difficult to interpret. Later, they proposed an algorithm for scalable discovery of shapelets using online clustering and pruning (Grabocka, Wistuba, and Schmidt-Thieme 2015). A supervised shapelet selection procedure is used that tries to filter out only candidates that improve accuracy. However, empirically, the prediction accuracy is poor.

Existing methods perform a combinatorial search for shapelet discovery. As can be seen from above, even with speedup heuristics such as pruning, clustering, and dimen-

sionality reduction, the search remains computationally expensive. In this paper, we take an entirely different approach and reformulate the shapelet discovery task as a numerical optimization problem. To ensure that the obtained shapelets are discriminative, we consider using a state-of-the-art feature extraction technique called the generalized eigenvector method (Karampatziakis and Mineiro 2014). On the other hand, to ensure that the shapelets are local, we employ sparse modeling techniques that are now commonly used in various machine learning applications (Bach et al. 2012). Since a timeseries is a time-ordered list of observations, we preserve the order information by using the fused lasso regularizer (Tibshirani et al. 2005). The discriminative sparse blocks identified from the timeseries data are then used as shapelets. As for the optimization problem associated with the resultant model, it can be efficiently solved by the alternating direction method of multipliers (ADMM) (Boyd et al. 2011). Extensive empirical results on a large number of benchmark timeseries data sets show that the proposed method is orders of magnitudes faster than the state-of-the-art, while achieving comparable or even better classification accuracies.

Related Work

Generalized Eigenvector Method (GEM)

The GEM is a state-of-the-art supervised feature extraction technique (Karampatziakis and Mineiro 2014). While principal component analysis (PCA) is unsupervised and finds directions that maximize the projected data variance, GEM finds the direction v that maximizes the ratio of projected data variances between classes i and j:

$$v = \arg\max_v \frac{v^\top C_i v}{v^\top C_j v} \ : \ v^\top v = \text{constant}, \qquad (1)$$

where C_k is the sample covariance matrix of class k. Up to a scaling on v, (1) can be rewritten as:

$$v = \arg\min_v v^\top C_j v \ : \ v^\top C_i v = 1.$$

Intuitively, this direction is discriminative as the information in one class is maximally retained while the other is compressed. Moreover, compared with other well-known discriminative feature extraction methods such as Fisher LDA (Fisher 1936) and sliced inverse regression (Li 1991), GEM is advantageous in that it uses the covariance matrix instead of simply the conditional mean.

Fused Lasso

In recent years, sparse modeling has been commonly used in diverse areas such as computer vision, signal processing, and bioinformatics. It selects a relevant subset of features while simultaneously learning the model. The most popular sparse modeling algorithm is the lasso (Tibshiranit 1996), which uses the ℓ_1 penalty to encourage sparsity.

In many applications, the features in a sample are ordered. For example, observations in a timeseries are ordered by time. While the lasso ignores this order information, the fused lasso regularizer $\alpha_1 \sum_i |v_i - v_{i-1}| + \alpha_2 \|v\|_1$ (where α_1, α_2 are regularization parameters and $\|v\|_1$ is the ℓ_1 norm of v) encourages parameter estimates at successive features to be similar (Tibshirani et al. 2005). The first term $\sum_i |v_i - v_{i-1}|$ is often called the total-variation (TV) regularizer. The fused lasso regularizer can be written more compactly as $\alpha_1 \|Dv\|_1 + \alpha_2 \|v\|_1$, where D is a matrix such that $D_{i,i} = 1, D_{i,i+1} = -1$, and $D_{i,j} = 0$ otherwise. Because of the use of both the TV- and ℓ_1-regularizers, the fused lasso solution is simultaneously blocky and sparse (Rinaldo 2009).

Proposed Method

We first consider the simpler scenario of binary classification. Let $S_N = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be the training set of N timeseries samples, where each timeseries $\mathbf{x}_i \in \mathbb{R}^Q$ is of length Q, and $y_i \in \{\pm 1\}$ is its class label.

By adding a fused lasso regularizer on the GEM formulation, we can obtain sparse, blocky indicator vector for the shapelet location. The optimization problem becomes:

$$\min_v v^\top C_2 v + \alpha_1 \|Dv\|_1 + \alpha_2 \|v\|_1 \ : \ v^\top C_1 v = 1, \qquad (2)$$

where $C_k = \sum_{i=1}^N \mathbb{I}(y_i = k) \mathbf{x}_i \mathbf{x}_i^\top / \sum_{i=1}^N \mathbb{I}(y_i = k)$ is the covariance matrix of class k (here, $\mathbb{I}(\cdot)$ denotes the indicator function which returns 1 when the argument is satisfied, and 0 otherwise). Note that using only the $\|v\|_1$ regularizer produces solutions with weak block structures, while using only $\|Dv\|_1$ produces solutions that are blocky but not sparse.

