Extracting Interpretable Features for Early Classification on Time Series[*]

Zhengzheng Xing[†] Jian Pei[†] Philip S. Yu[‡] Ke Wang[†]
[†] Simon Fraser University, Canada, {zxing,jpei,wangk}@cs.sfu.ca
[‡] University of Illinois at Chicago, USA. psyu@cs.uic.edu

Abstract
Early classification on time series data has been found highly useful in a few important applications, such as medical and health informatics, industry production management, safety and security management. While some classifiers have been proposed to achieve good earliness in classification, the interpretability of early classification remains largely an open problem. Without interpretable features, application domain experts such as medical doctors may be reluctant to adopt early classification. In this paper, we tackle the problem of extracting interpretable features on time series for early classification. Specifically, we advocate local shapelets as features, which are segments of time series remaining in the same space of the input data and thus are highly interpretable. We extract local shapelets distinctly manifesting a target class locally and early so that they are effective for early classification. Our experimental results on seven benchmark real data sets clearly show that the local shapelets extracted by our methods are highly interpretable and can achieve effective early classification.

1 Introduction

Early classification on temporal data makes prediction as early as possible provided that the prediction quality satisfies expectation. In other words, early classification tries to optimize earliness under a requirement on minimum accuracy, instead of optimizing accuracy in general classification methods. Early classification on time series data has been found highly useful in some time-sensitive applications. For example, a retrospective study of the clinical data of infants admitted to a neonatal intensive care unit [5] found that the infants, who were diagnosed with sepsis disease, had abnormal heartbeat time series patterns 24 hours preceding the diagnosis. Monitoring the heartbeat time series data and classifying the time series data as early as possible may lead to early diagnosis and effective therapy. As another example, Bernaille *et al.* [2] showed that by only observing the first five packages of a TCP connection, the application associated with the traffic flow can be classified accurately. The applications of online traffic can be identified without waiting for the TCP flow to end. Early classification of time series data can also find applications in, for example, anomaly detection, intrusion detection, health informatics, and process control.

Constructing classifiers capable of early prediction is far from trivial. It is important to note that we are not interested in classic time series prediction (see [12] and the references therein), which predicts the value of the time series at some lead time in the future. Most of the existing classification methods on time series data extract features from full-length time series, and do not opt for earliness of prediction. In those traditional time series classification methods, the optimization goal is often to maximize classification accuracy.

We proposed the ECTS method for early classification on time series [15], which is based on nearest neighbor classification. The central idea is to explore the stability of the nearest neighbor relationship in the full space and in the subspaces formed by prefixes of the training examples. Although ECTS has made good progress, it only provides classification results without extracting and summarizing patterns from training data. The classification results may not be satisfactorily interpretable and thus end users may not be able to gain deep insights from and be convinced by the classification results.

Interpretability is critical for many applications of classification, such as medical and health informatics, industry production management, safety and security management. Without interpretable features, application domain experts such as medical doctors may often be reluctant to adopt a classification approach. For example, predicting a patient belonging to the positive class by simply linking the patient to several existing patients may not be sufficiently convincing and useful,

[*]The authors are deeply grateful to Professor Eamonn Keogh for his insightful comments and suggestions on an early version of this paper. This research is supported in part by NSERC through an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplements Grant, and by US NSF through grants IIS 0905215, DBI-0960443, OISE-0968341 and OIA-0963278. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

since disease patterns can be highly complicated and the patient time series data may cover a long time period. Instead, a medical doctor strongly prefers an early classification statement referring to several highly interpretable features, such as a short segment of the patient's time series carrying a distinct feature shared by a significant subgroup of patients having the disease. Such a statement may help the doctor to link to some specific disease patterns. Moreover, classification is often employed as a data exploration step, where summarization of the data in a target class using interpretable distinct features becomes the central task.

To the best of our knowledge, the problem of extracting interpretable features for early classification on time series data largely remains untouched, which is the topic of this paper. To tackle the problem, we need to answer several essential questions. First, what kinds of feature can be easily understood? Second, what are the proper criteria for interpretable features for early classification? Last, how can we extract features that are effective for early classification?

