Learning DTW-Shapelets for Time-Series Classification

Mit Shah[1], Josif Grabocka[1], Nicolas Schilling[1], Martin Wistuba[1], Lars Schmidt-Thieme[1]
[1]Information Systems and Machine Learning Lab
University of Hildesheim
Universitätsplatz 1, Hildesheim 31141, Germany
{ mit, josif, schilling, wistuba, schmidt-thieme }@ismll.uni-hildesheim.de

ABSTRACT

Shapelets are discriminative patterns in time series, that best predict the target variable when their distances to the respective time series are used as features for a classifier. Since the shapelet is simply any time series of some length less than or equal to the length of the shortest time series in our data set, there is an enormous amount of possible shapelets present in the data. Initially, shapelets were found by extracting numerous candidates and evaluating them for their prediction quality. Then, Grabocka et al. [2] proposed a novel approach of learning time series shapelets called LTS. A new mathematical formalization of the task via a classification objective function was proposed and a tailored stochastic gradient learning was applied. It enabled learning near-to-optimal shapelets without the overhead of trying out lots of candidates. The Euclidean distance measure was used as distance metric in the proposed approach. As a limitation, it is not able to learn a single shapelet, that can be representative of different subsequences of time series, which are just warped along time axis. To consider these cases, we propose to use Dynamic Time Warping (DTW) as a distance measure in the framework of LTS. The proposed approach was evaluated on 11 real world data sets from the UCR repository and a synthetic data set created by ourselves. The experimental results show that the proposed approach outperforms the existing methods on these data sets.

1. INTRODUCTION

Applicability in broad range of real life domains makes time series classification an interesting topic for the research community. Time-series data often exhibits inter-class differences in terms of small sub-sequences rather than the full series structure [15]. A recently introduced concept, named *shapelet*, represents a maximally discriminative subsequence of time series data. Stated more directly, shapelets identify short discriminative series segments [15, 9]. Apart from their high prediction accuracy, shapelets also offer interpretable features to domain experts. Out of these reasons, discovering shapelets has been a hot topic in the time-series domain during the last five years [15, 9, 8, 16, 6, 11, 7].

Previously, state-of-the-art methods discovered shapelets by trying a pool of candidate sub-sequences from all possible series segments [15, 8] and then sorting the top performing segments according to their prediction qualities. Minimal distances between series and shapelets represent shapelet-transformed [8] classification features for a series of segregation metrics, such as information gain [15, 9], F-Stat [6] or Kruskall-Wallis [7]. The brute-force candidates search approach, which is simply based on an exhaustive search of candidates suffers from a high runtime complexity, therefore several speed-up techniques have aimed at reducing the discovery time of shapelets [9, 11, 1]. In terms of classification performance, the shapelet-transformation method constructs qualitative predictors for standard classifiers and has shown improvements with respect to prediction accuracy [8, 6].

Grabocka et al. [2] proposed an approach that learns these shapelets, by optimizing a classification loss as objective function, which is known as the LTS approach. They learned shapelets whose minimal distances to the time series instances can be used as features for a logistic regression model. The distance function they employed was the Euclidean distance. As a result, LTS is not able to learn a single shapelet, that can be representative of different subsequences of time series, which are simply just warped along the time axis. To overcome this issue, we propose to use Dynamic Time Warping (DTW) as a distance measure and call the resulting approach LTSD (**L**earning **T**ime-Series **S**hapelets using **D**TW). Additionally, DTW itself is used as a complete method for time series classification, where a test instance is assigned to the class of the nearest training instance using DTW as the distance measure.

We evaluate our proposed approach against LTS and DTW on 11 real world data sets and a synthetic data set created by ourselves. It was designed such that the class of a time series just depends on the sequence of the events and not on the temporal distance between these events. On the one hand, LTS was not performing well for the synthetic data set, whereas it's accuracy was up to the mark for the real world data sets. On the other hand, accuracies obtained by DTW were not good for the real world data sets, whereas it was up to the mark for the synthetic data set. For both the synthetic and the real-world data set, our approach LTSD shows good performance, meaning that it makes use of the best out of both the LTS and DTW approaches and works

CODS '16, March 13-16, 2016, Pune, India

DOI: http://dx.doi.org/10.1145/2888451.2888456

well in both the scenarios.

2. RELATED WORK

Shapelets were first proposed by [15] as time series segments that maximally predict the target variable. All possible segments were considered as potential candidates, while the minimum distances of a candidate to all training series were used as a predictor feature for ranking the information gain accuracy of that candidate on the target variable. Other quality metrics have been proposed for evaluating the prediction accuracy of a shapelet candidate such as F-Stats [8], Kruskall-Wallis or Mood's median [6]. In addition, the minimum distance of a set of shapelets to time series can be perceived as a data transformation [8] and standard classifiers have achieved high accuracy over the shapelet-transformed representation [6].