In GEM, maximizing either $\frac{v^\top C_1 v}{v^\top C_2 v}$ or $\frac{v^\top C_2 v}{v^\top C_1 v}$ returns the most discriminative projection direction. Similarly, we can swap the roles of C_1, C_2 in (2), and use C_2 as the dominant class instead. This leads to:

$$\min_v v^\top C_1 v + \alpha_1 \|Dv\|_1 + \alpha_2 \|v\|_1 \ : \ v^\top C_2 v = 1. \qquad (3)$$

ADMM Solver

The optimization problems in (2) and (3) have the same form. In this section, we focus on how to efficiently solve (2) using the alternating direction method of multipliers (ADMM) (Boyd et al. 2011). The whole procedure, which will be called FLAG (Fused LAsso Generalized eigenvector method) in the sequel, is shown in Algorithm 1.

First, (2) can be rewritten as

$$\begin{aligned} \min_{v,y,z} \quad & v^\top C_2 v + \alpha_1 \|z\|_1 + \alpha_2 \|y\|_1 \\ \text{s.t.} \quad & Dv = z, v = y, v^\top C_1 v = 1. \end{aligned}$$

Setting aside the constraint $v^\top C_1 v = 1$, the augmented Lagrangian is:

$$\begin{aligned} \mathcal{L}(v, z, y, \lambda_1^t, \lambda_2^t) \quad = \quad & v^\top C_2 v + \alpha_1 \|z\|_1 + \alpha_2 \|y\|_1 \\ & + \lambda_1^{t\top} (Dv - z) + \lambda_2^{t\top} (v - y) \\ & + \frac{\rho_1}{2} \|Dv - z\|^2 + \frac{\rho_2}{2} \|v - y\|^2, \end{aligned}$$

where λ_1^t, λ_2^t are dual variables at the tth iteration, and ρ_1, ρ_2 are penalty parameters. At the $(t+1)$th iteration, the values of v, z and y are updated by minimizing $\mathcal{L}(v, z, y, \lambda_1^t, \lambda_2^t)$ w.r.t. them in an alternating manner.

Algorithm 1 FLAG.

Input: timeseries of length Q, parameters $\alpha_1, \alpha_2, \rho_1, \rho_2$, stopping threshold ϵ.
Output: shapelet indicator vector v.

1: **Initialize:** compute covariance matrices C_1 and C_2, $v^1 = \text{randn}(Q,1)$, $z^1 = 0.1 \times \text{randn}(Q,1)$, $y^1 = 0.1 \times \text{randn}(Q,1)$, $\mu_1^1 = \mu_2^1 = 0$.
2: **for** $t = 1, 2, \ldots T_{max}$ **do**
3: update v^{t+1} using (5);
4: $z^{t+1} \leftarrow S_{\alpha_1/\rho_1}(Dv^{t+1} + \mu_1^t)$;
5: $y^{t+1} \leftarrow S_{\alpha_2/\rho_2}(v^{t+1} + \mu_2^t)$;
6: $\mu_1^{t+1} \leftarrow \mu_1^t + (Dv^{t+1} - z^{t+1})$;
7: $\mu_2^{t+1} \leftarrow \mu_2^t + (v^{t+1} - y^{t+1})$;
8: **if** relative error $< \epsilon$ **then**
9: break;
10: **end if**
11: **end for**

Updating v Set $\mu_1^t = \frac{\lambda_1^t}{\rho_1}$, $\mu_2^t = \frac{\lambda_2^t}{\rho_2}$, the optimization subproblem for v is:

$$v^{t+1} = \underset{v: v^\top C_1 v = 1}{\arg\min}\ v^\top C_2 v + \frac{\rho_1}{2}\|Dv - z^t + \mu_1^t\|^2 + \frac{\rho_2}{2}\|v - y^t + \mu_2^t\|^2.$$

Let $R^\top R$ be the Cholesky decomposition of C_1, $U\Sigma U^\top$ be the SVD of

$$R^{-\top}(C_2 + \frac{\rho_1}{2}D^\top D + \frac{\rho_2}{2}I)R^{-1}, \quad (4)$$

and $\Sigma = \text{diag}(\sigma_i)$. It can be shown that

$$v^{t+1} = \frac{1}{2}R^{-1}\left(R^{-\top}(C_2 + \frac{\rho_1}{2}D^\top D + \frac{\rho_2}{2}I)R^{-1} - \gamma I\right)^{-1} R^{-\top}(\rho_1 D^\top(z^t - \mu_1^t) + \rho_2(y^t - \mu_2^t)), \quad (5)$$

where $d = [d_1, d_2, \ldots d_Q]^\top = \frac{1}{2}U^\top(\rho_1 D^\top(z^t - \mu_1^t) + \rho_2(y^t - \mu_2^t))$, and γ is the smallest solution of

$$\sum_{j=1}^{Q} \frac{d_j^2}{(\sigma_j - \gamma)^2} = 1. \quad (6)$$

In the experiments, γ can be found efficiently using the Newton's method.