In this paper, we make concrete progress in answering the above questions. Specifically, we introduce local shapelets as features, which are essentially segments of time series. Since local shapelets are consistent with training time series, they are highly interpretable. We extract local shapelets distinctly manifesting the target class locally and early so that they are effective for early classification. Our experimental results on seven benchmark real data sets using a simple rule-based classifier clearly show that the features extracted by our methods are highly interpretable and can achieve effective early classification.

The rest of the paper is organized as follows. We review the related work in Section 2. We define local shapelets in Section 3. We describe the feature extraction step and the feature selection step in Sections 4 and 5, respectively. We report our experimental results in Section 6. Section 7 concludes the paper.

2 Related Work

Diez *et al.* [4] were the first who mentioned the term of early classification for time series. However, their method did not optimize earliness in classification. Instead, they referred early classification as classifying partial examples which are prefixes of complete time series. They simply ignored the predicates on unavailable suffixes and only used the linear combination of the available predicates for classification.

Aníbal *et al.* [3] applied a case based reasoning method to classify time series to monitor system failures in a simulated dynamic system. A KNN classifier was used to classify uncompleted time series using various distances, such as Euclidean distance, DTW (Dynamic time warping) and Manhattan distance. The simulation results showed that by using case based reasoning, the most important increase of classification accuracy occurs on the prefixes through 30-50% of the full length. The study demonstrated the opportunities for early classification, though it still did not optimize earliness systematically.

Recently, we tackled the problem of early classification on time series by exploring the earliness of classification [15]. However, as mentioned in Section 1, our previous study did not extract any explicit features.

Our other previous work [13] explored a feature based method for early classification on symbolic sequences. However, to apply that method on time series, one has to first discretize time series, which is a non-trivial task. Our extensive empirical study [13] showed that the method does not work well on time series data.

Classification on time series and temporal sequence data has been investigated extensively due to its fruitful applications. Please refer to a recent survey on the topic [14] and the references therein. The existing work on time series and sequence classification mainly focuses on optimizing classification quality, and does not consider optimizing earliness. This is the critical difference between this study and those methods. However, it is natural to ask what we can learn from those feature based classification methods for time series and sequences. We examine a recent and representative method [16] to address this issue.

Ye *et al.* [16] proposed the notion of shapelets as features for time series classification. Shapelets are subsequences of time series manifesting a class as much as possible. Technically, a shapelet is a pair (s, δ), where s is a time series subsequence, and δ is a distance threshold. A time series subsequence s' is considered matching a shapelet (s, δ) if $dist(s, s') \leq \delta$.

Ye *et al.* [16] adopted Euclidean distance as the measure, and matching is defined on time series subsequences of the same length. The distance threshold δ is learned by maximizing the information gain. Among all the possible features as such, a shapelet is the one separating the two classes with the best information gain. In other words, maximizing information gain is the criterion used in both learning distance thresholds for shapelet candidates and choosing the shapelets. Ideally, a perfect shapelet of a class is the one representing all time series in the class but does not cover any time series in other classes. For multiple class classification, the shapelet selection process is integrated with the construction of a decision tree. Features with higher information gain will be put in the upper part of the tree.

As shown in [16], shapelets provide an effective ap-

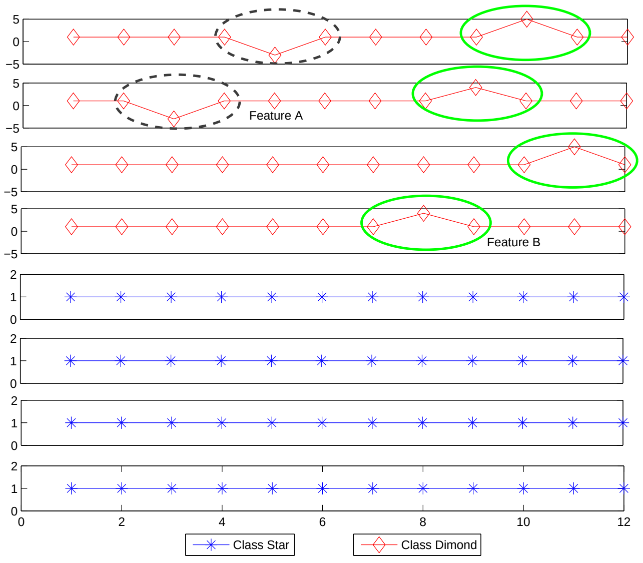


Figure 1: Locally distinctive feature for early classification.

