An Efficient and Accurate Time Series Classification Using Shapelets

M. Arathi and A. Govardhan

***Abstract*—Time series data are sequences of values measured over time. One of the most recent approaches to classification of time series data is to find shapelets within a data set. Time series shapelets are time series subsequences which represent a class. In order to compare two time series sequences using shapelets, existing work uses Euclidean distance measure. But Euclidean distance has following limitation : it requires data to be standardized if scales differ. In this paper, we perform classification of time series data using time series shapelets and used Mahalanobis distance measure. And also we have performed pessimistic pruning on decision tree. The Mahalanobis distance improves the accuracy of algorithm and pessimistic pruning method reduces the time complexity of testing and classification of unseen data. The Mahalanobis distance measure differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant. We show that our algorithm is much more accurate and faster than existing algorithms.**

***Index Terms*—Decision trees, information gain, mahalanobis distance measure, time series classification, shapelets, reduced error pruning.**

I. Introduction

Since a decade there have been enormous papers on time series classification. One of the most promising recent approaches is to find shapelets within a data set [1]. The shapelets are time series subsequences which represent a particular class. Algorithms that are based on shapelets are interpretable, more accurate and significantly faster than state-of-the-art classifiers [2], [3].

There are two types of classification algorithms: algorithms that consider whole (single) time series sequence (global features) for classification and algorithms that consider a portion of single time series sequence (local features) for classification. Shapelets are local features of the time series data. In classification by shapelets, a shapelet that represents a particular class is identified. And then, instead of comparing the entire time series sequence, only a small subsection of the time series (shapelets) from the two classes that is particularly discriminating are compared. Because shapelets are small in size compared to original data, algorithms that use shapelets for classification, results in less time and space complexity. Shapelets have also been used successfully in many other applications, such as early classification [4], gesture recognition [5] and as a filter transformation for TSC [6].

For classification with shapelets, decision trees (binary) are used, where each node represents a shapelet and leaf nodes represent class labels. To know how well the shapelet classified the data, information gain [7] is used. Apart from this, the other commonly used measures are such as the Wilcoxon signed-rank test [8], Kruskal-Wallis [9], Mood's Median [10] etc. The information gain/entropy measure is the better choice for two reasons. First, it can be easily generalized to the multiclass problem. Second, early entropy pruning can be done to avoid unnecessary distance calculations performed when finding the shapelet.

In classification of time series dataset using shapelets [1], Euclidean distance [11] has been used as similarity measure to compare two time series. There are some drawbacks of Euclidean distance measure. Firstly, it requires the time series data to be standardized, if scales differ. Secondly, it requires the two time series to be of same length. Thirdly, it does not take correlation of data items into consideration. To overcome some of the above problems, we have used Mahalanobis distance measure. It takes into account the correlations of the data items and is scale-invariant. In classification, the correlation among the dataset plays the key role. The accuracy has improved by using Mahalanobis distance measure instead of Euclidean distance measure.

To compare two time series data, a distance measure that is metric is used. A distance measure is said to be metric, if it satisfies following properties:1) $d(p, q) \geq 0$ for all p and q and $d(p, q)=0$ only if $p = q$. (Positive definiteness), 2) $d(p, q) = d(q, p)$ for all p and q. (Symmetry), 3) $d(p, r) \leq d(p, q) + d(q, r)$ for all points p, q, and r.(Triangle Inequality)

where $d(p, q)$ is the distance (dissimilarity) between points (data objects) p and q. Both the distance measures (Euclidean and Mahalanobis) are metric.

Some of the other distance measures are Dynamic Time Warping (DTW) [12], [13], distance based on Longest Common Subsequence (LCSS) [14], Edit Distance with Real Penalty (ERP) [15], Edit Distance on Real sequence (EDR) [16], DISSIM [17], Sequence Weighted Alignment model (Swale) [18], Spatial Assembling Distance (SpADe) [19] and similarity search based on Threshold Queries (TQuEST) [20].

The shapelet algorithm is embedded within the decision tree classifier. Whilst decision trees are highly interpretable, they have a tendency to overfit unless post-pruned or used with a conservative stopping condition. Furthermore, the recursive nature of the decision tree algorithm means that the relatively time-consuming shapelet detection method is called repeatedly. Hence, we have used pessimistic pruning technique on decision tree.

Manuscript received March 5, 2014; revised May 30, 2014.

M. Arathi and A. Govardhan are with Jawaharlal Nehru Technological University Hyderabad, Hyderabad-500085, Andhra Pradesh, India (e-mail: arathi.jntu@gmail.com).