Updating z **and** y The update for z is

$$z^{t+1} = \arg\min_z \alpha_1\|z\|_1 + \frac{\rho_1}{2}\|Dv^{t+1} - z + \mu_1^t\|^2 = S_{\alpha_1/\rho_1}(Dv^{t+1} + \mu_1^t),$$

where $S_k(a) = \text{sign}(a) \cdot \max(|a| - k, 0)$ is the soft-thresholding operator (Efron et al. 2004). Similarly, the update of y is $y^{t+1} = S_{\alpha_2/\rho_2}(v^{t+1} + \mu_2^t)$.

Updating μ_1 **and** μ_2 Updates for the scaled dual variables are given by:

$$\mu_1^{t+1} = \mu_1^t + (Dv^{t+1} - z^{t+1}),$$
$$\mu_2^{t+1} = \mu_2^t + (v^{t+1} - y^{t+1}).$$

From Shapelet Indicator to Shapelets

When class 1 is used as the dominant class, let there are B nonzero blocks in the obtained shapelet indicator vector v, with the tth block starting at time s_t and ending at e_t, i.e.,

$$v = [0, \ldots, 0, v_{s_1}, \ldots, v_{e_1}, 0, \ldots, 0, v_{s_B} \ldots, v_{e_B}, 0, \ldots, 0].$$

Denote the set of N_1 training samples from class 1 be $\{\mathbf{x}_1^{(1)}, \mathbf{x}_2^{(1)}, \ldots \mathbf{x}_{N_1}^{(1)}\}$. The set of shapelets constructed from v is

$$S_1 = \{[\mathbf{x}_k^{(1)}]_{s_t:e_t} \ :\ k = 1, 2, \ldots N_1, t = 1, 2, \ldots B\},$$

where $[\mathbf{x}_k^{(1)}]_{s_t:e_t}$ is the subsequence of $\mathbf{x}_k^{(1)}$ from s_t to e_t. This leads to a total of $|S_1| = N_1 B$ shapelets. The procedure is analogous when class 2 is used as the dominant class.

As in (Hills et al. 2014), we then transform the training data $X \in \mathbb{R}^{N \times Q}$ to a new representation $M \in \mathbb{R}^{N \times (|S_1| + |S_2|)}$ by measuring the Euclidean distance to the closest shapelet of the two classes.

Extensions

For multiclass classification involving K classes, we use the one-vs-rest strategy. Specifically, when class i is used as the dominant class, a shapelet indicator vector is obtained by solving the following optimization problem:

$$\min_v v^\top C_{(\neq i)} v + \alpha_1\|Dv\|_1 + \alpha_2\|v\|_1 \ :\ v^\top C_i v = 1,$$

where $C_{(\neq i)} = \frac{1}{K-1}\sum_{j \neq i} C_j$. Each class is treated as a dominant class to generate the shapelet indicators. The final set of shapelets is the union of all resultant shapelets.

Instead of learning just one single shapelet indicator, one can also learn k shapelet indicators $V = [v_1, v_2, \ldots v_k]$. It can be shown that the V update in (5) is then modified to

$$v_i^{t+1} = \frac{1}{2}R^{-1}\left(R^{-\top}(C_2 + \frac{\rho_1}{2}D^\top D + \frac{\rho_2}{2}I)R^{-1} - \gamma_i I\right)^{-1} R^{-\top}(\rho_1 D^\top(z^t - \mu_1^t) + \rho_2(y^t - \mu_2^t)),$$

where γ_i is the ith smallest solution of (6).

Time Complexity

For the initialization (step 1 in Algorithm 1), the covariance matrices can be computed in $O(NQ^2)$ time. For learning of the shapelet indicator, the v update (step 3) needs the SVD of an $Q \times Q$ matrix in (4). This takes $O(Q^3)$ time but only needs to be computed once. Obtaining γ using Newton's method takes $O(Q^2)$ time. Updates for z, y and the dual variables (steps 4-7) take $O(Q^2)$ time. Hence, the total time complexity of finding one shapelet indicator is $O(Q^3 + TQ^2)$, where T is the number of ADMM iterations (typically, $T < 150$). For K classes, the time is $O(K(Q^3 + TQ^2))$. Often, $NQ^2 \ll KQ^3$ and $K < 5$. Thus, the total time complexity is $O(Q^3 + TQ^2)$.