proach for time series classification with good interpretability. However, we cannot directly apply shapelets for early classification. The shapelet method, though capturing local features of time series, focused on learning *globally distinctive* features with the maximal information gain. For early time series classification, some *locally distinctive* features, which do not have the optimal information gain, may be important. Here, a locally distinctive feature is one representing a subset of time series in one class nicely and exclusively.

Example 1. *Consider Figure 1, where there are two classes of time series. Feature A is shared by a subset of time series in the diamond class and does not appear in the star class at all. Feature B covers all time series in the diamond class but not any in the star class. Feature B is the shapelet of this data set, since it covers more instances in the diamond class, and thus has a higher information gain than feature A. We consider feature A as a local feature comparing to feature B. Interestingly, for early classification, feature A is highly useful, since it represents half of the cases in the diamond class and precedes feature B.* ■

The above example clearly shows that, in order to select locally distinctive feature for early classification, we need new strategies in feature extraction other than those in general classification methods maximizing global classification quality.

3 Local Shapelets

Definition 1. (Preliminaries) *A* ***time series*** *t is a sequence of readings. For the sake of simplicity, in this paper we assume that each reading is a real number. The* ***length*** *of the time series t is the number of readings in t, denoted by $len(t)$. The i-th reading $(1 \leq i \leq len(t))$ of t is denoted by $t[i]$.*

A time series s is a ***subsequence*** *of another time series t, denoted by $s \sqsubseteq t$, if $len(s) \leq len(t)$ and there exists a positive integer i_0 $(1 \leq i_0 \leq len(t))$ such that $t[i_0 + j] = s[j]$ for $(0 \leq j < len(s))$.*

Let T be a ***training set*** *of time series where each time series $t \in T$ carries a* ***class label*** *$c \in C$, and C is the set of classes. A time series $t \in T$ is called a* ***training example****. For a time series $t \in T$, we denote by $C(t)$ the class label of t. For a class $c \in C$, let T_c be the set of time series in T that carry label c, that is, $T_c = \{t \in T | C(t) = c\}$.* ■

What kinds of features are highly interpretable to end users and highly useful for early classification? While there are many possible choices, in this paper, we argue that subsequences of time series are a natural and preferable choice due to at least two reasons.

First, *subsequences of time series stay in the same data space of the input data.* Subsequences of time series do not require any transformation in feature extraction, and thus are intuitive for end users. Subsequences can capture the characteristics and phenomena that can be easily linked to end users' original applications.

Second, *subsequences can capture the local similarity among time series effectively.* A major task in early classification is to capture the trends manifesting subtypes in the target class. Subsequences as features can provide the insights on what and when time series are similar, which are highly preferable for high interpretability.

We cannot expect that, in general, many time series have segments matching a subsequence feature exactly. Instead, if a subsequence feature and a segment of a time series are similar enough, we can regard that the time series matches the feature. The similar idea was explored in shapelets [16]. We define local shapelets.

Definition 2. (Local shapelets and matches) *A* ***local shapelet*** *is a triple $f = (s, \delta, c)$, where s is a time time series, δ is a* ***distance threshold****, and $c \in C$ is a class. We write a local shapelet $f = (s, ?, c)$ if the distance threshold is not determined yet. For a local shapelet f, we call class c the* ***target class****, and the other classes the* ***non-target classes****.*

A local shapelet $f = (s, \delta, c)$ ***matches*** *a time series t if there is a subsequence $s' \sqsubseteq t$ such that $dist(s', s) \leq \delta$, where $dist(\cdot, \cdot)$ is a similarity measure* *in question. In this paper, we use Euclidean distance for the sake of simplicity. Consequently, we require $len(s) = len(s')$. However, our method can be applied to any other similarity measures (not necessarily a metric). To keep our discussion simple, in the rest of the paper, we use the terms "similarity" and "distance" interchangeably.* ■

Now, the problem becomes how we can extract local shapelets for early classification. In the rest of the paper, we propose an approach called *EDSC* (for Early Distinctive Shapelet Classification). The framework consists of two steps.