Before time series data are compared, they must be normalized. Otherwise the results will be meaningless.

The rest of the paper is organized as follows. In Section II, we review related work. We define and compare distance measures in Section III. We discuss decision tree pruning in Section IV. We report our experimental results in Section V. We conclude our paper in Section VI.

II. Related Work

Here, we discuss various notations used in the algorithms and define various terms followed by the algorithm to find the shapelets and how they can be used in classification.

A. Notation

TABLE I: Notations

Symbol	Description
T, R	time series
S	subsequence
m, $\|T\|$	length of time series
l, $\|S\|$	length of subsequence
d	distance measure
D	time series dataset
A, B	class label
I	entropy
I_{WA}	weighted average entropy
sp	split strategy
k	number of time series objects in dataset
C	classifier
$S_{(k)}$	the k^{th} data point in subsequence S

Table I summarizes the notations used in the paper.

Some of the terms used in the algorithms and their definitions are:

Definition 1: *Time Series* : A time series $\mathbf{T} = t_1, \ldots, t_m$ is an ordered set of m real-valued variables.

Data points $t_1, \ldots, t_m$ are typically arranged by temporal order, spaced at equal time intervals.

Definition 2: *Subsequence* : Given a time series **T** of length m, a subsequence S of T is a sampling of length $l \leq m$ of contiguous positions from T, that is, $S = t_p, \ldots, t_{p+l-1}$, for $1 \leq p \leq m - l + 1$.

Definition 3: *Sliding Window* : Given a time series T of length m, and a user-defined subsequence length of l, all possible subsequences can be extracted by sliding a window of size l across **T** and considering each subsequence S_p^l of **T**. Here the superscript l is the length of the subsequence and subscript p indicates the starting position of the sliding window in the time series. The set of all subsequences of length l extracted from **T** is defined as S_T^l, $S_T^l = \{ S_p^l$ of **T**, for $1 \leq p \leq m - l + 1\}$.

Definition 4: *Distance between the time series* : Dist(T, R) is a distance function that takes two time series T and R which are of the same length as inputs and returns a nonnegative value d. It is also applicable to subsequences of same length.

Definition 5: *Distance from the time series to the subsequence*. SubseqDist(T, S) is a distance function that takes time series T and subsequence S as inputs and returns a nonnegative value d, which is the minimum possible distance from T to S. SubseqDist(T, S) = min(Dist(S, S')), for $S' \in S_T^{|S|}$.

Definition 6: *Entropy* : A time series dataset D consists of two classes, A and B. Given that the probability of a object belonging to class A is $p(A)$ and the probability of a object belonging to class B is $p(B)$. The entropy of D is defined as:

$$I(D) = -p(A)\log(p(A)) - p(B)\log(p(B)) \quad (1)$$

A splitting strategy is used which divides the whole dataset D into two subsets, D_1 and D_2. Therefore, the information remaining in the entire dataset after splitting is defined by the weighted average entropy of each subset. If the fraction of objects in D_1 is $f(D_1)$ and the fraction of objects in D_2 is $f(D_2)$, the total entropy of D after splitting is:

$$I_{WA}(D) = f(D_1)I(D_1) + f(D_2)I(D_2) \quad (2)$$

Definition 7: *Information Gain* : Given a certain split strategy sp, the entropy before and after splitting is $I(D)$ and $I_{WA}(D)$. So the information gain for this splitting strategy is:

$$\text{Gain}(sp) = I(D) - I_{WA}(D) \quad (3)$$

The distance to a shapelet is used as the splitting rule.

Definition 8: *Optimal Split Point (OSP)*: A time series dataset D consists of two classes, A and B. For a shapelet candidate S, some distance threshold d_{th} is choosen which splits D into D_1 and D_2, such that for every time series object $T_{1,i}$ in D_1, SubseqDist($T_{1,i}$, S) < d_{th} and for every time series object $T_{2,i}$ in D_2, SubseqDist($T_{2,i}$, S) $\geq d_{th}$. An Optimal Split Point is a distance threshold that:

$$\text{Gain}(S, d_{OSP(D,S)}) \geq \text{Gain}(S, d'_{th}) \quad (4)$$

for any other distance threshold d'_{th}.

So using the shapelet, the splitting strategy contains two factors: the shapelet and the corresponding optimal split point.