Experiments

Experiments are performed on the commonly used UCR[1] and UEA[2] data sets (Table 1) (Hills et al. 2014). As is

[1] http://www.cs.ucr.edu/~eamonn/time_series_data/.
[2] https://www.uea.ac.uk/computing/machine-learning/shapelets/shapelet-data.

common in the shapelet literature (Grabocka et al. 2014; Grabocka, Wistuba, and Schmidt-Thieme 2015; Hills et al. 2014; Rakthanmanon and Keogh 2013), we use the default training/testing splits.

Table 1: Data sets used in the experiments.

	#train	#test	seq length	#classes
Adiac	390	391	176	37
Beef	30	30	470	5
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	350	4
Gun_Point	50	150	150	2
ItalyPower.	67	1029	24	2
Lightning7	70	73	319	7
MedicalImages	381	760	99	10
MoteStrain	20	1252	84	2
MP_Little	400	645	250	3
MP_Middle	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
Sony.	20	601	70	2
Symbols	25	995	398	6
SyntheticC.	300	300	60	6
Trace	100	100	275	4
TwoLeadECG	23	1139	82	2

The following shapelet-based methods are compared:

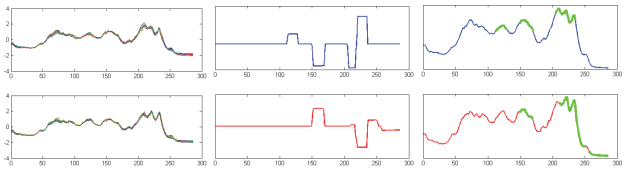
1. Standard shapelet-based classifiers using three shapelet quality measures : information gain (`IG`) (Ye and Keogh 2009), Kruskal-Wallis statistic (`KW`), and F-statistic (`FS`) (Hills et al. 2014);
2. `FSH`: The fast shapelet algorithm (Rakthanmanon and Keogh 2013), which searches for potential shapelets by first transforming the timeseries data to a discrete and low-dimensional representation;
3. `SD`: The scalable discovery algorithm in (Grabocka, Wistuba, and Schmidt-Thieme 2015), which avoids measuring prediction accuracies of similar shapelets via online clustering and pruning;
4. `LTS`: Learning timeseries shapelets (Grabocka et al. 2014), which learns the shapelets and decision function jointly using gradient descent;
5. `UFS`: The ultra-fast shapelet algorithm (Wistuba, Grabocka, and Schmidt-Thieme 2015), which uses random shapelets;
6. `IGSVM`: Shapelet-transform algorithm (Hills et al. 2014), which uses the linear SVM as classifier and information gain as shapelet quality measure;
7. The proposed `FLAG`, which also uses a linear SVM on the shapelet-transformed data.

The codes (except `UFS`) are provided by authors of the various baseline methods, and the parameters are set as recommended. `FSH` is implemented in C++; `IG`, `KW`, `FS`, `IGSVM` and `LTS` are in Java; `UFS` and ours are in Matlab. Experiments are performed on a desktop with Intel i7 CPU and 16GB memory.

Accuracy and Time

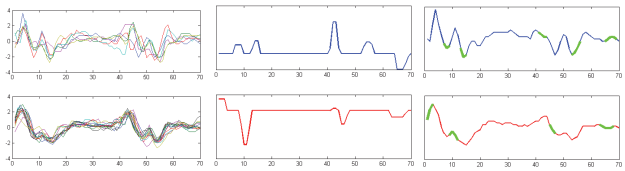
Example shapelets learned on two of the data sets, *coffee* and *SonyAIBORobotSurface*, are shown in Figures 1 and 2, respectively. Table 2 shows the prediction accuracies. `LTS` is the best, as it is most flexible in the choice of shapelets, while others can only choose existing subsequences from the timeseries data set. `FLAG` is the second best, while the standard shapelet-based classifiers have the lowest accuracies.

On the other hand, `LTS` is much slower than `FLAG`. Table 3 shows the total time for shapelet discovery and model training. As can be seen, `FLAG` is about three orders of magnitude faster than `IG`, `IGSVM`, and `LTS`. Compared to `FSH` which uses the SAX representation for dimension reduction, `FLAG` is still often an order of magnitude faster. `SD` is very fast and `UFS` is also relatively fast. However, they have much lower accuracies as we have seen from Table 2.



(a) Timeseries. (b) Indicators. (c) Shapelets.

Figure 1: (a) Example timeseries from the two classes of *coffee*; (b) Learned shapelet indicator vectors; (c) Example shapelets.



(a) Timeseries. (b) Indicators. (c) Shapelets.

Figure 2: (a) Example timeseries from the two classes of *SonyAIBORobotSurface*; (b) Learned shapelet indicator vectors; (c) Example shapelets.

Scalability

In this experiment, we use the largest timeseries data set *StarlightCurves* from UCR. It contains 9236 starlight curves from 3 classes, each of length 1024. We use the first 6000 curves for training, and the rest for testing. The shapelet discovery time is mostly affected by (i) the number of training curves (N); and (ii) the length of each curve (Q).

