Time Series Shapelet Classification through Learned Distances

Anonymous Submission
DM579

Abstract—**Shapelets are small subsequences of time series that can be used to classify unlabeled time series. Shapelets can be used in stand-alone classifiers and are an essential part of the most accurate ensemble methods for time series classification. Previous shapelet research has focused on finding the most efficient approaches to discover shapelets and how to incorporate shapelets into different classification methods. However, a shapelet's ability to accurately classify depends on a single measurement: its distance from each sample. Euclidean distance, while robust, is the only distance metric that has been used with shapelets. This paper explores an alternative to Euclidean distance. First, given a shapelet, we introduce a unique mapping that uses the shapelet to project time series training samples into Euclidean point space. In this space, we show that the current approach of using a fixed Euclidean distance threshold for shapelet classification is equivalent to creating a spherical decision boundary. We then use logistic regression in this Euclidean space to learn a decision boundary that can more accurately capture the structure of the data. This approach can improve the classification power of the shapelet by providing better separation between samples of different classes. We compare the performance of decision boundaries created by Euclidean distance and logistic regression using 2 shapelet methods on 30 publicly available datasets. Due to its simplicity, Euclidean distance tends to perform better than logistic regression when very few training samples are available. However, when given a sufficient number of training samples, results show that decision boundaries learned by logistic regression outperform fixed Euclidean distances. Finally, we show it is possible to predict whether Euclidean distance or logistic regression will perform better using only training data.**

Index Terms—**pattern analysis, supervised learning, time series classification**

I. Introduction

Time series classification has numerous applications to a wide variety of domains. For example, classification of ECG signals can identify when a patient is suffering from congestive heart failure so emergency medical personnel can be notified to provide life-saving medical treatment [1]. Classification in food spectroscopy can improve food safety and quality assurance. For instance, infrared spectra of beef that has been cut, ground, cooked, or is uncooked can identify whether it is pure or contains adulterations [2]. Finally, in the power grid, classification of electrical disturbance events can enable an automated relay to apply corrections faster than a human possibly could, preventing cascading blackouts before they occur [3]. Additional application domains include finance, entertainment, medicine, biometrics, human motion, chemistry, astronomy, robotics, entomology, and wildlife management [4]–[7]. Many other problems can also be mapped to time series [8].

The most studied time series classification methods are global, meaning they compare one entire time series to another. Well known global methods such as 1-Nearest Neighbor and Dynamic Time Warping [9] have been used extensively, but global classification methods suffer from three limitations. First, since global methods compute distances using entire time series, they are computationally costly both in space and time complexity. Second, they are sensitive to noise throughout an entire signal. Third, the classification results are not interpretable because they do not detail *why* an unlabeled object is assigned to a given class. Time series shapelets overcome these limitations by performing classification locally.

A shapelet [8] is a short segment of a time series that represents a highly discriminative feature and can be used for fast, accurate classification. Due to their short length, shapelets are robust to background noise, have faster computation, and provide intuition on the classification of unlabeled samples.

Shapelets can be used in stand-alone classification algorithms including decision trees [8], [10], [11], support vector machines, and Bayesian networks [12], [13], and are integral parts in the most accurate time series classification ensembles [14]. However, a shapelet's ability to distinguish between samples rests solely on its distance from each sample. Despite this dependency, Euclidean distance is the only metric that has been utilized during shapelet discovery and classification.

Contributions: This paper explores the impact of learning a custom distance instead of using Euclidean distance for shapelet classification. We start by introducing a novel mapping scheme to transform time series training samples into points in Euclidean space based on the shapelet used for classification. We show that in this space, a Euclidean distance threshold equates a spherical decision boundary. This spherical boundary can fail to capture the structure of the training data since the distribution of samples is rarely uniform, leading to incorrect classifications. Next, we apply logistic regression to the mapped training samples in this Euclidean space. The output from logistic regression is used in two ways.

First, for classification approaches that use shapelets in decision trees, the decision boundary learned by logistic regression can be used in place of a fixed Euclidean distance cutoff. The boundary learned by logistic regression has the ability to more effectively model sample distributions, leading to more accurate classification given sufficient samples.

Second, for shapelet classification methods that use multiple shapelets with multiple distances, we replace Euclidean distance with the probability distributions learned for each shapelet by logistic regression, effectively treating these distributions as custom-learned distances for each shapelet.

Results on 30 publicly available datasets show that Euclidean distance tends to perform better with few training samples, since only a single parameter is learned. However, when given 100 training samples or more, shapelets using a decision boundary learned through logistic regression tend perform better than a fixed Euclidean distance threshold. Furthermore, when sufficient training samples are available, we show the impact of each decision boundary is largely predictable using cross-validated training data.

II. Related Work

Time series shapelets require two separate steps: shapelet discovery and shapelet classification. Shapelet discovery generates shapelet candidates and evaluates them to determine the best shapelets for a given dataset. Classification then combines these shapelets into a classification algorithm.

There are two different approaches to shapelet discovery: searching for shapelets and learning shapelets. The original approach proposed by [8] is search-based and uses brute force to explore all possible subsequences of all training samples. Speedups to this brute force exploration include subsequence distance early abandon and admissible entropy pruning [8], multi-length indexing and dynamic stepping [15], removal of duplicate shapelets through clustering [12], and mapping time series to SAX words to compress the search space [10].

Alternatively, shapelet discovery can be formulated as an optimization problem. The best shapelets can then be learned through heuristic gradient descent [16] or generalized eigenvectors combined with the fused lasso regularizer (FLAG) [17]. The problem can also be formulated as a convex optimization [18].

Once shapelets are discovered, they can be used in a variety of classification schemes. The original approach [8] used the best shapelet found with a computed distance threshold to build a decision node, which can be executed recursively to build a decision tree. Shapelets can also be combined using logical AND and OR conjunctions [11]. Linear regression can be used to separate samples based on their distances from two shapelets [16]. Finally, Shapelet Transforms [12], [13] provide a transformation method to feed data from multiple shapelets into a machine learning ensemble consisting of C4.5 trees, random forests, rotational forests, Bayesian networks, nearest neighbor, support vector machines, and naïve Bayes.

III. Definitions and Notation

We start with the necessary notation to represent time series, shapelets, and distances, summarized in Table I.

Time Series: A *time series* $T = t_1, t_2, ..., t_L$ is an ordered set of points in $\mathbb{R}$. A *time series subsequence* $S_i = t_i, t_{i+1}, ..., t_{i+l-1}$ is a contiguous subsequence of points in T of length $|S_i| = l$.