Feature extraction In the first step, we extract local shapelets that are effective in classification. Specifically, we consider all subsequences up to a certain length of the time series in the training set as local shapelets. For each local shapelet, we learn a robust distance threshold. By doing so, we obtain a (possibly large) set of distinctive features for time series classification. The feature extraction step will be discussed in Section 4.

Feature selection In the second step, we select a small subset of local shapelets by the criteria of earliness and popularity. This step ensures that the features selected opt for early classification, and opt out of overfitting. We use a simple rule-based classifier to select features. The feature selection step will be discussed in Section 5.

Although we borrow the idea in shapelets [16] in the notion of local shapelets, as will be made clear in Section 4, the feature extraction and selection methods in this paper are drastically different from [16].

4 Feature Extraction

The first step in EDSC extracts features from the training data set. It takes two parameters: $minL$ and $maxL$. Then, it extracts all subsequences of length between $minL$ and $maxL$ from every training example as the local shapelets. For each shapelet, we need to learn a distance threshold. This is the major task in feature extraction, and the topic of this section.

A local shaplet $f = (s, \delta, c)$ is considered *distinctive* if all time series matching f have a high probability to belong to class c. In this section, we study how to learn local shapelets effective for classification.

As mentioned before, the shapelet method [16] learned the distance threshold by maximizing the information gain. For early classification, we prefer some local features which are distinctive and early. Therefore, we need to develop a new feature extraction method to harvest distinctive local shapelets.

4.1 Best Match Distances

A time series may have multiple segments that are similar to a local shapelet, that is, their distance to the local shapelet are under the distance threshold. How well does a local shapelet match a time series? It can be measured by the best match distance (BMD).

Definition 3. (Best match distance) *For a local shapelet $f = (s, ?, c)$ and a time series t such that $len(s) \leq len(t)$, the **best match distance (BMD** for short) between f and t is*

$$BMD(f, t) = \min_{s' \sqsubseteq t, len(s') = len(s)} \{Dist(s, s')\}$$

■

For a local shaplet f, we can consider the distribution of the BMDs of the time series in the target class and the non-target classes. If most of the time series close to f in BMD belong to the target class, then f is distinctive for the target class. We will further argue for the use of BMDs in Section 4.4.

We can calculate the BMD between a local shapelet f and every time series in the training data set. These BMDs can be used to approximate the distribution of BMDs between f and the time series to be classified.

Definition 4. (BMD-list) *For a local shapelet $f = (s, ?, c)$ and a training data set T of N time series, the **best match distance list (BMD-list** for short) of f is the list of the BMDs between f and the time series in T, sorted in ascending order, denoted by*

$$V_f = \langle d_{i_1}(C(t_{i_1})), d_{i_2}(C(t_{i_2})), \ldots, d_{i_N}(C(t_{i_N}))\rangle$$

where $t_{i_j} \in T$, $d_{i_j} = BMD(s, t_{i_j})$, and $d_{i_j} \leq d_{i_{j'}}$, for $j < j'$.

*Moreover, the **target BMD-list (non-target BMD-list**, respectively), denoted by $V_{f,c}$ ($V_{f,\bar{c}}$), is the BMD-list of f on T_c ($T - T_c$).* ■

For a local shapelet $f = (s, \delta, c)$, a time series t_i in the training set matches f if and only if $BMD(s, t_i) \leq \delta$. How can we find an effective distance threshold for a local shapelet according to its BMD-list?

For a local shapelet $f = (s, \delta, c)$ and a BMD-list $V_f = \langle d_1, d_2, \ldots, d_N\rangle$, the *precision* of f is

$$Precision(f) = \frac{\|\{d_i | d_i \leq \delta \wedge C(t_i) = c\}\|}{\|\{d_i | d_i \leq \delta\}\|}$$

To make a local shapelet as distinctive as possible, a naïve method is to choose a distance threshold maximizing $Precision(f)$. However, this may lead to very small distance thresholds for local shapelets and thus over fit the training data. If we take a subsequence of a training example as a local shapelet, trivially, setting $\delta = 0$ always achieves a 100% precision.