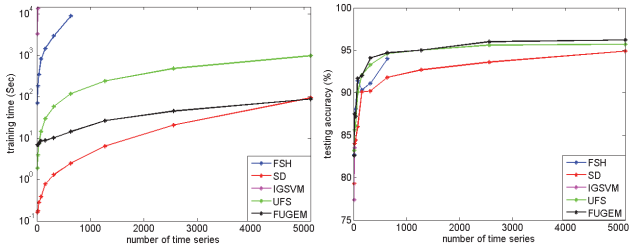
Table 2: Testing accuracies (%) on the data sets. Number in brackets is the method's ranking(1 is the best). The best method for each dataset is highlighted in bold. Note that IG, KW and FS sometimes cannot finish in 24 hours. The corresponding accuracies are denoted "-", and their rankings are taken as 8.

data set	IG	KW	FS	FSH	SD	IGSVM	LTS	UFS	FLAG
Adiac	29.9 (6)	26.6 (7)	15.6 (9)	57.5 (3)	52.2 (4)	23.5 (8)	51.9 (5)	69.8 (2)	**75.2 (1)**
Beef	50.0 (6)	33.3 (9)	56.7 (5)	50.0 (6)	50.0 (6)	**90.0 (1)**	76.7 (3)	66.7 (4)	83.3 (2)
Chlorine.	58.8 (5)	52.0 (9)	53.5 (8)	58.8 (5)	59.6 (4)	57.1 (7)	73.0 (3)	73.8 (2)	**76.0 (1)**
Coffee	96.4 (5)	85.7 (9)	**100.0 (1)**	92.9 (8)	96.4 (5)	**100.0 (1)**	**100.0 (1)**	96.4 (5)	**100.0 (1)**
Diatom.	76.5 (7)	62.1 (9)	76.5 (7)	87.3 (5)	86.6 (6)	93.1 (4)	94.2 (3)	95.8 (2)	**96.4 (1)**
DP_Little	- (8)	- (8)	- (8)	60.6 (5)	55.7 (6)	66.6 (4)	**73.4 (1)**	67.4 (3)	68.3 (2)
DP_Middle	- (8)	- (8)	- (8)	58.8 (5)	55.3 (6)	69.5 (3)	**74.1 (1)**	66.5 (4)	71.3 (2)
DP_Thumb	- (8)	- (8)	- (8)	63.4 (5)	54.4 (6)	69.6 (3)	**75.2 (1)**	68.5 (4)	70.5 (2)
ECGFiveDays	77.5 (9)	87.2 (8)	99.0 (4)	99.8 (3)	91.5 (7)	99.0 (4)	**100.0 (1)**	**100.0 (1)**	92.0 (6)
FaceFour	84.0 (6)	44.3 (8)	75.0 (9)	92.0 (4)	83.0 (7)	**97.7 (1)**	94.3 (2)	93.2 (3)	90.9 (5)
Gun_Point	89.3 (9)	94.0 (6)	95.3 (5)	94.0 (6)	93.1 (8)	**100.0 (1)**	99.6 (2)	98.7 (3)	96.7 (4)
ItalyPower.	89.2 (8)	91.0 (6)	93.1 (5)	91.0 (6)	88.0 (9)	93.7 (4)	**95.8 (1)**	94.0 (3)	94.6 (2)
Lightning7	49.3 (7)	48.0 (8)	41.1 (9)	65.2 (4)	65.2 (4)	63.0 (6)	**79.0 (1)**	68.5 (3)	76.7 (2)
MedicalImages	48.8 (8)	47.1 (9)	50.8 (7)	64.7 (5)	66.0 (4)	52.2 (6)	71.3 (2)	71.1 (3)	**71.4 (1)**
MoteStrain	82.5 (8)	84.0 (5)	84.0 (5)	83.8 (7)	78.3 (9)	88.7 (3)	**90.0 (1)**	87.2 (4)	88.8 (2)
MP_Little	- (8)	- (8)	- (8)	56.9 (6)	62.7 (5)	70.7 (3)	**74.3 (1)**	71.7 (2)	69.3 (4)
MP_Middle	- (8)	- (8)	- (8)	60.3 (6)	64.5 (5)	76.9 (2)	**77.5 (1)**	74.8 (4)	75.0 (3)
Otoliths	67.2 (1)	60.9 (6)	57.8 (8)	60.9 (6)	64.1 (2)	64.1 (2)	59.4 (5)	57.8 (8)	64.1 (2)
PP_Little	- (8)	- (8)	- (8)	57.6 (5)	55.8 (6)	**72.1 (1)**	71.0 (2)	66.8 (4)	67.1 (3)
PP_Middle	- (8)	- (8)	- (8)	61.6 (5)	60.5 (6)	**75.9 (1)**	74.9 (3)	75.4 (2)	73.8 (4)
PP_Thumb	- (8)	- (8)	- (8)	55.8 (6)	61.8 (5)	**75.5 (1)**	70.5 (2)	67.2 (4)	67.4 (3)
Sony.	85.7 (5)	72.7 (8)	**95.3 (1)**	68.6 (9)	85.0 (6)	92.7 (3)	91.0 (4)	79.0 (7)	92.9 (2)
Symbols	78.4 (8)	55.7 (9)	80.1 (7)	92.4 (2)	86.5 (5)	84.6 (6)	**94.5 (1)**	88.8 (3)	87.5 (4)
SyntheticC.	94.3 (7)	90.0 (8)	95.7 (5)	94.7 (6)	98.3 (3)	87.3 (9)	97.3 (4)	**99.7 (1)**	**99.7 (1)**
Trace	98.0 (5)	94.0 (9)	**100.0 (1)**	**100.0 (1)**	96.0 (7)	98.0 (6)	**100.0 (1)**	96.0 (7)	99.0 (4)
TwoLeadECG	85.1 (7)	76.4 (9)	97.0 (4)	92.5 (5)	86.7 (6)	**100.0 (1)**	**100.0 (1)**	83.6 (8)	99.0 (3)
average rank	6.9	7.7	6.3	5.2	5.9	3.5	2.0	3.7	2.6