Due to the high number of candidates, the runtime of brute-force shapelet discovery is not feasible. Therefore, a series of speed-up techniques such as early abandoning of distance computations and entropy pruning of the information gain metric have been proposed [15]. Other speed-ups rely on the reuse of computations and pruning of the search space [9], as well as exploiting projections on the SAX representation [11]. Alternatively, the training time has been reduced by elaborating the usage of infrequent shapelet candidates [5]. Moreover, hardware-based optimization have assisted the discovery of shapelets using GPUs [1]. Shapelets have been applied in a series of real-life applications. Unsupervised shapelets have also been utilized for clustering time series [16]. Shapelets have been found useful for identifying humans through their gait data [12]. Gesture recognition is another application domain that has benefited from the discovery of shapelets [3, 4]. In the domain of medical and health informatics, interpretable shapelets have been used to enable efficient early classification of time series [14, 13].

Grabocka et al. [2] proposed learning (and not searching) time series shapelets using the Euclidean distance as a distance measure, which we improve by using Dynamic Time Warping (DTW) as a distance measure.

3. PROPOSED METHOD

For calculating the distance between two time series, DTW is the most commonly used distance measure. Unlike the Euclidean distance, it can create a perfect mapping between the points in two time series, if they are warped along the time axis. As a simple example, consider Figure 1 which shows two time series consisting of a common subsequence that appears at differing time-stamps. It is clear that Euclidean distance always maps the points in two time series according to their time-stamps, which is not the case for DTW distance, hence the DTW distance is much smaller. With this in mind, we propose a method to learn the shapelets using DTW as a distance measure for the time series classification task.

3.1 Background

Before we present the model for learning DTW-Shapelets, let us define time series, shapelet and DTW distance.

1. **Time Series**: It is an ordered set of Q real-valued variables. We denote it by T. Data points $T_1, \ldots, T_Q$ are typically arranged by temporal order, spaced at equal time intervals. For a particular time series data set, let us denote the number of instances of time series in it by I.

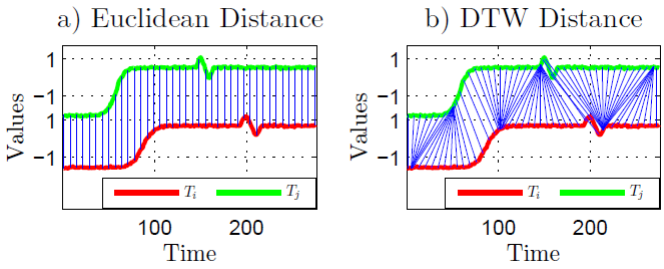

Figure 1: Illustrating DTW

		5	2	6	7	9	11
	0	∞	∞	∞	∞	∞	∞
1	∞	16	17	42	78	142	242
2	∞	25	16	32	57	106	187
3	∞	29	17	25	41	77	141
5	∞	29	26	18	22	38	74
6	∞	30	42	18	19	28	53
7	∞	34	55	19	18	22	38

Figure 2: Calculation of DTW matrix using Dynamic Programming (DTW path is highlighted in blue)

2. **Shapelets**: Informally, shapelets are time series subsequences which are in some sense maximally representative of a class. We will denote a shapelet by S and denote the length of a shapelet by L and the total number of shapelets to be learned by K.

3. **DTW Distance**: In time series analysis, dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed. The sequences are "warped" non-linearly in the time dimension. As mentioned above, figure 1 illustrates the idea behind DTW. Details and boundary conditions about the algorithm can be found in [10]. Here, we directly proceed to explain the estimation of the DTW distance using dynamic programming. The DTW value between two time series T and T' of lengths q and q' respectively, can be calculated recursively using Equation 1. Here, in the first term of the Equation 1, we have used Squared Distance as a distance measure between two points T_i and T'_j, whereas one can use any other distance measure. The value $DTW_{T_q,T'_{q'}}$ will be the final DTW distance between T_q and $T'_{q'}$, we will denote it by $DTW\left(T,T'\right)$ as defined in Equation 2.

$$DTW_{T_i,T'_j} = (T_i - T'_j)^2 + \min[DTW_{T_{i-1},T'_{j-1}}, DTW_{T_{i-1},T'_j}, DTW_{T_i,T'_{j-1}}] \quad (1)$$

$$DTW\left(T,T'\right) = DTW_{T_q,T'_{q'}} \quad (2)$$

An example for calculating DTW distance is shown in Figure 2. It presents two time series T_i = [1, 2, 3, 5, 6, 7] and T'_j = [5, 2, 6, 7, 9, 11]. Initially, a blank matrix is created with dimensions as $(q+1) \times (q'+1)$, which for this case results in a 7×7 matrix. The first cell (0,0) is filled as 0. Other cells from the first row and the first column

are filled with ∞. Values in the remaining cells of the from (i, j) are then calculated as the Squared Distance between corresponding points in time series plus the minimum of values in the adjacent cells $(i-1, j)$, $(i, j-1)$ and $(i-1, j-1)$. For example, the value in the cell (2,3) will be $(2-6)^2 + \min[16, 42, 17] = 32$. The DTW path can be traced backwards by starting from the last cell (q, q') and at each cell, moving to the minimum of previous 3 cells. In Figure 2, the resulting DTW path is highlighted in blue.