Definition 9: *Shapelet*: Given a time series dataset D which consists of two classes, A and B, shapelet(D) is a subsequence that, with its corresponding optimal split point,

$$\text{Gain}(\text{shapelet}(D), d_{OSP(D,\text{shapelet}(D))}) \geq \text{Gain}(S, d_{OSP(D,S)}) \quad (5)$$

for any other subsequence S.

The minimum and maximum lengths for shapelets were computed using the simple cross-validation approach [6].

B. Classification Using Shapelets

First, let us look at the algorithm for finding shapelets, and then classifying the time series data using the shapelets, and at last testing the accuracy of classifier.

The dataset D has two classes i.e. class A and class B. The algorithm in Fig. 1 takes the dataset D, maximum and minimum length of the shapelet as input and returns best shapelet. Line 1 generates the subsequences of all possible lengths, and stores them in the unordered list *subseq_list*. In line 2, the maximum information gain *max_gain* is initialized to zero. From lines 3 to 8, it checks how well each subsequence can separate objects into class A and class B. CheckSubseq() returns True if there is a high probability of S resulting in best shapelet. If it returns True, then information

gain is calculated. If the information gain is higher than *max_gain*, the algorithm updates the *max_gain* and the corresponding best shapelet *best_shapelet*. The subroutines GenerateAllSubseq() and CheckSubseq() are outlined in Fig. 2 and Fig. 3 respectively.

```
BestShapelet(dataset D, MAXLEN, MINLEN)
1. subseq_list ← GenerateAllSubseq(D, MAXLEN,
MINLEN)
2. max_gain ← 0
3. For each S in subseq_list
4.    obj_hist←null
5.    flg ← CheckSubseq(D, S,obj_hist)
6.    if flg is True
        gain←InfoGain(obj_hist)
        If gain > max_gain
7.          max_gain ← gain
8.          best_shapelet ← S
9.       EndIf
10. EndFor
11. Return best_shapelet
```

Fig. 1. Algorithm for finding shapelet.

```
GenerateAllSubseq(dataset D, MAXLEN, MINLEN)
1. list ←null
2. l ← MAXLEN
3. While l ≥ MINLEN
4.       For T in D
5.          list ← list ∪ S_T^l
6.       EndFor
7.       l←l - 1
8. EndWhile
9. Return list
```

Fig. 2. Generate all the possible subsequences.

In Fig. 2, line 1 initializes the subsequence list to null. In line 2, the shapelet length *l* is initialized to MAXLEN. From line 4 to 6, the algorithm slides a window of size *l* across all of the time series objects in the dataset D, extracts all of the possible candidates and adds them to the list. The above process is repeated for *l* values ranging from MAXLEN TO MINLEN. Finally, in line 9, it returns the list which is the set of all possible subsequences for given data set.

The algorithm for CheckSubseq() is shown in Fig. 3. It takes dataset *D*, subsequence *S* and objects histogram *objects_hist* as input and returns True if there is a high probability of S resulting in best shapelet.

```
CheckSubseq(dataset D, subsequence S, objects_hist)
1. For each T in D
2.    dist ← SubseqDist(T, S)
3.    insert T into objects_hist by the key dist
4.    flag←EntropyPrune(max_gain, objects_hist, C_A,
                  C_B)
5.    If flag is True
6.       Return False
7. EndFor
8. Return True
```

Fig. 3. Checking the subsequence.

From lines 2 to 6, it checks whether subsequence S will result in a best shapelet or not. In line 2, the distance from the time series T to the subsequence S is obtained by calculating the Euclidean distance of every subsequence of length |S| in T and S and choosing the minimum(Fig. 4). In line 3, it inserts all of the time series objects into the *object_hist* according to the distance from the time series object to the candidate. In line 4, it calls EntropyPrune() which returns True if the subsequence can be pruned, otherwise False.

Fig. 4 shows an optimization in computing distance between the time series *T* and subsequence *S*. Instead of computing the exact distance between every subsequence of *T* and the subsequence *S*, the distance calculations can be stopped once the partial computation exceeds the minimum distance known so far. This is known as early abandon [21].

```
SubseqDist(T, S)
1. min_dist ←∞
2. stop← False
3. For S_i in S_T^|S|
4.    dist ← 0
5.    For k← 1 to |S|
6.       dist ← dist + (S_i(k) – S_(k))^2
7.       If dist ≥ min_dist
8.             stop← True
9.             Break
10.         EndIf
11.   EndFor
12.   If not stop
13.      min_dist← dist
14.   EndIf
15 EndFor
16 Return min_dist
```

Fig. 4. Optimization in subsequence distance calculation by early abandon approach.