Learning Environment: We assume a supervised training environment consisting of dedicated training and test datasets. The *training dataset*, $\mathbb{T}_{train} = \{T_1, T_2, ..., T_n\}$, is a set of time series. $\mathbb{C}_{train}$ denotes the list of class labels. Each time series T_i has a class label $c_i \in \{1, ..., C\}$, where C is the number of classes. A *test dataset*, $\mathbb{T}_{test}$, is similarly formatted with different samples, but the associated class labels $\mathbb{C}_{test}$ are hidden during testing and used only for performance evaluation. While the scope of this paper is limited to supervised learning in one-dimensional time series, this can be easily expanded to use shapelets in multiple dimensions [15], [18] or in an unsupervised setting [19].

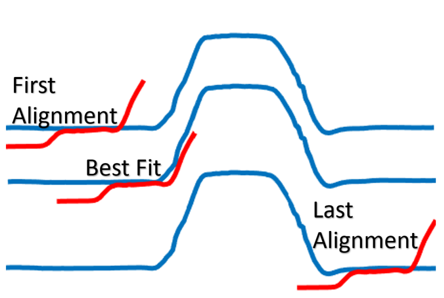
Shapelet Variable Definitions	
Datasets, Samples, and Class Labels	
Variable	**Explanation**
$\mathbb{T}_{train}$	Training dataset of time series samples
$\mathbb{T}_{test}$	Test dataset of time series samples
$\mathbb{C}_{train}$, $\mathbb{C}_{test}$	The vectors of class labels for each dataset
T_i	The i^{th} sample in a dataset
C	Number of classes in the datasets
c_i	The class label for the i^{th} sample
n	Number of samples in a dataset
L	Length of each time series in the datasets
Shapelet Discovery/Classification	
Variable	**Explanation**
Sh_{cand}	A shapelet candidate
l	Length of a shapelet or shapelet candidate
$Dist()$	Distance function used for classification
$I(D)$	Entropy, measures the diversity of a dataset
d	Distance threshold, partitions the training dataset
$Gain(Sh_{cand})$	Gain, measures the quality of a candidate
Sh	A shapelet, chosen for classification
j	The offset denoting the best fit for a shapelet Sh to a sample T_i

TABLE I: Shapelet Terminology

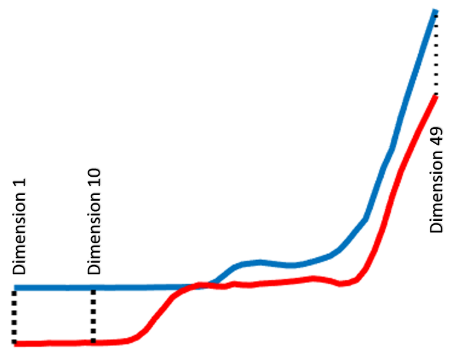
Shapelet Candidate: A *shapelet candidate* $Sh_{cand} = sh_1, sh_2, ..., sh_l$ is a short time series of length l that is evaluated as a possible shapelet during shapelet discovery. Typically $l << L$, where L is the length of the time series in the training dataset, $\mathbb{T}_{train}$. Most discovery methods are search-based, meaning they search for shapelets in subsequences of time series samples in the training dataset [8], [10]–[13]. For these methods, any subsequence in any training sample is a shapelet candidate. However, learning-based methods are not restricted to subsequences in the training dataset [16]–[18]. For these methods, *any* time series of length l is a shapelet candidate, meaning they cover an infinite number of candidates.

Distance: Given a time series sample T_i and a shapelet candidate Sh_{cand}, the distance between them is defined by a *distance function*, abbreviated as $Dist(T_i, Sh_{cand})$. Since the time series is longer than the shapelet candidate, the best fit distance is used, shown in Figure 1a and defined as follows.

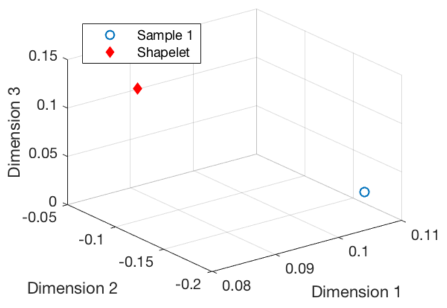
The *closest subsequence*, $S_{closest}$, of T_i to Sh_{cand} is the contiguous subsequence $S \in T_i$, $s.t.|S| = |Sh_{cand}|$, which minimizes $Dist(S, Sh_{cand})$ out of all $S \in T_i$. This *best fit distance*, $Dist(S_{closest}, Sh_{cand})$, is used as the distance between the time series T_i and the shapelet candidate Sh_{cand} (i.e. $Dist(T_i, Sh_{cand}) \equiv Dist(S_{closest}, Sh_{cand})$. While $Dist()$ could be any distance function, point-wise Euclidean distance



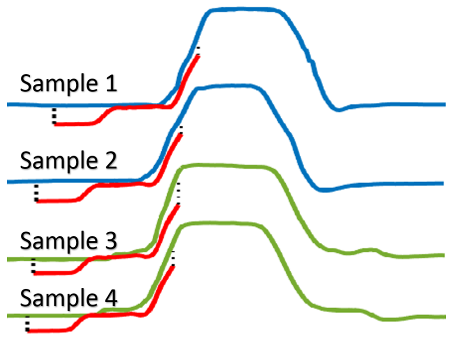
(a) For a single sample (blue), there are multiple possible alignments for a shapelet (red). Shapelets only use the best fitting alignment.



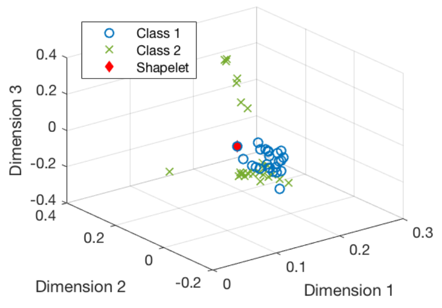
(b) Each point in the closest subsequence from the best fit is treated as a dimension, mapping the subsequence to a point in $\mathbb{R}^l$.



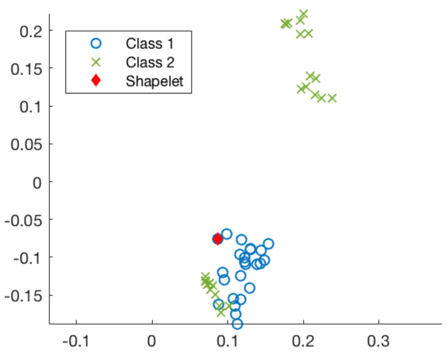
(c) This shapelet has length $l = 49$, so the Euclidean space has 49 dimensions. Here 3 dimensions of the Euclidean space are shown.



(d) There are 50 samples in this training dataset, 4 of which are shown above. Using the closest alignment for each sample, ...



(e) ... the entire training dataset is mapped to Euclidean space $\mathbb{R}^l$. The mapped points from all 50 training samples are shown here.



(f) Finally, since this space is sparse, PCA is used to reduce the number of dimensions. Classification is now performed in this space.