3.2 Calculating Distance Between Time Series and Shapelets

For calculating the distance between, the i^{th} time series T^i and the k^{th} shapelet S^k, the concept of *Sliding Windows* is used. All possible segments of T^i can be extracted by sliding a window of size L (which is also the length of shapelet) across T^i. Concretely, the segment of time series T^i starting at time-stamp j is defined as $T_j, \ldots, T_{j+L-1}$ and will be denoted by $T^{i,j}$. If the starting index of the sliding window is incremented by one, then there are total $J = Q - L + 1$ segments. For all of these segments we use Dynamic Time Warping (DTW) to calculate the distance between the shapelet S^k and every segment of the time series T^i.

The distance between the i^{th} time series T^i and the k^{th} shapelet S^k is defined as the minimum distance among all the DTW distances between the shapelet and each of the segments of the time series, as shown in Equation 3.

$$M_{i,k} = \min_{j=1,\ldots,J} \frac{1}{L} DTW\left(T^{i,j}, S^k\right) \quad (3)$$

We use these minimum distances to a finite number of shapelets as features for classifying the time series instances with relatively simple models such as a logistic regression.

3.3 Learning Time Series Shapelets with DTW (LTSD)

In this section we will show the linear classification model that we employ and discuss how its parameters as well as the shapelets can be learned using stochastic gradient descent.

3.3.1 Model

Having computed minimum distances as the new predictors, a linear model can predict target values $\hat{Y} \in \mathbb{R}^I$ via the predictors M and the linear weights $W \in \mathbb{R}^K$ as shown in Equation 4.

$$\hat{Y}_i = W_0 + \sum_{k=1}^{K} M_{i,k} W_k, \quad \forall i \in \{1, \ldots, I\} \quad (4)$$

Our model learns R different scales of shapelet lengths starting at a minimum $L^{\min}$ as $\{L^{\min}, 2L^{\min}, \ldots, RL^{\min}\}$. The shapelets therefore can be defined as $S \in \mathbb{R}^{R \times K \times *}$, which means that for each scale we will learn K-many shapelets, i.e. in total KR shapelets. The length of a shapelet at scale $r \in \{1, \ldots, R\}$ is $r \cdot L^{min}$. Consequently, the number of segments in a time series depends on the scale of the shapelet's length to be matched against and is $J(r) = Q - r \cdot L^{min} + 1$. We slightly reformulate Equation 4 to account for the R different length scales, the final prediction is then computed as shown in Equation 5.

$$\hat{Y}_i = W_0 + \sum_{r=1}^{R} \sum_{k=1}^{K} M_{r,i,k} W_{r,k} \quad (5)$$

In order to learn from multi-class targets $Y \in \{1, \ldots, C\}^I$ with C categories, we will convert the problem into C-many one-vs-all sub-problems. Each sub-problem will discriminate one class against all the others. The one-vs-all binary targets $Y^b \in \{0, 1\}^{I \times C}$ are defined in Equation 6.

$$Y^b_{i,c} = \begin{cases} 1 & Y_i = c \\ 0 & Y_i \neq c \end{cases}, \quad \forall i \in \{1, \ldots, N\}, \ \ \forall c \in \{1, \ldots, C\} \quad (6)$$

So, the target value will be as shown in Equation 7.

$$\hat{Y}^b_{i,c} = W_{c,0} + \sum_{r=1}^{R} \sum_{k=1}^{K} M_{r,i,k} W_{c,r,k} \quad (7)$$

We then use logistic regression classification model so we can interpret the predicted targets as probabilistic confidence. It operates by minimizing the logistic loss between the true targets Y and the predicted ones $\hat{Y}$ as shown in Equation 8.

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln\left(1 - \sigma(\hat{Y})\right) \quad (8)$$

where, $\sigma(Y) = \left(1 + e^{-Y}\right)^{-1}$

The logistic loss function together with regularization terms represent the regularized objective function, denoted as F in Equation 9. We aim to jointly learn the optimal shapelets S and the optimal linear hyper-plane W that minimizes the classification objective F.

$$\underset{S,W}{\operatorname{argmin}} \ \mathcal{F}(S, W) = \sum_{i=1}^{I} \mathcal{L}(Y_i, \hat{Y}_i) + \lambda_W ||W||^2 \quad (9)$$

In case of multi-class labels we minimize the following Equation 10,

$$\underset{S,W}{\operatorname{argmin}} \ \mathcal{F}(S, W) = \sum_{i=1}^{I} \sum_{c=1}^{C} \mathcal{L}(Y^b_{i,c}, \hat{Y}^b_{i,c}) + \lambda_W ||W||^2 \quad (10)$$

3.3.2 Classification of Test Instances

Classification of test instances is obvious in case when classes are binary. For multi-class classification, once the model is learned, a test instance indexed t is classified using the one-vs-all classifier which yields maximum confidence, as presented in Equation 11.