In line 1, the *min_dist* is initialized to infinity. Now, for each subsequence S_i from *T* of length |*S*|, the distance *dist* between S_i and *S* is accumulated, one data point at a time (line 6). Once *dist* is larger than or equal to the minimum distance known so far, the distance calculation between S_i and *S* is abandoned (lines 7 to 9). If the distance calculation between S_i and *S* is completed, then the distance is smaller than the minimum distance known so far. Thus, *min_dist* is updated in line 13. The algorithm returns the distance from the time series T to the subsequence S in line 16.

The computation of information gain is outlined in Fig. 5. The optimal split point for the object histogram is computed and stored in *split_dist*(line 1). From line 4 to 7, it divides the time series objects into two subsets by comparing the distance with *split_dist*. If the distance is less than *split_dist*, then the object is placed in D_1, otherwise in D_2. Finally, in line 10, it computes and returns the information gain.

```
InfoGain (distance histogram obj_hist)
1. split_dist← OptimalSplitPoint(obj_hist)
2. D1←null, D2 ←null
3. For d in obj_hist
4.       If d.dist < split_dist
5.          D1← D1 ∪ d.objects
6.       Else
7.          D2←D2 ∪ d.objects
8.       EndIf
9. EndFor
10. Return I(D) - I_WA(D)
```

Fig. 5. Information gain of distance histogram optimal split.

The entropy pruning is shown in Fig. 6. It tries to optimize the time complexity. The algorithm takes as the inputs the maximum information gain(till now), distance histogram and

the remaining time series objects in class A and B, and returns TRUE if the sequence can be pruned. The algorithm begins by finding the two ends of the histogram. For simplicity, the distance values at two ends are set as 0 and (maximum distance+1) as shown in lines 1 and 2. To build the optimistic histogram of the whole dataset based on the existing one (lines 3 and 8), the remaining objects of one class are assigned to one end and those of the other class to the other end (lines 4 and 9). If in either case, the information gain of the optimistic histogram is higher than the best so far (lines 5 and 10), it is still possible that the actual information gain of the candidate can beat the best so far. Thus, we should continue to test the candidate (lines 6 and 11). Otherwise, the remaining calculations with the candidate are pruned (line 13).

```
EntropyPrune (max_gain, dist_hist, c_A, c_B)
1. min← 0
2. max←( largest distance value in dist_hist) + 1
3. pred_hist ← dist_hist
4. Add to the pred_hist, c_A at minend and c_B at maxend
5. If InfoGain (pred_hist) > max_gain
6.    Return FALSE
7. EndIf
8. pred_hist← dist_hist
9. Add to the pred_hist, c_A at maxend and c_B at minend
10. If InfoGain (pred_hist) > max_gain
11.      Return FALSE
12. EndIf
13. Return TRUE
```

Fig. 6. Entropy prunning.

It is often the case that different candidates will have the same best information gain. This is particularly true for small datasets. Such ties can be broken by favoring the longest candidate, the shortest candidate or the one that achieves the largest margin between the two classes.

Classifying with a shapelet and its corresponding split point produces a binary decision as to whether a time series belongs to a certain class or not. Obviously, this is not enough to deal with a multi-class problem. In order to make the shapelet classifier universal, decision trees [22] are used. In decision tree induction, the shapelet and the corresponding split point are determined at each node. Once the decision tree is constructed, the accuracy of decision tree is computed using testing dataset (Fig. 7).

```
CalculateAccuracy ( decision tree classifier C, dataset D_t)
1. For each T in D_t
2.       predicted_label← Predict(C, T)
3        If predicted_label is the same as actual class label
4.          count← count + 1
5.       EndIf
6.EndFor
7. Return count / | D_t |
```

Fig.7. Accuracy of classifier.

The algorithm takes the decision tree classifier and testing dataset as input and returns the accuracy of the classifier. In line 2, the class label for the current time series data *T* is predicted and stored in *predicted_label*. If the predicted label is same as actual class label, then it is correctly classified and the number of correctly classified count is increased by 1(line 3 and 4). The above steps are repeated for all the time series data in test dataset D_t. In line 7, it returns the accuracy of the classifier.