Fig. 1: A proposed method to map time series samples to Euclidean points using a shapelet previously discovered for the GunPoint dataset. Each sample is mapped to a point $\in \mathbb{R}^l$ using the sample's closest subsequence to the shapelet.

is the only distance metric that has been used with shapelets.

Binary Partition: Using a distance threshold d, a shapelet candidate Sh_{cand} creates a binary partition of the training dataset $\mathbb{T}_{train}$ into two smaller datasets, $\mathbb{T}_{train\_L}$ and $\mathbb{T}_{train\_R}$. $\mathbb{T}_{train\_L}$ contains samples within the distance d of the shapelet candidate, and $\mathbb{T}_{train\_R}$ contains samples that exceed the distance d. d is always the distance that results in the optimal information gain, defined in Equation 1, and can be found algorithmically, as in [8].

$$Gain(Sh_{cand}) = I(\mathbb{T}_{train}) - \frac{|\mathbb{T}_{train\_L}|}{|\mathbb{T}_{train}|} I(\mathbb{T}_{train\_L}) - \frac{|\mathbb{T}_{train\_R}|}{|\mathbb{T}_{train}|} I(\mathbb{T}_{train\_R}) \tag{1}$$

Entropy: *Entropy* measures the diversity of the class labels for a given dataset D, and is computed as $I(D) = \sum_{i=1}^{C} -p_i log(p_i)$. Here $p_i = \frac{n_i}{n}$ is the proportion of samples n_i to the total samples in the dataset $n = |D|$ for class i.

Information Gain: *Information gain* is the standard metric used to measure the quality of shapelets based on how they partition the training dataset [8]. Given a shapelet candidate Sh_{cand} for the training dataset $\mathbb{T}_{train}$, the gain is defined in Equation 1 as the reduction in entropy after the dataset is partitioned by the shapelet with its distance threshold d. Shapelet candidates that cleanly separate samples from different classes have high information gain.

Shapelet: A *shapelet* is a shapelet candidate chosen by an algorithm for classification based on its information gain. Depending on the classification method, the top k shapelets with the highest gain are chosen. Shapelet candidates can be clustered after they are evaluated so that multiple shapelets that are nearly identical are not chosen [12].

We detail how methods build shapelets into classifiers in Section V. First, we introduce an approach to learn a custom distance for each shapelet in the next section.

IV. Transforming Euclidean Space through Logistic Regression

Euclidean distance is the only metric that has been used with shapelets. This section introduces an approach to view shapelets and samples from the training dataset in Euclidean point space instead of as time series. Instead of computing

Algorithm 1: Learning Distance with Logistic Regression

```
  Input: Training Dataset $\mathbb{T}_{train} = [T_1, ..., T_n]$,
         Training Class Labels $\mathbb{C}_{train} = [c_1, ..., c_n]$,
         Shapelet $Sh = sh_1, sh_2, ..., sh_l$,
         PCA Minimum Variance Explained Threshold $\nu$
  Output: PCA Coefficients $PCA_{coeff}$,
          Parameter Coefficients $\Theta = [\Theta_0, \Theta_1, ...\Theta_{2m+1}]$
  /* First, for each sample, map the closest
     subsequence to the shapelet to a point
     in Euclidean Space $\mathbb{R}^l$.                      */
1 $\mathbb{E}_{map}$ = {}
2 for i = 1 to $n = |\mathbb{T}_{train}|$ do
3     $f_{Sh} : T_i \mapsto <t_j, t_{j+1}, ..., t_{j+l-1}>$
4     $\mathbb{E}_{map}[i] = <t_j, t_{j+1}, ..., t_{j+l-1}>$
  /* Next, reduce the dimensions of the point
     space using PCA so that at least $\nu$% of
     the variance is explained.                  */
5 $PCA_{coeff} = PCA(\mathbb{E}_{map}, \nu)$
6 $\mathbb{E}_{pca} = \mathbb{E}_{map}$ x $PCA_{coeff}$
  /* Use a quadratic, axis-aligned decision
     function.                                   */
7 $X = [x_0 + x_1 + x_1^2 + x_2 + x_2^2 + ... + x_m + x_m^2]$
  /* Solve for parameter coefficients and
     return.                                     */
8 $\Theta = \arg\min_{\Theta} J(\Theta|\mathbb{E}_{pca}, \mathbb{C}_{train})$
9 Return $\Theta, PCA_{coeff}$
```

a fixed Euclidean cutoff distance, this will enable custom distances to be learned for each shapelet.

A. Mapping Time Series to Euclidean Points

Suppose a shapelet has already been discovered for training dataset T_{train}. Instead of viewing every training sample as a time series, we project each training sample into an l-dimensional Euclidean space, where l is the length of the shapelet. This projection will allow more complex classification methods to be used in place of Euclidean distance, described in Section IV-C. The projection is explained below.

For a shapelet $Sh = sh_1, ..., sh_l$ of length l and a sample $T_i = t_1, ..., t_L$ of length L, there are $L - l + 1$ possible alignments to compute the distance between them. In other words, there are $L-l+1$ contiguous subsequences from sample T_i that are compared to the shapelet. The first subsequence is $t_1, t_2, ..., t_l$. The second subsequence is $t_2, t_3, ...t_{l+1}$, and so on, with the last subsequence being $t_{L-l+1}, t_{L-l+2}, ...t_L$ as shown in Figure 1a.

Instead of viewing each of these subsequences as time series, the subsequences compared to the shapelet can be viewed as points $\{< t_1, t_2, ..., t_l >, < t_2, t_3, ...t_{l+1} >, ..., < t_{L-l+1}, t_{L-l+2}, ...t_L >\}$ in the Euclidean space $\mathbb{R}^l$. However, only one of these subsequences determines the best fit distance. We therefore ignore all subsequences of T_i except the one that is closest to the shapelet (Figure 1b). Suppose the offset of the closest subsequence is j. The closest subsequence

Algorithm 2: Classifying Samples with Learned Distance

```
   Input: Test Dataset $\mathbb{T}_{test} = [T_1, ..., T_n]$,
          Shapelet $Sh = sh_1, sh_2, ..., sh_l$,
          PCA Coefficients $PCA_{coeff}$,
          Parameter Coefficients $\Theta = [\Theta_0, \Theta_1, ...\Theta_{2m+1}]$
   Output: Hypothesis Vector $H = [h_1, h_2, ..., h_n]$
   /* Map each sample to $\mathbb{R}^l$ using its closest
      subsequence to $Sh$.                        */
 1 $\mathbb{E}_{map}$ = {}
 2 for i = 1 to $n = |\mathbb{T}_{test}|$ do
 3     $f_{Sh} : T_i \mapsto <t_j, t_{j+1}, ..., t_{j+m-1}>$
 4     $\mathbb{E}_{map}[i] = <t_j, t_{j+1}, ..., t_{j+m-1}>$
   /* Next, reduce dimensions through PCA from
      the training dataset.                        */
 5 $\mathbb{E}_{pca} = \mathbb{E}_{map}$ x $PCA_{coeff}$
   /* Finally, compute hypotheses using
      logistic regression learned on the
      shapelet and training data.                  */
 6 for i = 1 to $n = |\mathbb{T}_{Test}|$ do
 7     $<p_1, p_2, ..., p_m> = \mathbb{E}_{pca}[i]$
 8     $X = [1 + p_1 + p_1^2 + p_2 + p_2^2 + ... + p_m + p_m^2]$
 9     $H(i) = (1 + e^{-\Theta^T X})^{-1}$
10 Return $H$
```

is written as $t_j, t_{j+1}, ..., t_{j+l-1}$, and we map the time series T_i to a single point $<t_j, t_{j+1}, ..., t_{j+l-1}>$.