$$\hat{Y}_{\mathbf{t}} \leftarrow \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}} \ \sigma\left(\hat{Y}^b_{\mathbf{t},c}\right), \quad \forall t \in \{1, \ldots, I^{Test}\} \quad (11)$$

3.3.3 Differentiable Soft-Minimum Function

In order to compute the derivative of the objective function, all the involved functions of the model need to be differentiable. Unfortunately, the minimum function of Equation 3 is not differentiable and the partial derivative $\frac{\partial M}{\partial S}$ is not defined. A differentiable approximation to the minimum function is introduced in this section. For the sake of organizational clarity, let us call the distance between the j-th segment of the series i and the k-th shapelet at scale r as $D_{r,i,k,j}$ and define it as in Equation 12.

A differentiable approximation of the minimum function is the popular *Soft Minimum* function that is depicted in

Equation 13. A parameter $\alpha < 0$ controls the *precision* of the function and the soft minimum approaches the true minimum for $\alpha \to -\infty$.

$$D_{r,i,k,j} = \frac{1}{r \cdot L^{\min}} DTW\left(T^{i,j}, S^{r,k}\right) \quad (12)$$

$$M_{r,i,k} \approx \hat{M}_{r,i,k} = \frac{\sum_{j=1}^{J(r)} D_{r,i,k,j}\ e^{\alpha D_{r,i,k,j}}}{\sum_{j'=1}^{J(r)} e^{\alpha D_{r,i,k,j'}}} \quad (13)$$

3.3.4 Per-Instance Objective

We use stochastic gradient descent approach that remedies the classification error caused by one instance at a time. Equation 14 demonstrates the decomposed objective function F_i, which corresponds to a decomposition of the objective of Equation 9 into per-instance losses for each time series.

$$\mathcal{F}_i = \mathcal{L}(Y_i, \hat{Y}_i) + \frac{\lambda_W}{I} \sum_{r=1}^{R} \sum_{k=1}^{K} {W_{r,k}}^2 \quad (14)$$

In case of multi-class labels, it will be as shown in Equation 15

$$\mathcal{F}_{i,c} = \mathcal{L}(Y^b_{i,c}, \hat{Y}^b_{i,c}) + \frac{\lambda_W}{IC} \sum_{r=1}^{R} \sum_{k=1}^{K} {W_{c,r,k}}^2 \quad (15)$$

3.3.5 Gradients for Shapelets

To update the shapelets, we need to compute gradients of the objective function with respect to the shapelets. The gradient at the l^{th} point in the k^{th} shapelet with respect to the objective of the i^{th} time series is defined in Equation 16.

$$\frac{\partial \mathcal{F}_i}{\partial S_{r,k,l}} = \frac{\partial \mathcal{L}(Y_i, \hat{Y}_i)}{\partial \hat{Y}_i} \frac{\partial \hat{Y}_i}{\partial \hat{M}_{r,i,k}} \sum_{j=1}^{J} \frac{\partial \hat{M}_{r,i,k}}{\partial D_{r,i,k,j}} \frac{\partial D_{r,i,k,j}}{\partial S_{r,k,l}} \quad (16)$$

Also, the gradient of the loss with respect to the predicted target and the gradient of the predicted target with respect to the minimum distances is shown in Equations 17- 18.

$$\frac{\partial \mathcal{L}(Y_i, \hat{Y}_i)}{\partial \hat{Y}_i} = -\left(Y_i - \sigma\left(\hat{Y}_i\right)\right) \quad (17)$$

$$\frac{\partial \hat{Y}_i}{\partial \hat{M}_{r,i,k}} = W_{r,k} \quad (18)$$

The gradient of the overall minimum distance with respect to a segment distance is presented in Equation 19.

$$\frac{\partial \hat{M}_{r,i,k}}{\partial D_{r,i,k,j}} = \frac{e^{\alpha D_{r,i,k,j}} \left(1 + \alpha \left(D_{r,i,k,j} - \hat{M}_{r,i,k}\right)\right)}{\sum_{j'=1}^{J(r)} e^{\alpha D_{r,i,k,j'}}} \quad (19)$$

Now, what remains is to calculate the gradient of a segment distance with respect to a shapelet point, which is covered in the following. We need to the find the gradient of $D_{r,i,k,j}$ with respect to each point l in the shapelet $S_{r,k}$ in order to update each of the shapelet's entries. From Equation 1, the resulting gradient can be derived easily. Differentiation of the first term is straightforward. The second term is the minimum of three previously calculated distance terms, where we could also use the differentiable Soft Minimum, but it is not needed. As we have already computed the DTW path, we know which of these three terms is the minimum and therefore is contributing to the current term. Then we can just check, whether that term includes the same point on the shapelet as the current term. If this is the case, we need to add its gradient into the current calculation. If it is otherwise, we do not need to add anything, as the previous term consists of a different shapelet point and does not count here, as we are calculating the gradient with respect to the current shapelet point. The resulting recursion is shown in Equation 20.