Table 3: Total time (in seconds) for shapelet discovery and model training.

	IG	KW	FS	FSH	SD	IGSVM	LTS	UFS	FLAG
Adiac	11684.6 (8)	4794.2 (6)	5377.9 (7)	105.2 (4)	**0.3 (1)**	2510.3 (5)	61977.0 (9)	16.1 (3)	5.9 (2)
Beef	1055.6 (6)	1085.2 (7)	1292.5 (8)	77.3 (4)	**0.1 (1)**	976.4 (5)	6957.4 (9)	2.3 (2)	2.6 (3)
Cholorine.	51621.6 (9)	17008.0 (8)	16146.5 (7)	181.9 (4)	**0.2 (1)**	6253.0 (5)	9347.4 (6)	28.6 (3)	0.9 (2)
Coffee	130.6 (9)	127.3 (7)	127.8 (8)	6.3 (4)	**0.1 (1)**	93.0 (6)	65.3 (5)	0.3 (2)	0.7 (3)
Diatom.	43.2 (8)	41.5 (6)	42.5 (7)	7.0 (4)	**0.1 (1)**	42.9 (5)	1960.1 (9)	2.3 (3)	1.8 (2)
DP_Little	- (8)	- (8)	- (8)	401.4 (4)	**0.9 (1)**	30586.7 (6)	1608.1 (5)	4.7 (3)	**0.9 (1)**
DP_Middle	- (8)	- (8)	- (8)	456.6 (4)	**0.7 (1)**	36879.9 (6)	12528.7 (5)	4.5 (3)	1.1 (2)
DP_Thumb	- (8)	- (8)	- (8)	392.3 (4)	**0.7 (1)**	46979.9 (6)	2073.8 (5)	4.6 (3)	1.2 (2)
ECGFiveDays	76.2 (9)	73.7 (7)	74.3 (8)	1.4 (4)	**0.1 (1)**	48.2 (6)	17.1 (5)	0.3 (3)	0.2 (2)
FaceFour	3402.0 (9)	3372.6 (8)	3365.0 (7)	28.3 (4)	**0.1 (1)**	1364.6 (5)	2485.7 (6)	0.5 (2)	0.8 (3)
Gun_Point	291.1 (8)	280.5 (7)	372.2 (9)	2.6 (4)	**0.1 (1)**	186.9 (5)	276.7 (6)	1.5 (3)	**0.1 (1)**
ItalyPower.	1.9 (8)	1.1 (5)	1.1 (5)	0.2 (3)	**0.1 (1)**	1.1 (5)	7.3 (9)	0.4 (4)	**0.1 (1)**
Lightning7	14244.6 (8)	14225.9 (7)	14830.5 (9)	98.5 (4)	**0.4 (1)**	6095.2 (6)	1132.8 (5)	3.3 (3)	1.2 (2)
MedicalImages	14490.8 (9)	8749.4 (8)	8710.5 (7)	58.2 (4)	**0.5 (1)**	5155.7 (6)	4530.2 (5)	3.6 (3)	0.7 (2)
MoteStrain	4.7 (8)	4.3 (6)	4.3 (6)	0.5 (3)	**0.1 (1)**	2.8 (5)	57.7 (9)	0.7 (4)	**0.1 (1)**
MP_Little	- (8)	- (8)	- (8)	460.6 (4)	**1.0 (1)**	33607.9 (6)	10779.6 (5)	4.5 (3)	**1.0 (1)**
MP_Middle	- (8)	- (8)	- (8)	421.8 (4)	**0.8 (1)**	49508.5 (6)	1448.8 (5)	4.5 (3)	0.9 (2)
Otoliths	44278.6 (8)	45028.5 (9)	44093.2 (7)	54.7 (4)	**0.2 (1)**	21157.3 (6)	1715.4 (5)	14.7(3)	2.9 (2)
PP_Little	- (8)	- (8)	- (8)	393.7 (4)	**0.6 (1)**	30883.6 (6)	9899.0 (5)	4.5 (3)	1.1 (2)
PP_Middle	- (8)	- (8)	- (8)	403.7 (4)	**1.0 (1)**	23766.1 (6)	6819.0 (5)	4.5 (3)	1.2 (2)
PP_Thumb	- (8)	- (8)	- (8)	416.9 (4)	**1.2 (1)**	35182.9 (6)	13675.4 (5)	4.6 (3)	**1.2 (1)**
Sony.	3.9 (8)	3.4 (6)	3.4 (6)	0.4 (3)	**0.1 (1)**	2.5 (5)	20.0 (9)	0.4 (3)	**0.1 (1)**
Symbols	6893.6 (8)	6878.9 (7)	7509.2 (9)	27.7 (4)	**0.1 (1)**	2619.1 (6)	277.6 (5)	0.9 (2)	1.7 (3)
syntheticC.	1663.8 (8)	938.6 (7)	918.0 (6)	14.4 (4)	**0.1 (1)**	5268.6 (9)	467.5 (5)	5.8 (3)	0.4 (2)
Trace	43284.2 (8)	43583.9 (9)	42791.4 (7)	50.0 (4)	**0.1 (1)**	18141.4 (6)	2637.3 (5)	4.0 (3)	1.2 (2)
TwoLeadECG	2.4 (8)	2.1 (6)	2.1 (6)	0.3 (2)	**0.1 (1)**	1.3 (5)	35.3 (9)	0.8 (4)	0.3 (2)
average rank	8.1	7.3	7.4	3.8	1.0	5.8	6.0	2.9	2.0