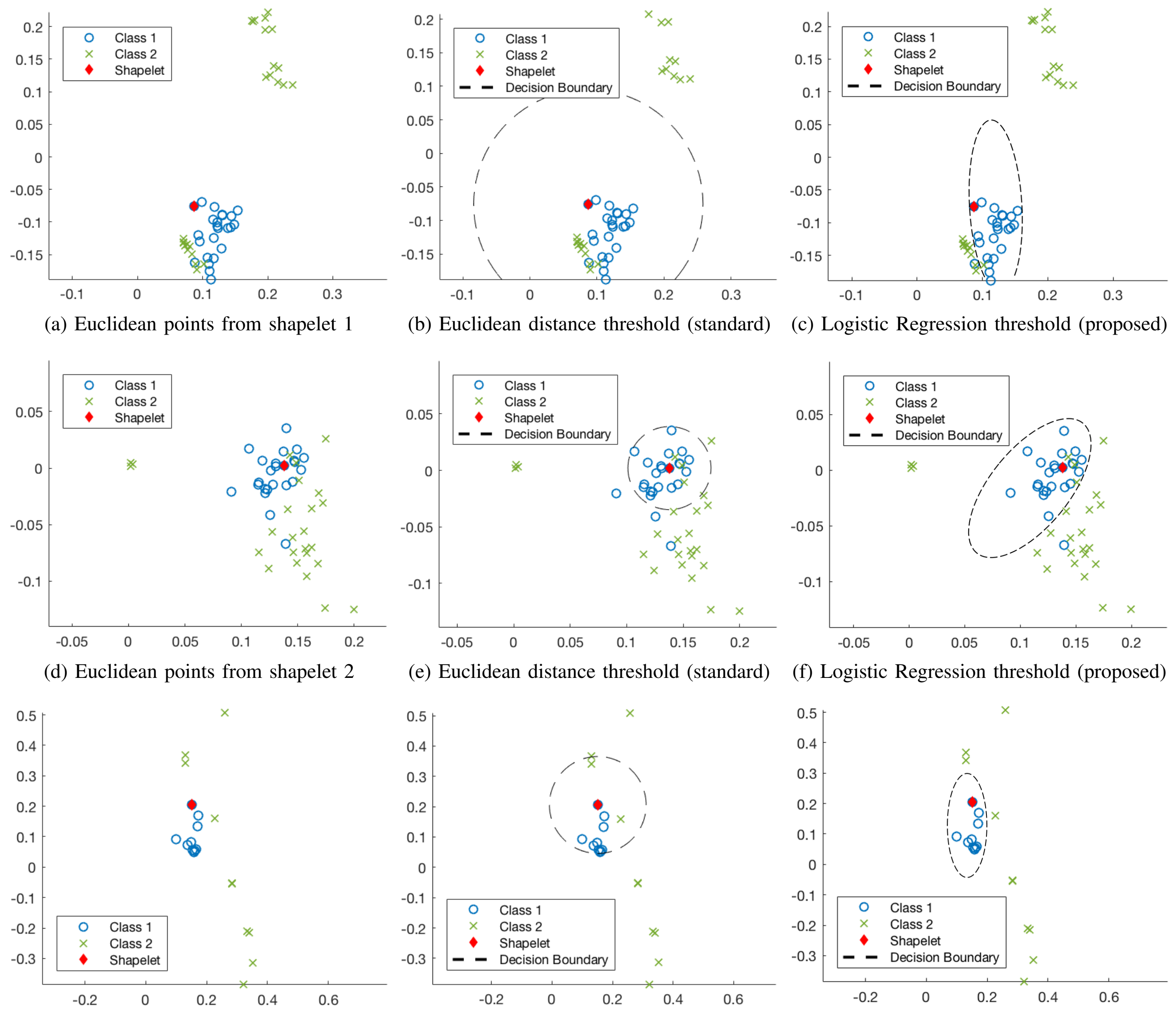
More formally, given a shapelet $Sh = sh_1, sh_2, ..., sh_l$ and a sample $T_i = t_1, t_2, ..., t_L$ whose closest subsequence to the shapelet is $t_j, t_{j+1}, ..., t_{j+l-1}$, the function f_{Sh} maps the time series T_i to the point $<t_j, t_{j+1}, ..., t_{j+l-1}>$ as in Equation 2.

$$f_{Sh} : T_i \mapsto <t_j, t_{j+1}, ..., t_{j+l-1}> \in \mathbb{R}^l \qquad (2)$$

Algorithm 1 describes this mapping scheme, and Figures 1b and 1c show an example using a shapelet of length 49 from the GunPoint dataset. Only 3 dimensions are shown, since the resulting Euclidean space has 49 dimensions, making fully viewing the Euclidean space impossible. Using f_{Sh} to map $\mathbb{T}_{train} = \{T_1, ..., T_n\}$ produces n points in $\mathbb{R}^l$, each of which has its own class label in $\mathbb{C}_{train}$, as shown in Figures 1d and 1e. We next use this mapping scheme to illustrate the limitations of computing a fixed Euclidean distance threshold.

B. Viewing a Euclidean Distance Threshold in $\mathbb{R}^m$

As mentioned in Section III, a shapelet is evaluated by information gain, which is the reduction in entropy after it performs a binary partition on the dataset. This partition is always determined by the optimal split point. In the projected Euclidean space shown in Figure 1e, partitioning using a the split distance d is mathematically equivalent to creating spherical decision boundary centered around the shapelet. This sphere is of the form $d^2 = \sum_{i=1}^{l}(x_i - sh_i)^2$ where the only parameter learned is d, the radius of the sphere. A projected sample inside the sphere is classified as class 1, while a sample outside of it is classified as class 2. Figure 2 shows this



(a) Euclidean points from shapelet 1 (b) Euclidean distance threshold (standard) (c) Logistic Regression threshold (proposed)

(d) Euclidean points from shapelet 2 (e) Euclidean distance threshold (standard) (f) Logistic Regression threshold (proposed)

(g) Euclidean points from shapelet 3 (h) Euclidean distance threshold (standard) (i) Logistic Regression threshold (proposed)

Fig. 2: Supervised learning of time series samples mapped to Euclidean space. The left-most images (a,d,g) each contain a shapelet (red) and closest subsequences for samples from class 1 (blue) and class 2 (green) mapped to $\mathbb{R}^m$ Euclidean space. PCA has been used to reduce the number of dimensions to 2 for visualization purposes. In this space, finding the optimal split distance for a shapelet is mathematically equivalent to having a spherical decision boundary centered around the shapelet, shown in the middle figures (b,e,h). The right-most images (c,f,i) show an elliptical boundary learned by logistic regression. Shapelets 1 and 2 in (a - f) are from the GunPoint dataset, while shapelet 3 in (g - i) is from BirdChicken.

boundary using 3 different shapelets. However, the training points in this space are rarely spherical in nature.