	5	2	6	7	9	11
1	-8	-10	-20	-32	-48	-68
2	-6	0	-8	-18	-32	-50
3	-4	2	-6	-14	-26	-42
5	0	6	-2	-6	-14	-26
6	2	8	0	-2	-8	-18
7	4	10	2	0	-4	-12

Figure 3: Calculating DTW Gradient(Values that are highlighted in yellow are taken as Gradients for the L points of the shapelet)

$$\delta_{l,j+l_1} = 2\left(S_l - T_{j+l_1}\right) + \begin{cases} \delta_{l,j+l_1-1} & \text{if the minimum term} \\ & \text{is } [l, j+l_1-1] \\ 0 & Otherwise \end{cases} \quad (20)$$

where $0 \le l, l_1 \le L-1$.

An example calculation is shown in Figure 3, by taking [1, 2, 3, 5, 6, 7] as the shapelet S and [5, 2, 6, 7, 9, 11] as the time series segment T_j. The DTW matrix for the same shapelet and time series is already shown in Figure 2. Again, a matrix of size $q \times q'$ is created in order to compute the gradient values. For the first column, the second term of Equation 20 is always set to 0. Out of this reason, it is just filled with first term. Consider for example cell (0,0), it's value resorts to simply $2 \times (1-5) = -8$. All remaining cells are filled as given in Equation 20. For example, cell (1,1) will contain the value $2 \times (2-2) + 0 = 0$, because in the DTW matrix calculation the minimum term comes from the cell $[l-1, l_1-1]$. On the other hand, cell (5,5) is $2 \times (7-11) + (-4) = -12$, because in the DTW matrix calculation the minimum term stems from the cell $[l, l_1 - 1]$.

Now, we need to compute the final gradient corresponding to each point in the shapelet. Points on the shapelets correspond to rows in the gradient matrix. The rightmost value on the DTW path in each row is the most updated gradient value, which shows how particular points (associated with row) on the shapelet should change. Thus, we use it as the gradient for that point on the shapelet. For this example, gradient values of the six points on the shapelet are [-8,0,2,-2,0,-12]. They are highlighted in yellow in the Figure 3.

We can also compute the same values by simply applying Equation 20 on the DTW path only and not calculating the other terms in the gradient matrix, which involves less time. For multi-class classification, the derivative of the per-cell objective $F_{i,c}$ with respect to each shapelet $S_{r,k,l}$ is shown in Equation 21.

$$\frac{\partial \mathcal{F}_{i,c}}{\partial S_{r,k,l}} = -\left(Y^b_{i,c} - \sigma\left(\hat{Y}^b_{i,c}\right)\right) W_{c,r,k} \frac{\partial \hat{M}_{r,i,k}}{\partial S_{r,k,l}} \quad (21)$$

Algorithm 1 Learning Time-Series Shapelets

Require: Time series $T \in \mathbb{R}^{I \times Q}$, Binary labels $Y^b \in \mathbb{R}^{I \times C}$, Number of Shapelets K, Scales of shapelet lengths $R \in \mathbb{N}$, Minimum Shapelet Length $L^{\min}$, Regularization λ_W, Learning Rate η, Number of iterations: maxIter

Ensure: Shapelets $S \in \mathbb{R}^{R \times K \times *}$, Classification weights $W \in \mathbb{R}^{R \times K \times C}, W_0 \in \mathbb{R}^C$

Initialize S, W, W_0
for iteration=$\mathbb{N}_1^{\text{maxIter}}$ **do**
  for $i = 1, \ldots, I$ **do**
    for $c = \{1, \ldots, C\}$ **do**
      for $r = \{1, \ldots, R\}$, $k = \{1, \ldots, K\}$ **do**
        $\mathbf{W}_{c,r,k} \leftarrow W_{c,r,k} - \eta \frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,r,k}}$
        for $l = 1, \ldots, L$ **do**
          $S_{r,k,l} \leftarrow S_{r,k,l} - \eta \frac{\partial \mathcal{F}_{i,c}}{\partial S_{r,k,l}}$
        end for
      end for
      $\mathbf{W}_{c,0} \leftarrow W_{c,0} - \eta \frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,0}}$
    end for
  end for
end for
return S, W, W_0

3.3.6 Gradients for Classification Weights

The hyper-plane weights W are also learned to minimize the classification objective via stochastic gradient descent. The partial gradient for updating each weight $W_{r,k}$ is defined in Equation 22 and the partial gradient for the bias term W_0 is defined in Equation 23.

$$\frac{\partial \mathcal{F}_i}{\partial W_{r,k}} = -\left(Y_i - \sigma\left(\hat{Y}_i\right)\right) \hat{M}_{r,i,k} + \frac{2\lambda_W}{I} W_{r,k} \quad (22)$$

$$\frac{\partial \mathcal{F}_i}{\partial W_0} = -\left(Y_i - \sigma\left(\hat{Y}_i\right)\right) \quad (23)$$