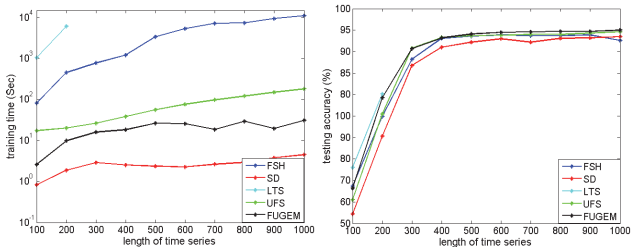
We first fix Q to 1024, and vary N from 10 to 5120. Note that IG, KW, and FS are too slow to be run, while the LTS code has large memory requirement and also cannot be run on our machine. The total time for shapelet discovery and model training of the various methods are shown in Figure 3(a). Same as the trends in Table 3, IGSVM and FSH are slow, while UFS, FLAG and SD are more scalable. Though FLAG is slower than SD for small N, it becomes faster than SD when N is large. This is because FLAG operates on the covariance matrix instead of the original timeseries. Computing the covariance matrix is cheap relative to the ADMM algorithm, especially when N is large. Moreover, even though SD is fast for small N, its testing accuracy is much inferior to the other methods (Figure 3(b)).



(a) time. (b) testing accuracy.

Figure 3: Varying the number of timeseries (N) on the *StarlightCurves* data set.

Next, we fix N to 1000, and vary Q from 100 to 1000. Note that IGSVM is very slow, and is thus dropped from the comparison. On the other hand, LTS can now be run for $Q = 100$ and 200. Results are shown in Figure 4. Again, in terms of speed, FLAG clearly outperforms LTS, FSH and UFS (Figure 4(a)). FLAG scales similarly as SD but is more accurate (Figure 4(b)).

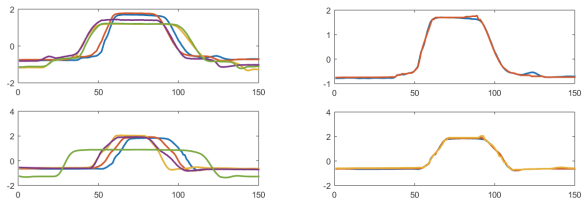


(a) time. (b) testing accuracy.

Figure 4: Varying the length of timeseries (Q) on the *StarlightCurves* data set.

Timeseries with Significant Phase Shifts

In some data sets, the timeseries may have significant phase shifts. An example is shown in Figure 5(a). To alleviate this problem, improved performance can be obtained by using dynamic time warping (DTW) (Keogh and Ratanamahatana 2005) to pre-process and align the timeseries (Figure 5(b)). Warping N timeseries takes $O(NQ^2)$ time, which is inexpensive compared to the ADMM iterations. Moreover, DTW only needs to be performed once and can be accelerated (Al-Naymat, Chawla, and Taheri 2009; Salvador and Chan 2004).