In the next section we propose an alternate decision boundary that is learned for each shapelet through logistic regression.

C. Learning a New Decision Boundary

Logistic regression classifies samples using the sigmoid hypothesis function $h(\Theta, X^{(i)}) = (1 + e^{-\Theta^T X^{(i)}})^{-1}$ where $X^{(i)}$ is a list of features for sample T_i and Θ is a vector of parameter weights for each feature. This function assigns to any point a probability that it has the same class label as the sample containing the shapelet. Assume, without loss of generality, that the sample containing the shapelet is from class 1. In binary classification, logistic regression uses a decision boundary at the threshold $h(\Theta, X^{(i)}) = 0.5$. If a sample creates a hypothesis $h(\Theta, X^{(i)}) \leq 0.5$, it is classified as class 1; otherwise it is classified as class 2 as shown in Figure 3.

The set of parameters Θ is learned according to the convex cost function in Equation 3 for $J(\Theta)$ below. Algorithm 1 describes the learning process.

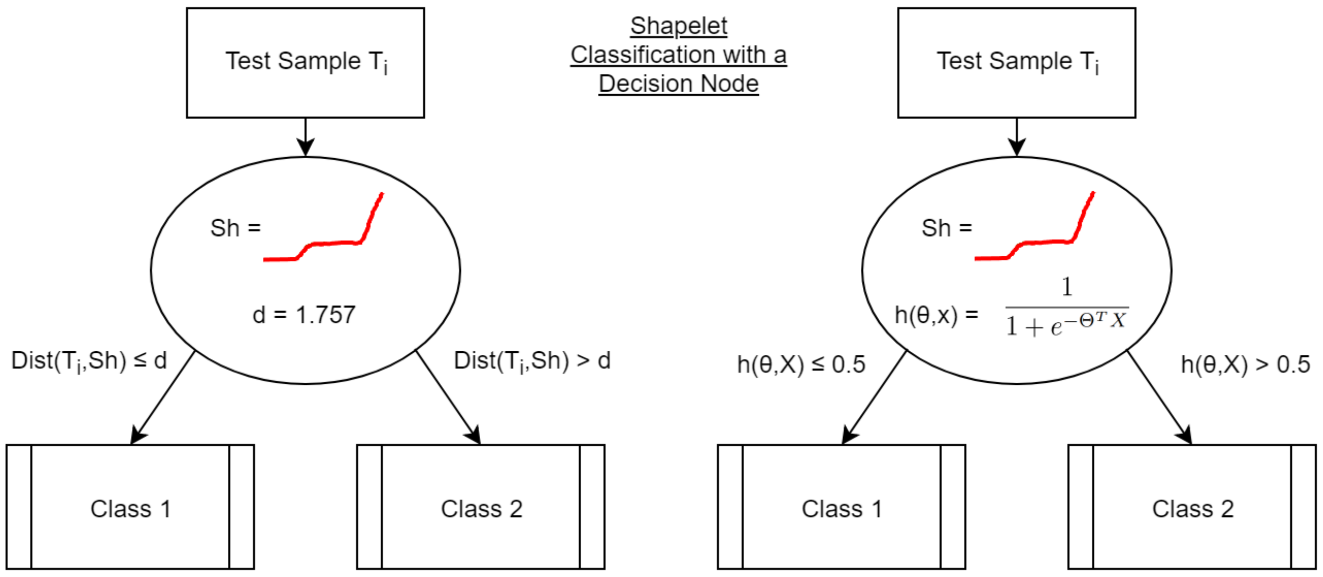


Fig. 3: Shapelet classification using a decision node. Standard practice finds the Euclidean distance that provides the optimal gain. This distance threshold d is then used to classify unlabeled samples (left). Instead, our proposed approach (right) uses logistic regression to learn a hypothesis function, providing a more complex boundary to model sample distributions.

$$J(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}[c_i log(h(\Theta, X^{(i)})) + (1-c_i)log(1-h(\Theta, X^{(i)}))] \quad (3)$$

Logistic regression could be used in the l-dimensional shapelet Euclidean space to separate samples instead of using a fixed Euclidean cutoff. However, since there are only n training samples in l dimensional space, this space can be very empty and contain excessive dimensions. As such, prior to performing logistic regression, we use PCA to reduce the number of dimensions while retaining at least $\nu\%$ of the data. The results described below were obtained using $\nu = 99\%$, which can significantly reduce overfitting of logistic regression without losing much of the data variance.

Instead of a sphere centered around the shapelet, we can learn a more flexible axis-aligned quadratic decision boundary of the form $X = [x_0, x_1, x_1^2, x_2, x_2^2, ..., x_m, x_m^2]$ with parameters $\Theta = [\Theta_0, \Theta_1, ...\Theta_{2m+1}]$, where m is the number of dimensions left after PCA.

We choose an axis-aligned decision boundary for two reasons. First, the decision to keep the quadratic function axis-aligned reduces overfitting by limiting the number of parameters that logistic regression must learn. Second, unlike a linear model, a quadratic equation has the ability to capture a cluster of points from class 1. This may not always be centered around the shapelet, but this structure can be captured by the X and Θ designated above.

Once training is complete, and unlabeled time series sample is mapped to the same Euclidean space and classified according to its hypothesis value, as described in Algorithm 2.