For multi-class classification, the gradients of the per-cell objective with respect to the generalized weights and the bias terms are presented in Equations 24-25.

$$\frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,r,k}} = -\left(Y^b_{i,c} - \sigma\left(\hat{Y}^b_{i,c}\right)\right) \hat{M}_{r,i,k} + \frac{\lambda_W W_{c,r,k}}{IC} \quad (24)$$

$$\frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,0}} = -\left(Y^b_{i,c} - \sigma\left(\hat{Y}^b_{i,c}\right)\right) \quad (25)$$

3.3.7 Learning Algorithm

As we have derived the gradients of the shapelets and the weights, we can now introduce the overall learning algorithm. It iterates in a series of epochs and updates the values of the shapelets and weights in the negative direction of the derivative with respect to the classification objective of each training instance and class pair. The steps of the learning process are shown in Algorithm 1. Shapelets and weights are initialized by random values. The algorithm iterates over all training instances I, classes C and updates all RK shapelets S and the weights W, W_0 by a learning rate η.

3.3.8 Convergence

The convergence of the algorithm depends on two parameters, the learning rate η and the maximum number of iterations. High values for the learning rate can minimize the objective in less iterations, but pose the risk of divergence, while small learning rates require more iterations and therefore more time. Subsequently, the learning rate and the number of iterations should be estimated via cross-validation from the training data.

4. EXPERIMENTAL RESULTS

4.1 Datasets

We evaluated our method LSTD on 11 real world data set from the UCR[1] repository using the default train and test splits and on a synthetic data set created by us. Table 1 summarizes the details about all the data sets. In the next section we explain the creation of the synthetic data set.

4.1.1 Constructing Synthetic Dataset

Consider a classification task, where the class of a time series instance depends on a particular sequence of events in time series, and does not depend on the temporal distance between these events. Additionally, let the shape of the events be same across all the classes. By the shape of an event, we mean the plot of the event against time axis. Thus, the shape of any of the events can not be used as a discriminative factor between the classes. Here, the LTS approach will fail, as Euclidean distance will fail to take care of the varying distances between events. For LTS to work, it will need a very large training data set to learn many shapelets for each of the class, corresponding to all the possible positions of the events in a particular sequence. This will neither lead to interpretable shapelets nor is it feasible. Out of this reason, we need an approach that can learn just a few shapelets instead of infinitely many of them to perform the classification task. If we use DTW as the distance measure for learning shapelets, it can warp the portion of time axis in a training instance where there is no event, and learn a very few number of shapelets, representative of the overall sequence of events in a class. While, this may be achieved by using DTW, when there is no noise at all in the time series; DTW can't work well if the amount of the noise is increased, simply because as soon as noise is sufficient enough, DTW will mix up the noise with the actual events.

We have constructed the synthetic data set having two classes C_1 and C_2, which defers in the order of two events. Let these events be a low peak and a high peak. Denote them by E_1 and E_2 respectively. The average height of the E_1 is kept at 50% of the average height of the E_2. For class C_1, let the order of the events be first E_1 and then E_2. For class C_2, it is first E_2 and then E_1. Also, we constructed two data sets based on the amount of the background noise added. Noise was added by selecting a random value in the range $[0, 1]$ and then scaling it with an appropriate integer, to keep the average noise level at a fixed value. For the first data set, average noise level was kept at 25% of the average height of the E_1. For the second one, it was kept at 40% of the average height of the E_1. Denote these data sets by LNLD (Low Noise Level Dataset) and HNLD (High Noise Level Dataset) respectively. Both of them had 50 training instances and 100 test instances. Lengths of the time series instances were kept at 300.