Fig. 8 shows how to predict the class label of the time series data object. Each non leaf node of the decision tree has shapelet information, split point, the left subtree and the right subtree. Each leaf node of the decision tree specifies the predicted class label. The prediction process starts from the root node. The distance between the time series object *T* and the shapelet in the root node is computed and is compared with optimal split point. If the distance is less than the split point, then left branch is taken which leads to left subtree (lines 6 and 7) and otherwise the right branch taken which leads to right subtree (lines 8 and 9). The above procedure is repeated in the subtree. This procedure continues until the leaf node is reached which contains the class label information and that class label is returned to the calling function (lines 1 and 2).

```
Predict (decision tree classifier C, time series object T)
1.If C is the leaf node
2.    Return label of C
3.Else
4.    S← shapelet on the root node of C
5.    split_pt← split point on the root of C
6.    If SubseqDist (T, S) < split_pt
7.       Predict (left subtree of C, T)
8.    Else
9.       Predict (right subtree of C, T)
10.   EndIf
11.EndIf
```

Fig. 8. Predicting the class label of a testing object.

III. Proposed Method

The Similarity is a numerical measure of how alike two data objects are. It is higher when objects are more alike. It often falls in the range [0, 1]. Dissimilarity is numerical measure of how different are two data objects. It is lower when objects are more alike. Minimum dissimilarity is often 0. We use distance measure to compare two data objects. We show that using Mahalanobis Distance measure instead of Euclidean distance measure improves the accuracy of the algorithm.

A. Euclidean Distance

In mathematics, the *Euclidean distance or Euclidean metric* is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. By using this formula as distance, Euclidean space (or even any inner product space) becomes a metric space. It is defined as,

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2} \tag{6}$$

where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k^{th} attributes (components) of data objects p and q.

Here, standardization is necessary, if scales differ.

B. Mahalanobis Distance

The Mahalanobis distance is a descriptive statistic that provides a relative measure of a data point's distance (residual) from a common point. It is a unitless measure introduced by P. C. Mahalanobis in 1936 [23]. The Mahalanobis distance is used to identify and gauge *similarity* of an unknown sample set to a known one. It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant.

Given a time series $x^{(k)}$, let the i^{th} data point be $x_i^{(k)}$. We compute the(sample) covariance matrix $C = (c_{ij})$ of a family of time series $x^{(1)}$, $x^{(2)}$,…, $x^{(N)}$ of lengths n by $c_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(x_i^{(k)} - \bar{x}_i)(x_j^{(k)} - \bar{x}_j)$ where N is the number of instances and where $\bar{x}_i$ is the average of the i^{th} data point of the time series ($\bar{x}_i = \frac{1}{N}\sum_{k=1}^{N} x_i^{(k)}$).

The Mahalanobis distance measure is a special case of the generalized ellipsoid distance measure $D_M(x, y) = (x-y)^T M(x-y)$ where M is proportional to the inverse of the covariance matrix i.e., $M \propto C^{-1}$. Though the Mahalanobis distance measure is often defined by setting M to the inverse of the covariance matrix ($M = C^{-1}$), it is convenient to normalize it when possible so that the determinant of the matrix M is one: $M = (\det(c))^{\frac{1}{n}} c^{-1}$ where n is the length of the time series. The Mahalanobis distance measure minimizes the sum of distances between time series $\sum_{x,y} D_M(x, y)$ subject to a regularization constraint on the determinant ($\det(M) = 1$). In this sense, it is optimal.

When the covariance is non-singular ($\det(C) \neq 0$) then the covariance is positive definite, and so is the matrix M: it follows that the square root of the generalized ellipsoid distance measure is a metric. That is, we have $D_M(x, y) = 0 \Leftrightarrow x = y$, it is symmetric, non-negative and it satisfies the triangle inequality $\sqrt{D_M(x, z)} + \sqrt{D_M(z, y)} \geq \sqrt{D_M(x, y)}$.

We have used Mahalanobis distance measure instead of Euclidean Distance measure due to its merits over Euclidean distance as shown in following subsection.

C. Euclidean vs Mahalanobis

The Mahalanobis distance takes the co-variances into account, which lead to elliptic decision boundaries in the 2D case, as opposed to the circular boundary in the Euclidean case. The Euclidean distance may be seen as a special case of the Mahalanobis distance with equal variances of the variables.

The Mahalanobis distance is a fine way to reduce linear correlation and some scaling, so if one is looking at distance and has enough data, it makes more sense than Euclidean. In statistics, sometimes the nearness or farness is measured in terms of the scale of the data. Often scale means standard deviation. For univariate data, an observation that is one standard deviation away from the mean is closer to the mean than an observation that is three standard deviations away.