(a) Original. (b) After DTW alignment.

Figure 5: The two classes in the *Gun_Point* data set, before and after DTW alignment.

In this section, experiments are performed on several data sets in Table 2 with significant phase shifts. Results are shown in Table 4. As can be seen, the testing accuracies are all improved with DTW.

Table 4: Testing accuracy improvement using DTW.

	w/o DTW	w/ DTW
Gun_Point	96.7%	99.3%
syntheticC.	99.7%	100.0%
Trace	99.0%	100.0%
DP_Little	68.3%	71.9%
DP_Middle	71.3%	74.4%
DP_Thumb	70.5%	71.2%
MP_Little	69.3%	72.2%
MP_Middle	75.0%	77.5%
PP_Little	67.1%	72.7%
PP_Middle	73.8%	74.6%
PP_Thumb	67.4%	69.6%

Conclusion

In this paper, we proposed the learning of shapelets as a numerical optimization problem instead of using combinatorial search. We combined the state-of-the-art feature extraction technique of generalized eigenvector with the fused lasso regularizer. This encourages the formation of a discriminative, sparse, and blocky shapelet location indicator, which can then be easily turned into shapelets. The resultant optimization problem can be solved efficiently by ADMM. Extensive experiments are performed on a number of benchmark data sets. Results show that the proposed method is orders of magnitudes faster than existing shapelet-based methods, while achieving comparable or even better classification accuracies.

Acknowledgments

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614513).

References

Al-Naymat, G.; Chawla, S.; and Taheri, J. 2009. SparseDTW: A novel approach to speed up dynamic time warping. In *Proceedings of the 8th Australasian Data Mining Conference*, 117–127.

Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning* 4(1):1–106.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3(1):1–122.

Chang, K.-W.; Deka, B.; Hwu, W.-M. W.; and Roth, D. 2012. Efficient pattern-based time series classification on gpu. In *Proceedings of the 12th International Conference on Data Mining*, 131–140.

Efron, B.; Hastie, T.; Johnstone, I.; Tibshirani, R.; et al. 2004. Least angle regression. *Annals of Statistics* 32(2):407–499.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7(2):179–188.

Grabocka, J.; Schilling, N.; Wistuba, M.; and Schmidt-Thieme, L. 2014. Learning time-series shapelets. In *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining*, 392–401.

Grabocka, J.; Wistuba, M.; and Schmidt-Thieme, L. 2015. Scalable discovery of time-series shapelets. Technical Report arXiv:1503.03238.

Hamilton, J. D. 1994. *Time Series Analysis*. Princeton University Press.

Hills, J.; Lines, J.; Baranauskas, E.; Mapp, J.; and Bagnall, A. 2014. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery* 28(4):851–881.

Karampatziakis, N., and Mineiro, P. 2014. Discriminative features via generalized eigenvectors. In *Proceedings of the 31st International Conference on Machine Learning*, 494–502.

Keogh, E., and Ratanamahatana, C. A. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7(3):358–386.

Keogh, E.; Chakrabarti, K.; Pazzani, M.; and Mehrotra, S. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record* 30(2):151–162.

Kruskal, W. H., et al. 1952. A nonparametric test for the several sample problem. *Annals of Mathematical Statistics* 23(4):525–540.

Li, K.-C. 1991. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association* 86(414):316–327.

Lin, J.; Keogh, E.; Wei, L.; and Lonardi, S. 2007. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15(2):107–144.

Mueen, A.; Keogh, E.; and Young, N. 2011. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1154–1162.

Rakthanmanon, T., and Keogh, E. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 13th SIAM International Conference on Data Mining*, 668–676.

Rinaldo, A. 2009. Properties and refinements of the fused lasso. *Annals of Statistics* 37(5B):2922–2952.

Salvador, S., and Chan, P. 2004. FastDTW: Toward accurate dynamic time warping in linear time and space. In *KDD Workshop on Mining Temporal and Sequential Data*.

Tibshirani, R.; Saunders, M.; Rosset, S.; Zhu, J.; and Knight, K. 2005. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B* 67(1):91–108.

Tibshiranit, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B* 58(1):267–288.

Wistuba, M.; Grabocka, J.; and Schmidt-Thieme, L. 2015. Ultra-fast shapelets for time series classification. Technical Report arXiv:1503.05018.

Wright, S. 1949. The genetical structure of populations. *Annals of Eugenics* 15(1):323–354.

Ye, L., and Keogh, E. 2009. Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th Internatioanl Conference on Knowledge Discovery and Data Mining*, 947–956.