[1] http://www.cs.ucr.edu/~eamonn/time_series_data/

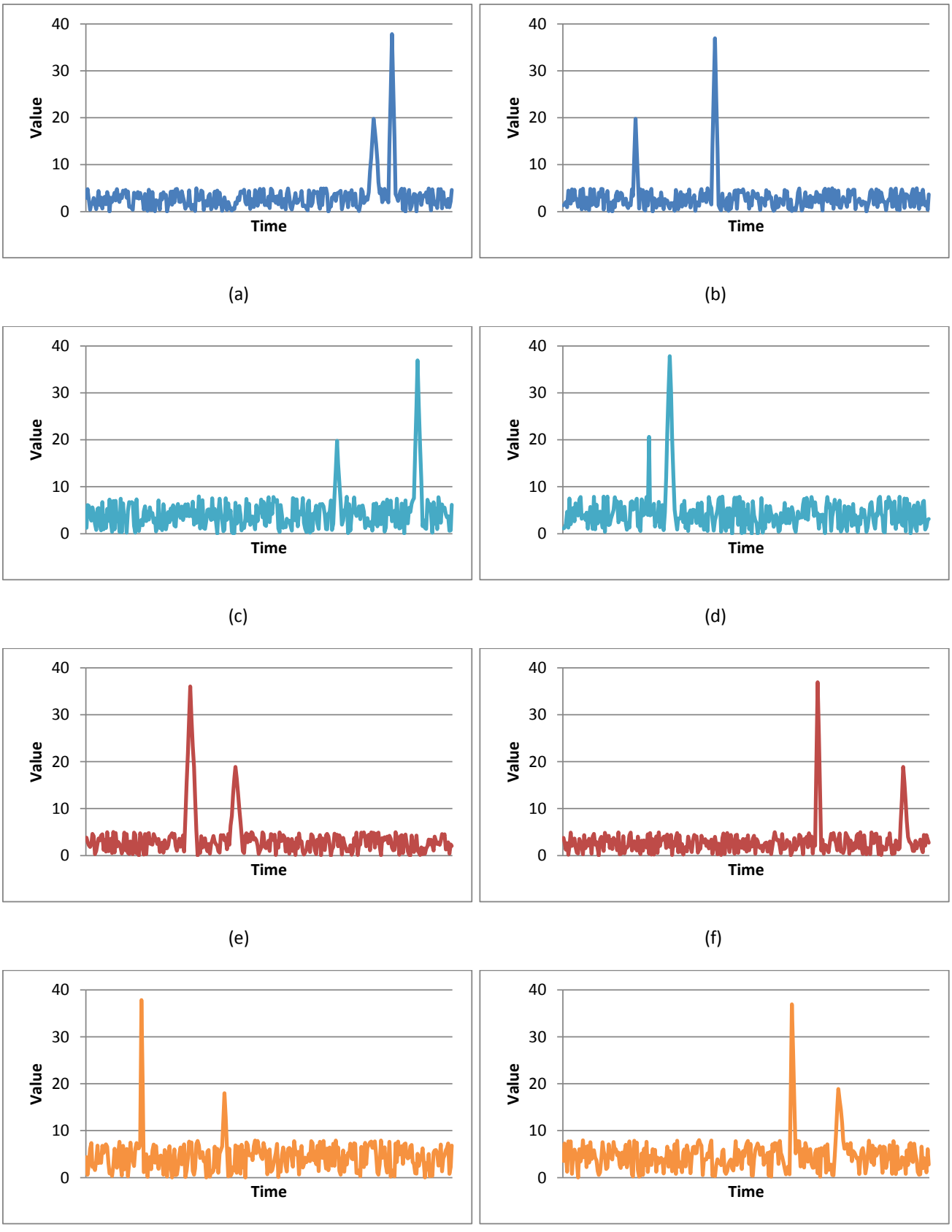

Figure 4: Samples from Synthetic Dataset
(a)(b) Instances of class C_1 having low noise level
(c)(d) Instances of class C_1 of having high noise level
(e)(f) Instances of class C_2 of having low noise level
(g)(h) Instances of class C_2 having high noise level

Table 1: Dataset Information and Parameter Search Results

Dataset Information				Parameter Values				
Dataset	*Train/Test*	*Length*	*Classes*	L^{min}	K	R	λ_W	*maxIter*
Coffee	28/28	286	2	0.2	0.05	3	0.01	600
Diatom	16/306	345	4	0.1	0.15	3	0.1	1000
ECGFiveDays	23/861	136	2	0.2	0.05	3	0.1	2400
GunPoint	50/150	150	2	0.15	0.05	4	0.1	1600
ItalyPowerDemand	67/1029	24	2	0.2	0.3	4	0.01	1800
MoteStrain	20/1252	84	2	0.1	0.3	3	0.01	800
SonyAIBO	20/601	70	2	0.1	0.1	2	0.1	1600
Symbols	25/995	398	6	0.2	0.05	2	0.01	200
SyntheticControl	300/300	60	6	0.1	0.05	4	0.01	200
Trace	100/100	275	4	0.1	0.1	2	0.1	200
TwoLeadECG	23/1139	82	2	0.1	0.3	3	0.01	1000
LNLD	50/100	300	2	0.15	0.05	4	0.1	200
HNLD	50/100	300	2	0.15	0.05	4	0.1	200

Some instances of the synthetic data sets are given in the Figure 4.

4.2 Hyper-parameter Search

Similar to LTS, LTSD also requires tuning of some hyper-parameters. They were found through using a grid search approach using cross-validation over the training data. The number of shapelets was searched in a range of $K \in \{0.05, 0.1, 0.15, 0.3\}$, as a fraction of the series length, e.g. $K = 0.3$ means 30% of Q. Similarly, $L^{\min} \in \{0.1, 0.15, 0.2\} \times 100\%$ of Q, while three scales of shapelet lengths were searched from $R \in \{2, 3, 4\}$. The regularization parameter was one of $\lambda_W \in \{0.01, 0.1\}$. The learning rate was kept fixed at a small value of $\eta = 0.01$, while the number of iterations can be maximum 4000, at intervals of 200; i.e., 200, 400, ..., 3800, 4000. The hyper-parameters chosen are shown in Table 1.