Alternatively, we can set a precision threshold, e.g., 90%, and choose a distance threshold that can achieve a precision above the precision threshold. If there are multiple distance thresholds that meet the requirement, we can pick the largest one, since it enables the local shapelet to cover more training data.

EXAMPLE 2. (THE NAÏVE METHOD) *Consider a training data set T with 11 time series in the target class c, and 9 in the other class $\bar{c}$. Suppose the BMD-list of a local shapelet f is $V_f = \langle 0(c), 0.89(c), 2.54(c), 3.11(c), 3.26(c), 4.28(c), 9.70(c), 15.29(\bar{c}), 15.99(c), 16.96(c), 18.28(c), 18.57(\bar{c}), 19.02(\bar{c}), 19.25(\bar{c}), 19.36(\bar{c}), 20.09(\bar{c}), 21.21(\bar{c}), 22.56(\bar{c}), 25.84(\bar{c})\rangle$.*

Suppose the precision threshold is 90%. We can set a distance threshold satisfying the precision threshold and maximizing the number of time series covered. Any distance threshold in the range $[18.28, 18.57)$ can achieve a precision of $\frac{10}{11} = 91.67\%$.

However, one serious concern is that the distance threshold chosen as such lays in a dense region of the non-target class ($\bar{c}$). Such a distance threshold is unlikely robust in classifying unseen time series. In this naïve method, we only count the training examples in different classes but do not consider the distribution of the BMDs of the target/non-target classes. ■

In the rest of this section, we will propose two methods to learn robust distance thresholds for local shaplets. The first approach uses density estimation and the second approach uses Chebyshev's inequality.

4.2 KDE: Learning Distance Thresholds Using Kernel Density Estimation

The central idea of the KDE method is to apply kernel density estimation [8] on the BMD-list to estimate the probability density functions of the target class and the non-target classes, and then set the distance threshold δ so that at every point in the range of $[0, \delta]$ the probability density of belonging to the target class passes a probability threshold.

Given a random sample $\{x_1, x_2, \ldots, x_N\}$ drawn from a probability density distribution $f(X)$, the kernel density of $f(X = x)$ can be estimated by

$$\hat{f}(X = x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x - x_i}{h}), \quad (4.1)$$

where K is a kernel function and h is a smoothing factor [8]. In this paper, we adopt the Gaussian kernel

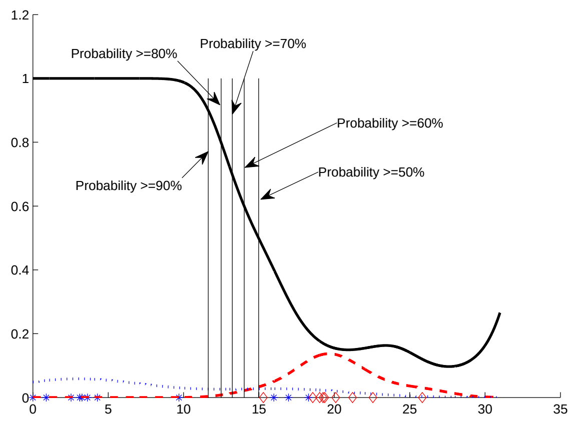


Figure 2: Learning distance threshold by KDE.

which has been popularly used.

$$K(\frac{x - x_i}{h}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x - x_j)^2}{2h^2}}. \quad (4.2)$$

To select an appropriate bandwidth, we use a widely adopted approach [11] to estimate the bandwidth by

$$h_{optimal} = 1.06\sigma N^{-\frac{1}{5}}, \quad (4.3)$$

where σ is the standard deviation of the sample.

Suppose we have m classes. Let $C(x) \in \{1, \ldots, m\}$ for any sample x. By estimating the density distribution $\hat{f}_j(x)$ for each class j ($1 \leq j \leq m$), we can calculate the probability that a sample x belongs to class j by

$$Pr(C(x) = j | X = x) = \frac{p_j \hat{f}_j(x)}{\sum_{k=1}^{m} p_k \hat{f}_k(x)}, \quad (4.4)$$

where p_k is the class prior [8].