Mapping and Logistic Regression Variable Definitions	
Mapping Time Series to Euclidean points	
Variable	**Explanation**
f_{Sh}	Maps time series samples to points in $\mathbb{R}^l$ using the closest subsequences to the shapelet Sh
$\mathbb{E}_{map}$	The set of Euclidean points resulting from using f_{Sh} to map all samples in $\mathbb{T}_{train}$
$PCA()$	Principal Component Analysis; Reduces the number of dimensions of $\mathbb{E}_{map}$
ν	Cutoff for minimum PCA percent explained
PCA_{coeff}	Coefficient matrix produced by PCA
m	The number of dimensions needed to explain $\nu\%$ of the variance in $\mathbb{E}_{map}$
$\mathbb{E}_{pca}$	Euclidean points in PCA-reduced space Computed by $\mathbb{E}_{map}$ x PCA_{coeff}
Logistic Regression	
Variable	**Explanation**
X	List of features using the dimensions of $\mathbb{E}_{pca}$
Θ	Parameter coefficients for X
J	Cost function used to learn Θ using $\mathbb{E}_{pca}$
H	The hypothesis vector for the test samples

TABLE II: Logistic Regression Terminology

V. Experimental Setup

We use 2 different shapelet discovery and classification methods to compare classification accuracy using Euclidean distance versus accuracy with a learned distance. The first method, Fast Shapelets, is a search-based discovery method that uses decision trees for classification. The second method, FLAG, is a learning-based discovery method paired with SVM for classification. Performance for both of these methods is compared using multiple datasets from the UCR/UEA archive.

A. Fast Shapelets

Fast Shapelets [10], is a brute force search-based shapelet discovery method. Prior to discovery, Fast Shapelets first uses SAX word mapping [20], [21] to compress time series training samples. Fast Shapelets then finds the top k shapelets in the SAX space, maps them back to their original time series, and chooses the best shapelet using information gain. This approach makes Fast Shapelets the fastest search-based shapelet discovery method available. For classification, Fast Shapelets builds a decision node consisting of the discovered shapelet and its Euclidean cutoff distance. This process is executed recursively to build a decision tree for classification.

Decision trees are well known to propagate classification errors to successive levels of the tree. To avoid this error propagation, we focus only on the root node's shapelet, which is the single best shapelet found by Fast Shapelets using the entire training dataset. Figure 3 illustrates this comparison.

B. Fused LAsso Generalized eigenvectors (FLAG)

Fused LAsso Generalized eigenvectors (FLAG) is significantly different from Fast Shapelets since it frames shapelet discovery as an optimization problem and then learns the best shapelets. Its combination of the Generalized Eigenvector Method (GEM) [22] with a Fused Lasso regularizer [23] encourages a sparse, blocky solution and makes FLAG scalable. FLAG has reported runtimes that are faster than Fast Shapelets. It also has higher reported accuracy than Fast Shapelets since it learns k shapelets simultaneously and creates a Shapelet Transform [12] which is input into a support vector machine (SVM) for classification [17] as in Table III.

C. UCR/UAE Time Series Archive

As a testbed, we use the diverse and well-cited UCR/UAE data archive[1,2] [24], [25]. While this archive contains many datasets, we focus our results on the 30 datasets containing 2 classes to compare the impact of logistic regression versus Euclidean distance on individual shapelets. These 30 datasets were produced by 6 different authors and are from dozens of different domains. Each dataset has disjoint training and test datasets, with the training dataset containing between 20 to 1800 training samples. The length of each time series ranges from as few as 24 data points to over 600, but the lengths of all time series for a single dataset are the same.

VI. Results

A. Fast Shapelets

Tables IV and V show the results by dataset for using logistic regression versus the standard Euclidean decision threshold. Each entry is the average accuracy for 30 runs of Fast Shapelets using the single-best shapelet found. Table IV contains all of the UCR 2-class datasets with 100 or more training samples, while Table V has the 2-class datasets with less than 100 training samples. Results that were statistically

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

[2] http://timeseriesclassification.com/dataset.php

Euclidean Distance Shapelet Transform

	T_1	...	T_n
Sh_1	7.23	...	2.15
Sh_2	1.52	...	3.58
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sh_k	6.88	$\cdots$	3.23

Logistic Regression Shapelet Transform

	T_1	...	T_n
Sh_1	0.17	...	0.15
Sh_2	0.62	...	0.86
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sh_k	0.12	$\cdots$	0.04

TABLE III: An example of Shapelet Transforms used by FLAG. The standard practice is to use Euclidean distance $\in \mathbb{R}^+$(left). Instead, the hypotheses probabilities $\in [0, 1]$ produced by logistic regression can be used as a learned distance (right).

better with a p-value of $p \leq 0.05$ are shown in bold. Datasets with no bolded values had no statistically significant difference between the two methods.

For datasets containing 100 or more training samples in Table IV, logistic regression provided statistically better performance in 8 of the 13 datasets. Results from two of the datasets, Computers and Ham, were not statistically different. Euclidean distance was statistically better on 3 of the datasets, with small improvements on two datasets and a large improvement of 10.5% on Wafer. Overall, logistic regression increased accuracy an average of 3.8% across all datasets with 100 or more samples.

For datasets with less than 100 samples, Euclidean distance tended to perform slightly better than logistic regression as Table V shows. Of these 17 datasets, Euclidean distance performed statistically better on 8, logistic regression was statistically better on 4 datasets, and 5 datasets had no statistical difference. Compared to Euclidean distance, logistic regression decreased accuracy an average of 1.2% across all 17 datasets with under 100 samples.

The number of training samples influences which approach provides higher accuracy since logistic regression learns multiple parameters for its decision boundary while Euclidean distance learns only one. However, four additional factors impact the performance of each method.

First, some shapelets are longer than others, and longer shapelets create more parameters for logistic regression. Second, some datasets are imbalanced, meaning they have a disproportionate number of samples for each class. Third, some datasets contain test samples that do not have representative samples in the training dataset. Finally, the distribution of samples can vary significantly between classes.

As an example, the Wafer dataset contains sensor data from manufactured silicon wafers. All 30 runs of Fast Shapelets discovered the same shapelet of length 5, a relatively short shapelet. This is further reduced down to 3 dimensions using PCA, which means logistic regression is expected to perform better, especially since there are 1,000 training samples.

However, Wafer's training dataset is extremely imbalanced because it contains 903 Normal samples and only 97 Abnormal samples. Normal samples all come from sensors measuring

Logistic Regression vs. Euclidean Distance on Fast Shapelets			
Dataset Name	**# Samples**	**Log. Reg.**	**Euclidean**
Computers	250	62.6%	63.3%
DistalPhalanxOutline	276	74.2%	**76.3%**
ECG200	100	**81.5%**	70.9%
Earthquakes	139	**71.8%**	59.4%
FordA	1320	**81.6%**	74.4%
FordB	810	**81.9%**	72.6%
Ham	109	63.8%	63.2%
MiddlePhalanxOutline	291	**52.0%**	47.2%
PhalangesOutlines	1800	66.9%	**67.6%**
ProximalPhalanxOutline	600	**81.0%**	71.3%
Strawberry	370	**93.7%**	86.8%
Wafer	1000	89.3%	**99.8%**
Yoga	300	**61.7%**	60.3%

TABLE IV: Average fast shapelet accuracy on 2-class UCR datasets with 100 or more samples. For the single best shapelet found, logistic regression increased accuracy on 8 datasets, decreased accuracy on 3 datasets, and was statistically no different on 2 datasets (Computers and Ham).