4.3 Competing Methods

We compare LTSD to two competing methods, LTS and DTW. LTS was already shown to be better than other techniques such as Shapelet Tree Method, Classifiers and other related methods like Fast Shapelets on real world data sets in [2]. To avoid the redundancy we do not compare against them again. Our proposed approach is compared against DTW, so that it can reflect, how usage of DTW in LTSD affects the final results.

4.4 Algorithmic Complexity

For extracting the most discriminating shapelet of a particular length L in a binary-class classification, LTS requires $\mathcal{O}(IL^2 \times \text{maxIter})$ running time, while LTSD requires $\mathcal{O}(IL^3 \times \text{maxIter})$. The resulting increase in time complexity is due to the usage of DTW as a distance measure instead of the Euclidean distance.

4.5 Accuracy

LTSD is compared against the baselines mentioned in the section above. Accuracy is used as a measure for the comparison, which is the fraction of the test instances that are truly classified. Tables 2 and 3 summarizes the results for the real world data sets and the synthetic data set respectively.

Classification of time series in all the real world data sets is based on the different shapes present in it. None of these data sets have classification based on the order of the events (as explained in Section 4.1.1). As we can see in the table 2, LTS works well on them by learning the shapelets, with almost no scope of improvement. But DTW fails to perform well, as it just assigns the class of the nearest training instance to the test instances. Out of these 11 data sets, LTS and LTSD give same accuracy in 7 of them. Among the remaining 4 data sets, LTSD perform better in 3 of them. In summary, it shows a little improvement over LTS.

Table 2: Accuracies for the 11 real world data sets

Dataset	LTS	DTW	Proposed Approach - LTSD
Coffee	**1.00**	0.464	**1.00**
Diatom	0.951	0.925	**0.984**
ECGFiveDays	**1.00**	0.828	**1.00**
GunPoint	**1.00**	0.913	**1.00**
ItalyPowerDemand	**0.962**	0.961	**0.962**
MoteStrain	**0.913**	0.816	0.91
SonyAIBO	0.952	0.699	**0.959**
Symbols	0.959	0.934	**0.981**
SyntheticControl	**1.00**	0.973	**1.00**
Trace	**1.00**	0.990	**1.00**
TwoLeadECG	**1.00**	0.795	**1.00**

Table 3: Accuracies for the synthetic data sets

Dataset	LTS	DTW	Proposed Approach - LTSD
SNLD	0.66	**1.00**	**1.00**
HNLD	0.60	0.82	**0.85**

For the synthetic data sets, from table 3 we can see that LTS fails to work properly, as expected. Relatively, DTW and LTSD work much better than it, for the reasons explained in Section 4.1.1. Although for SNLD, both DTW and LTSD gives the same accuracy, for HNLD, LTSD performs better than DTW.

Results on these data sets show that the proposed approach works well in both the types of data sets, when the classification is based on the discriminative shapes present in the time series instances, as well as when it is based on the order of events.

5. CONCLUSION

In this paper, we have extended the LTS method by using Dynamic Time Warping as the distance measure. It proved to be more accurate in both the real world data sets and the synthetic data set over LTS and DTW. It outperformed LTS significantly in the synthetic data sets and DTW significantly in the real world data sets. Thus, it is suitable for both the types of the classification tasks, either it is based on the discriminative shapes present in the time series instances, or it is the order of the events that matter. So, it combines the good characteristics of both the LTS and DTW approaches and therefore yields the best results.

6. ACKNOWLEDGMENT

Mit Shah was co-funded by the German Academic Exchange Service (DAAD).

7. REFERENCES

[1] 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 131–140. IEEE Computer Society, 2012.

[2] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401. ACM, 2014.

[3] B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized dtw prototypes. In *IEEE International Conference on Systems Man and Cybernetics*, 2010.

[4] B. Hartmann, I. Schwab, and N. Link. Prototype optimization for temporarily and spatially distorted time series. In *the AAAI Spring Symposia*, 2010.

[5] Q. He, F. Zhuang, T. Shang, Z. Shi, et al. Fast time series classification based on infrequent shapelets. In *11th IEEE International Conference on Machine Learning and Applications*, 2012.

[6] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 2013.

[7] J. Lines and A. Bagnall. Alternative quality measures for time series shapelets. In *Intelligent Data Engineering and Automated Learning*, volume 7435 of *Lecture Notes in Computer Science*, pages 475–483. 2012.

[8] J. Lines, L. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

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

[10] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

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

[12] P. Sivakumar and T. Shajina. Human gait recognition and classification using time series shapelets. In *IEEE International Conference on Advances in Computing and Communications*, 2012.

[13] Z. Xing, J. Pei, and P. Yu. Early classification on time series. *Knowledge and information systems*, 31(1):105–127, 2012.

[14] Z. Xing, J. Pei, P. Yu, and K. Wang. Extracting interpretable features for early classification on time series. *Proceedings of the 11th SIAM International Conference on Data Mining*, 2011.

[15] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[16] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised-shapelets. In *Proceedings of the 12th IEEE International Conference on Data Mining*, 2012.