To learn a robust distance threshold for a local shapelet $(s, ?, c)$, we propose to use kernel density estimation to utilize the distribution of the BMDs. We call this the *KDE method*, which runs as follows.

For a local shapelet $f = (s, ?, c)$, we estimate the kernel density for the target BMD-list $V_{f,c}$ and the non-target BMD-list $V_{f,\bar{c}}$, respectively. Then, we estimate the class probabilities of any time series given its best match distance to s by Equation 4.4. Given the class probabilities, we learn the distance threshold for f.

Let us exemplify the KDE method.

EXAMPLE 3. (KDE) *Consider the same training set and local shapelet f as in Example 2. Figure 2 plots the distribution of the BMDs between f and the training examples. Moreover, the dotted curve is the estimated* *density function $\hat{f}_c(x)$ for the target class and the dashed curve is the estimated kernel density $\hat{f}_{\bar{c}}(x)$ for the non-target class. The solid curve is the estimated probability density of time series belonging to the target class.*

We can choose the distance threshold according to the estimated probability of time series belonging to the target class. In the figure, the solid vertical lines are several distance thresholds learned using different probability thresholds. A distance threshold corresponding to a high probability threshold captures a region dominated by the target class, while a distance threshold corresponding to a low probability threshold moves into a region where the two classes are mixed. ■

Formally, we define the KDE feature extraction.

Definition 5. (KDE extraction) *Given a local shapelet $f = (s, ?, c)$ and its BMD-list V_f, the KDE method learns a distance threshold δ for f such that for any time series x that $BMD(f, x) \leq \delta$, $P(C(x) = c|X = x) \geq \beta$, where β is a user defined probability threshold, and $P(C(x) = c|X = x)$ is obtained by kernel density estimation using V_f.* ■

The kernel density estimation takes time $O(N^2)$, where N is the number of time series in the training data set. In our implementation, if the learned distance threshold $\delta = 0$, this local shapelet is discarded.

4.3 CHE: Learning Distance Thresholds Using Chebyshev's Ineqaulity Let X be a random variable with a finite mean μ and a finite variance σ^2. Then, for any positive number $k \geq 0$, the one tail Chebyshev's inequality states [1]

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2+1}. \tag{4.5}$$

To learn a distance threshold for a local shapelet $f = (s, ?, c)$, we can treat the BMDs of the time series in the non-target class in V_f as a sample of a random variable and compute its mean and variance. Then, we can compute the range where the non-target class has a very low probability to appear using Equation 4.5.

Definition 6. (CHE extraction) *Given a local shapelet $f = (s, ?, c)$ and its BMD-list V_f, the CHE method learns a distance threshold δ for f such that*

$$\delta = \max\{Mean(V_{f,\bar{c}}) - k * Var(V_{f,\bar{c}}), 0\}$$

where $k \geq 0$ is a user specified parameter. ■

The user specified parameter k in the CHE method ensures that, if the training data set is a consistent sample of the data to be classified, the probability of a non-target time series matching f is no more than $1/(k^2 + 1)$.

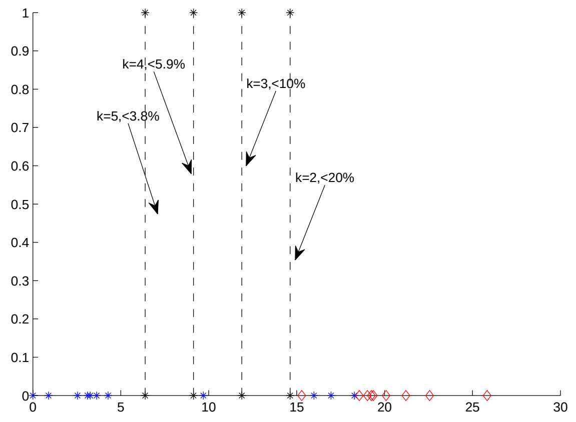


Figure 3: Learning distance threshold by CHE.