Logistic Regression vs. Euclidean Distance on Fast Shapelets			
Dataset Name	**# Samples**	**Log. Reg.**	**Euclidean**
BeetleFly	20	79.5%	77.0%
BirdChicken	20	**82.8%**	74.3%
Coffee	28	**93.7%**	92.6%
ECGFiveDays	23	98.4%	**99.8%**
GunPoint	50	90.1%	**93.4%**
Herring	64	**54.7%**	50.5%
ItalyPowerDemand	67	64.0%	**69.4%**
Lightning2	60	59.3%	60.5%
MoteStrain	20	75.1%	**81.5%**
ShapeletSim	20	97.6%	**100%**
SonyAiboRoboSurfaceI	20	77.5%	**92.0%**
SonyAiboRoboSurfaceII	27	83.3%	81.1%
ToeSegmentationI	40	91.4%	90.4%
ToeSegmentationII	36	69.0%	70.4%
TwoLeadECG	23	**98.2%**	93.9%
Wine	57	73.6%	**77.3%**
WormsTwoClass	77	62.8%	**69.0%**

TABLE V: Average fast shapelet accuracy on 2-class UCR datasets with under 100 samples. Due to limited samples, logistic regression decreased accuracy on 8 datasets, but it was statistically no different on 5 datasets and it increased accuracy on 4 datasets.

Logistic Regression vs. Euclidean Distance on FLAG			
Dataset Name	**# Samples**	**Log. Reg.**	**Euclidean**
Coffee	28	100%	100%
ECGFiveDays	23	81.7%	92.0%
GunPoint	50	96.0%	93.3%
ItalyPowerDemand	67	94.8%	94.6%
MoteStrain	20	88.2%	81.8%
SonyAiboRoboSurfaceI	20	65.1%	92.8%
TwoLeadECG	23	99.7%	99.0%

TABLE VI: Deterministic FLAG results using logistic regression vs. Euclidean distance. Despite all datasets having fewer than 100 training samples, logistic regression still provided improvements on 4 datasets.

manufactured silicon wafers that have no defects. These samples have very low variation, meaning all the Normal samples occur in a very tight region. In contrast, Abnormal samples consist of defective wafers which exhibit a wide variety of behaviors. It is difficult to capture all possible Abnormal behaviors, particularly with so few training samples from the Abnormal class. This is an example of anomaly detection where Normal samples are very tightly clustered, a natural application for a Euclidean distance based method.

In summary, while multiple factors influence the classification accuracy, logistic regression tends to improve performance when given a sufficient number of samples since its decision boundary has the capability to capture different sample distributions. When there are few training samples, Euclidean distance typically performs slightly better since it only learns a single parameter.

B. Fused LAsso Generalized eigenvectors (FLAG)

Table VI displays the results of FLAG using distances learned through logistic regression compared to Euclidean distance. Each entry is the accuracy produced by FLAG, which is deterministic. We focus results on the 7 datasets listed, since FLAG's code requires parameters to be tuned to specific datasets, and parameter values tuned to the training samples for these datasets are available in the original author's code[3].

Given the low number of training samples for these 7 datasets, we expected Euclidean distance to outperform logistic regression, but the results are more mixed. As Table VI shows, logistic regression achieves the same accuracy on the Coffee dataset as Euclidean distance, decreases accuracy on two datasets (ECGFiveDays and SonyAIBORobotSurfaceI), and marginally improves accuracy on 4 datasets.

In addition to the characteristics that influence accuracy listed in Section VI-A, FLAG's mixed behavior is also caused by its simultaneous use of multiple shapelets. By using a Shapelet Transform (see Table III) with SVM, FLAG's classification scheme is more robust than Fast Shapelets and can mask impact of a single shapelet.

For instance, consider the Coffee dataset. Coffee contains time series spectrographs for two different types of coffee beans. FLAG discovers 98 shapelets containing between 14 to 43 data points. After PCA, longer shapelets can create more parameters than the 28 training samples, which causes some overfitting by logistic regression. However, many shorter shapelets have clearer separation between samples. While the performance using each individual shapelet varies between the two distance metrics, using all 98 shapelets together to train SVM results in 100% accuracy for both distance metrics.

C. Predicting Performance

In addition to evaluating both the Euclidean and logistic regression approaches using accuracy, it is also useful to be

[3]https://github.com/houlu369/FLAG_shapelets/tree/master/functions

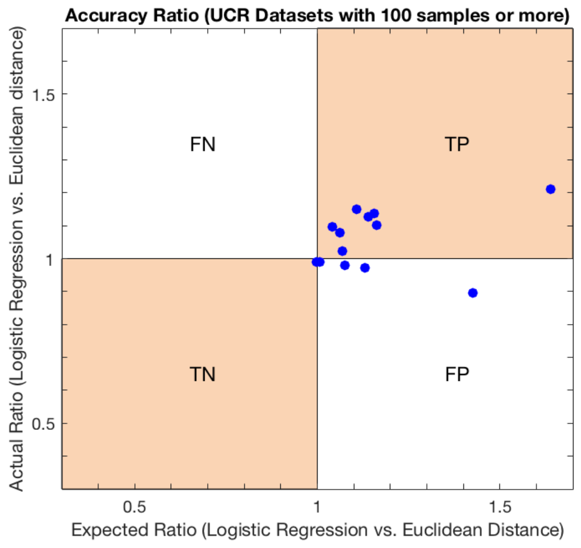


Fig. 4: Predictions for shapelet accuracy using logistic regression vs. Euclidean distance for datasets containing 100 training samples or more. Each blue point corresponds to a 2-class UCR dataset containing listed in Table IV. Due to the sufficient sample size, comparative performance is largely predictable using only training data.

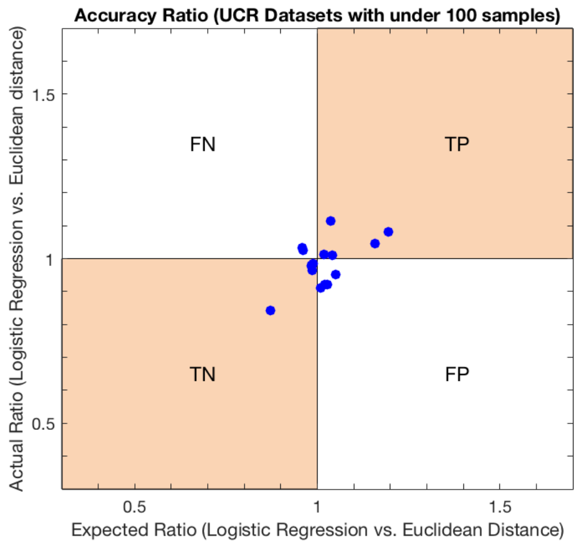


Fig. 5: Predictions for shapelet accuracy using logistic regression vs. Euclidean distance for datasets with less than 100 training samples. Each blue point corresponds to a 2-class UCR dataset containing listed in Table V. Comparative performance is more difficult to predict given the limited number of training samples.

able to predict which method will perform better using only the training dataset. For this, we use the $Expected\ Ratio$, introduced in [26] and also used by [10].