The graph in Fig. 9 shows simulated bivariate normal data that is overlaid with prediction ellipses. The ellipses in the graph are the 10% (innermost), 20%, and son on till 90% (outermost) prediction ellipses for the bivariate normal distribution that generated the data. The prediction ellipses are contours of the bivariate normal density function. The probability density is high for ellipses near the origin, such as the 10% prediction ellipse. The density is low for ellipses are further away, such as the 90% prediction ellipse.

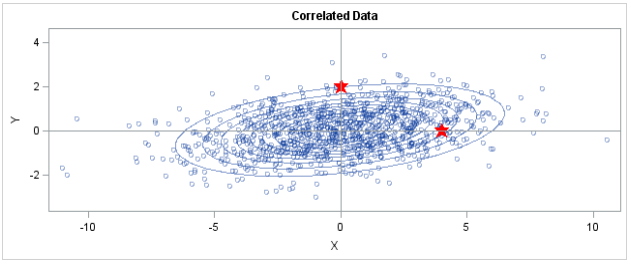

Fig. 9. Bivariate normal data with predicted ellipses.

In the graph, two observations are displayed by using red stars as markers. The first observation is at the coordinates (4, 0), whereas the second is at (0, 2). To see which mark is closer to origin, let us consider the two distance measures. The Euclidean distances are 4 and 2, respectively. Hence, according to Euclidean distance measure, the point at (0, 2) is closer to the origin. However, for this distribution, the variance in the Y direction is less than the variance in the X direction, so in some sense the point (0, 2) is more standard deviations away from the origin than the point (4, 0).

Notice the position of the two observations relative to the ellipses. The point (0, 2) is located at the 90% prediction ellipse, whereas the point at (4, 0) is located at about the 75% prediction ellipse. It means that the point at (4, 0) is closer to the origin in the sense that you are more likely to observe an observation near (4, 0) than to observe one near (0, 2). The probability density is higher near (4, 0) than it is near (0, 2). Hence, according to Mahalanobis distance, the point at (4, 0) is closer to origin than the point at (0, 2).

In this sense, prediction ellipses are a multivariate generalization of units of standard deviation. The bivariate probability contours can be used to compare distances to the bivariate mean. A point p is closer than a point q if the contour that contains p is nested within the contour that contains q.

The Mahalanobis distance has the following properties: 1) It accounts for the fact that the variances in each direction are different. 2) It accounts for the covariance between variables. 3) It reduces to the familiar Euclidean distance for uncorrelated variables with unit variance.

IV. Decision Tree Pruning

In decision tree induction process, if we have a tightly stopping criteria, it will lead to small and underfitted decision trees. On the other hand, if we have loosely stopping criteria, it will lead to generate large decision trees that are overfitted to the training set. Many pruning methods have been introduced to solve later problem [22]. In decision tree pruning, the overfitted tree is cut back into a smaller tree by identifying and removing those subtrees that will not reduce

the accuracy of the decision tree. That is, given a decision tree classifier C and an inner (non-root, non-leaf) node t. Then pruning of C with respect to t is the deletion of all successor nodes of t in C which makes t a leaf node. This process is repeated on all nonleaf nodes. Hence, it leads to a smaller and accurate decision tree.

A. Pessimistic Pruning

The pessimistic pruning method uses the pessimistic statistical correlation test [24]. Here, the continuity correction for binomial distribution is used:

$$\varepsilon'(T,S) = \varepsilon(T,S) + \frac{|\,leaves(T)\,|}{2.|\,S\,|} \quad (7)$$

where $\varepsilon(T,S)$ indicates the error rate of the tree T over the sample S and |leaves(T)| denotes the number of leaves in T. The internal node t should be pruned only if its error rate is within one standard error from a reference tree, namely:

$$\varepsilon'(pruned(T,t),S) \le \varepsilon(T,S) + \sqrt{\frac{\varepsilon'(T,S).(1-\varepsilon'(T,S))}{|\,S\,|}} \quad (8)$$

where $pruned(T, t)$ denotes the tree obtained by replacing the node t in T with a suitable leaf.

The last condition is based on statistical confidence interval for proportions. Usually the last condition is used such that T refers to a subtree whose root is the internal node t and S denotes the portion of the training set that refers to the node t.

The pessimistic pruning procedure performs top–down traversing over the internal nodes. If an internal node is pruned, then all its descendants are removed from the pruning process, resulting in a relatively fast pruning. We have performed pessimistic pruning on decision tree as it is fast and results in fast classification.

V. Experimental Results

The experiments are conducted on standard datasets such as wheat, mallet, coffee, gun, projectile points, historical documents, beef, car etc. [25]. On all the datasets, our proposed method has shown around 10% to 15% increase in accuracy and 20% to 25 % increase in speed.

The wheat dataset consists of 775 spectrographs of wheat samples grown in Canada between 1998 and 2005. There are different types of wheat, such as Soft White Spring, Canada Western Red Spring, Canada Western Red Winter, etc. The wheat dataset composes of all the above mentioned wheat types. The class label given for this problem is the year in which the wheat was grown. For this dataset, our method has shown 12% increase in the accuracy due to Mahalanobis distance as shown in Fig. 10. And there was 20% increase in speed (due to decision tree pruning) during testing phase and classification of unseen data.

There has been extensive study on Gun/NoGun motion capture time series dataset [2], [26]. This data has two classes: gun and no gun. The classification algorithm should be able to identify whether the actor is holding gun or not. The difference between the two classes can be identified if we observe the time series data of the actor the way he/she puts his/her hand down by his/her side. Our method has shown 8% increase in accuracy for Gun/NoGun problem due to Mahalanobis distance as shown in Fig. 11. And there was 16% improvement in speed (due to decision tree pruning) during testing phase and during classification of unseen data. Hence, the proposed method is more accurate and fast than existing method.

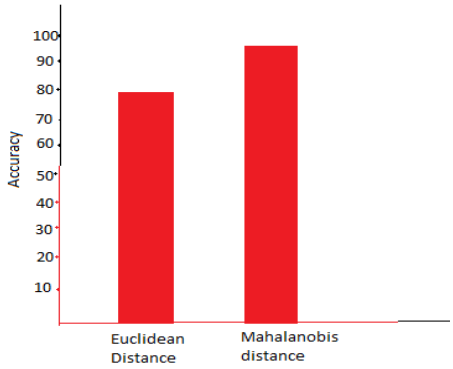

Fig. 10. Accuracy for wheat dataset using Euclidean *vs* Mahalanobis distance.

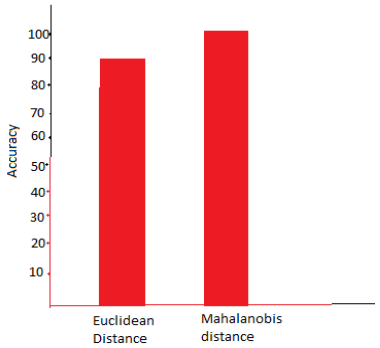

Fig. 11. Accuracy for Gun/NoGun dataset using Euclidean *vs* Mahalanobis distance.

VI. Conclusion

The classification of the time series dataset is performed using shapelets. The shapelets are time series subsequences and are highly representative of a class. Because one shapelet is not sufficient to classify the data, we have used a number of shapelets which clearly distinguishes one class from other. The shapelets are used along with distance threshold, which divides the data into two sets. The decision trees are used for classification. The non leaf nodes of the decision tree specify shapelet and distance threshold; and leaf nodes specify the class label. To classify a time series data, it is fed into decision tree classifier, which moves it from root node to leaf node, which in turn gives the predicted class label. While moving from root to leaf node, the time series data is compared with every shapelet on the path using Mahalanobis distance measure. The Mahalanobis distance measure is a good choice for classification as it takes the correlation of data items into consideration and is scale in-variant. Hence, it is obvious that Mahalanobis distance measure will give more accurate results. We have also shown with experiments that the distance measure results in more accuracy than the

Euclidean distance measure. And we have performed pessimistic pruning on decision tree. The pruning method reduces the size of the decision tree which leads to reduction in time taken in testing phase and also in classification of unseen data. In future, we are going to compare it with other distance measures. We are also going to check how the algorithm will perform on reduced representation of time series dataset. We also wish to do signature verification using the proposed method.

REFERENCES

[1] L. X. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, June 29–July 1, 2009, pp. 947-956.

[2] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," in *Proc the 34th VLDB*, 2008, pp. 1542–1552.

[3] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," in *Proc. the 8th ACM SIGKDD*, 2002, pp. 102-111.

[4] P. Yu, K. Wang, Z. Xing, and J. Pei, "Extracting interpretable features for early classification on time series," presented at 11th SDM

[5] B. Hartmann and N. Link, "Gesture recognition with inertial sensors and optimized DTW prototypes," in *Proc. IEEE SMC*, Istanbul, 2010, pp. 2102-2109.