The expected ratio uses two-fold cross validation on the training dataset to compute the *Expected Accuracy* of both methods. First, the training dataset is partitioned in two with each partition containing the same number of samples for each class. For each class c_i, we placed the first $\frac{|c_i|}{2}$ samples from class c_i into the first partition and the remaining samples in the second partition. This ensures each partition has nearly the same number of samples for each class.

Each approach is trained on one partition and tested on the other. The two resulting accuracies are averaged, producing the expected accuracy. The expected accuracies are then used to computed the expected ratio, shown in the equation below.

$$Expected\ Ratio = \frac{Expected\ Accuracy\ (Log.\ Reg.\ Dist.)}{Expected\ Accuracy\ (Eucl.\ Dist.)}$$

The expected ratio is then compared to the $Actual\ Ratio$, obtained by training each method using the entire training dataset and testing on the test dataset, as defined in the following equation.

$$Actual\ Ratio = \frac{Actual\ Accuracy\ (Log.\ Reg.\ Dist.)}{Actual\ Accuracy\ (Eucl.\ Dist.)}$$

Figures 4 and 5 show the predictability of each method. The x-axis shows the Expected Ratio (i.e. which method was expected to perform better), while the y-axis shows the Actual Ratio (which method *did* perform better). Each blue point corresponds to a single dataset, with datasets containing 100 training samples or more shown in Figure 4 (listed in Table IV), while Figure 5 shows datasets with under 100 training samples (those listed in Table V). We only perform the Expected Ratio prediction on Fast Shapelets, since the tuned parameters for FLAG were obtained using all samples from each entire training dataset [17].

The actual and expected ratios partition the 30 datasets from the UCR data archive using Fast Shapelets into four quadrants: true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

- **True Positives (TP):** Datasets in this category had $Expected\ Ratio > 1$ and $Actual\ Ratio > 1$, meaning better classification was expected with logistic regression, and it *was* better.
- **True Negatives (TN):** Datasets in category had $Expected\ Ratio < 1$ and $Actual\ Ratio < 1$, meaning better results were expected with Euclidean distance, and that prediction was correct.
- **False Negatives (FN):** Datasets in this category had $Expected\ Ratio < 1$ and $Actual\ Ratio > 1$. This means Euclidean distance was expected to perform better, but instead, logistic regression provided higher accuracy.
- **False Positives (FP):** Finally, datasets in this category had $Expected\ Ratio > 1$ and $Actual\ Ratio < 1$, meaning logistic regression was expected to perform better, but Euclidean distance actually did.

In short, the shaded quadrants (TP and TN) show correct predictions where the Expected and Actual Ratios are either both above 1 or both below 1, while the white areas (FP and FN) show incorrect predictions.

Figure 4 shows the 13 UCR datasets that have at least 100 training samples. All 8 of the datasets where logistic performed better were correctly predicted, but only 1 dataset where Euclidean distance performed better was predicted.

The predictions for datasets containing under 100 samples are shown in Figure 5. As expected, the accuracy was difficult to predict. As logistic regression attempts to learn $2m + 1$ parameters, where $m < l$ (the length of the shapelet), this can often exceed the number of training samples, particularly for the smallest datasets which have as few as 20 training samples. This overfitting makes it difficult to predict the accuracy of logistic regression and also produces the decrease in accuracy for the majority of datasets in Table V.

VII. Conclusions and Future Work

This paper has introduced an approach to learning shapelet-specific custom distances through two steps. First, given a shapelet, the proposed mapping translates time series samples to Euclidean space using each sample's closest subsequence to the shapelet. This space is then reduced through principal component analysis and logistic regression learns a distance metric custom to the shapelet. Results show that shapelet classification under this approach is more accurate than Euclidean distance when given sufficient training samples.

Future work should focus on improving logistic regression's performance on datasets with limited training samples. This could be accomplished by adding a regularizer to reduce overfitting and experimenting with different types of functions for decision boundaries. In addition, minority oversampling techniques such as SMOTE [27] could be applied to datasets with limited samples or datasets that suffer from class imbalances. Other classifiers could also be applied to the Euclidean space following the mapping scheme described in Section IV-A to further improve shapelet classification accuracy.

References

[1] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[2] O. Al-Jowder, E. Kemsley, and R. H. Wilson, "Detection of adulteration in cooked meat products by mid-infrared spectroscopy," *Journal of agricultural and food chemistry*, vol. 50, no. 6, pp. 1325–1329, 2002.

[3] M. Biswal, Y. Hao, P. Chen, S. Brahma, H. Cao, and P. De Leon, "Signal features for classification of power system disturbances using PMU data," in *Power Systems Computation Conference (PSCC), 2016*. IEEE, 2016, pp. 1–7.

[4] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu, and E. Keogh, "A general framework for never-ending learning from time series streams," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1622–1664, 2015.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[6] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[7] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 1033–1040.

[8] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the ACM KDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 947–956.

[9] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[10] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proceedings of the International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[11] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: An expressive primitive for time series classification," in *Proceedings of the ACM KDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 1154–1162.

[12] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[13] J. Lines, L. M. 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*. ACM, 2012, pp. 289–297.

[14] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recently proposed algorithms," *Data Mining and Knowledge Discovery*, pp. 1–55, 2016.

[15] M. S. Cetin, A. Mueen, and V. D. Calhoun, "Shapelet ensemble for multi-dimensional time series," in *Proceedings of the International Conference on Data Mining*. SIAM, 2015, pp. 307–315.

[16] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the ACM KDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 392–401.

[17] L. Hou, J. T. Kwok, and J. M. Zurada, "Efficient learning of timeseries shapelets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[18] H. Zou, Y. Zhou, J. Yang, W. Gu, L. Xie, and C. Spanos, "Wifi-based human identification via convex tensor shapelet learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[19] J. Zakaria, A. Mueen, and E. Keogh, "Clustering time series using unsupervised-shapelets," in *Proceedings of the International Conference on Data Mining*. IEEE, 2012, pp. 785–794.

[20] L. Wei, E. Keogh, and X. Xi, "Saxually explicit images: Finding unusual shapes," in *Proceedings of the International Conference on Data Mining*. IEEE, 2006, pp. 711–720.

[21] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[22] N. Karampatziakis and P. Mineiro, "Discriminative features via generalized eigenvectors," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 494–502.

[23] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[24] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR time series classification archive," July 2015.

[25] A. Bagnall, J. Lines, W. Vickers, and E. Keogh, "The UEA and UCR time series classification repository," 2016, www.timeseriesclassification.com.

[26] S. L. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 317–328, 1997.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.