[6] J. Lines, L. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," *Tech. Report*, University of East anglia, UK, 2012.

[7] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed., Elsevier Publisher, 2008, ch. 6, pp. 297-300.

[8] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80-83, 1945.

[9] W. H. Kruskal, "A Nonparametric test for the several sample problem," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 525–540, 1952.

[10] A. M. F. Mood, *Introduction to the Theory of Statistics*, 1950.

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time series databases," in *Proc. SIGMOD Conference*, 1994, vol.23, no.2, pp. 419-429.

[12] P. Geurts, "Pattern extraction for time series classification," in *Proc. the 5th PKDD*, 2001, pp. 115-127.

[13] E. J. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time wraping," *Knowl. Inf. Syst*., vol.7, no. 3, 2005.

[14] D. Gunopulos, and G. Kollios, "Discovering similar multidimensional trajectories," in *Proc. ICDE*, 2002, pp. 673-684.

[15] L. Chen and R. T. Ng, "On the marriage of Lp-norms and edit distance," in *Proc. VLDB*, 2004, pp. 792-803.

[16] L. Chen, M. T. Dzsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. SIGMOD Conference*, 2005, pp. 491-502.

[17] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *Proc. ICDE*, Istanbul, 2007, pp. 816-825.

[18] M. D. Morse and J. M. Patel, "An efficient and accurate method for evaluating time series similarity," in *Proc. SIGMOD Conference*, 2007, pp. 569-580.

[19] Y. Chen, M. A. Nascimento, B. C. Oosi, and A. K. H. Tung, "SpADe: On Shape-based pattern detection in streaming time series," in *Proc. ICDE*, 2007, Istanbul, pp. 786-795.

[20] J. Abflag, H.-P. Kriegel, P. Kreger, P. Kunath, A. Pryakhin, and M. Renz, "Similarity search on time series based on threshold queries," in *Proc. EDBT*, 2006, pp. 276-294.

[21] E. Keogh, L. Wei, X. Xi, S. Lee, and M. Vlachos, "LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," in *Proc. 32nd VLDB*, 2006, PP. 882-893.

[22] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[23] P. C. Mahalanobis, "On the generalized distance in statistics," in *Proc. the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[24] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221-234, 1987.

[25] Time_series_data. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/

[26] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using Numerosity reduction," in *Proc. the 23rd ICML*, 2006, pp. 1033-1040.

M. Arathi was born in Hyderabad, Andhra Pradesh, INDIA on 8th October 1979. She pursued B.E.(CSE) from MVSREC, Hyderabad, Andhra Pradesh, India, in 2001; M.Tech(CS), JNTUH, Hyderabad, Andhra Pradesh, India, in 2008. Her major field of study is data mining.

She has worked as an assistant professor in Sant Samarth Engineering College, Hyderabad, Andhra Pradesh for 11 months. Next, she is working as assistant professor in JNTUH, Hyderabad, Andhra Pradesh. It is more than 10 years since she has been with JNTUH, Hyderabad, Andhra Pradesh. She has many publications in journals, international and national conferences.

Mrs. Arathi is an expert committee member in Institute for Innovations in Science and Technology. She has been judge for many paper presentation contests in JNTUH. She has been subject expert for QTP testing tool.

A. Govardhan was born in Nalgonda, Andhra Pradesh, INDIA, on 10th March 1970. He pursued B.E.(CSE) from Osmania University, Hyderabad, Andhra Pradesh in 1992; M.Tech(CS) from JNU, New Delhi, India in 1994; Ph.D(CS) from JNTU, Hyderabad, Andhra Pradesh in 2003. His areas of research include Databases, Data Mining and Information Retrieval Systems.

He is presently a director at SIT and executive council member at Jawaharlal Nehru Technological University Hyderabad (JNTUH), India. He has 2 monographs and has guided 125 M.Tech projects, 20 Ph.D thesis and has published 152 research papers at journals/conferences including *IEEE, ACM, Springer, Elsevier and Inder Science*. Delivered more than 50 Keynote addresses. He held several positions including director of evaluation, principal, HOD and students' advisor.

Prof. A. Govardhan is a member on the editorial boards for Eight International Journals, Member of several Advisory & Academic Boards & Professional Bodies and a Committee Member for several International and National Conferences. He is a chairman and member on several Boards of Studies of various Universities and the chairman of CSI Hyderabad Chapter. He is the recipient of 21 international and national awards.