Example 4. (CHE) *Consider again the same training set and local shapelet f as in Example 2. Figure 3 shows the distance thresholds learned by CHE with $k = 2, 3, 4, 5$, respectively. For example, when $k = 3$, the probability of a non-target class time series matching f is no more than $1/(3^2 + 1) = 10\%$.* ■

Estimating probability $P(t|\bar{c})$ using Chebyshev's inequality is to prevent the distance threshold learned from laying in the dense region of the non-target classes. For a time series t matching feature $f = (s, \delta, c)$, by Bayesian theorem [9], the probability that t is in the target class c is,

$$P(c|t) = \frac{P(t|c)P(c)}{P(t|\bar{c})P(\bar{c}) + P(t|c)P(c)}. \tag{4.6}$$

The distance threshold learned by the CHE method only considers the distribution of the BMDs of the time series in the non-target class. Therefore, it only ensures that $P(t|\bar{c})$ in Equation 4.6 is small but does not guarantee that $P(c|t)$ is high. This problem will be handled in the feature selection step.

To compute the threshold, we only need to compute the mean and the variance of the BMD-list $V_{f,\bar{c}}$. The time complexity is $O(N)$, where N is the number of training examples. This is more efficient than the KDE method.

4.4 Why Should We Use BMDs? When we derive the distance threshold for a local shapelet $f = (s, ?, c)$, we use the BMDs between f and the time series in the training data set. Why do we use the BMDs?

Alternatively, let us consider the *worst match distance* (*WMD* for short) between a local shapelet $f =$

(s, δ, c) and a time series t, that is,

$$WMD(f,t) = \max_{s' \sqsubseteq t, len(s')=len(s)} \{Dist(s,s')\}.$$

Local shapelet f is considered matching t if $WMD(f,t) \leq \delta$, that is, every subsequence of t of length $len(s)$ has a distance to s of no more than δ. We can also try to learn the distance threshold for f given a training data set.

However, WMDs cannot lead to any early classification. Before examining the whole time series, we cannot determine the corresponding WMD value, and consequently cannot determine whether the time series matches a local shapelet.

Using BMDs,to classify a time series t online in the classification step, once we find a match between t and a local shapelet $f = (s, \delta, c)$ of distance no greater than the distance threshold δ, we can classify t based on this feature. Although the match may not be the best match between t and f since we may not finish reading t yet, the condition $BMD(f,t) \leq \delta$ is established. The prediction is safe. This property facilitates early prediction.

5 Feature Selection

A good feature for early classification should have three properties: frequent, distinctive and early [13]. In Section 4, we discuss how to make each local shapelet as distinctive as possible. In this section, we discuss how to select a set of local shapelets that are early and frequent.

5.1 Feature Selection

Given a local shapelet $f = (s, \delta, c)$, we need to measure the utility of f in earliness, frequency, and distinctiveness.

Definition 7. (Utility) *For a time series t and a local shapelet $f = (s, \delta, c)$, let the* ***earliest match length (EML*** *for short)*

$$EML(f,t) = \min_{len(s) \leq i \leq len(t)} dist(t[i - len(s) + 1, i], s) \leq \delta.$$

EML measures the earliness of f in classifying t. If $BMD(f,t) > \delta$, $EML(f,t) = \infty$.

The ***weighted recall*** *of f on a training data set T is*

$$WRecall(f) = \frac{1}{\|T_{\bar{c}}\|} \sum_{t \in T} \frac{1}{\sqrt[\alpha]{EML(f,t)}}, \tag{5.7}$$

where α is a parameter to control the importance of earliness. When α increases, the earliness weights less comparing to the support. Especially,when $\alpha = \infty$, the weighted recall converges to the classical recall.

The utility *of f is*

$$Utility(f) = \frac{2 \times Precision(f) \times WRecall(f)}{Precision(f) + WRecall(f)}. \tag{5.8}$$

∎

This utility function extends the well known *F-measure* by taking the earliness of a feature into consideration. This utility measure carries the same spirit as the utility measure proposed in [13].

The set of local shapelets extracted in the first step of EDSC may be large. Moreover, it may contain many redundant local shapelets. A group of similar time series subsequences belonging to the same class may capture the similar features and thus are redundant. This situation is very similar to the feature selection problem in the associative classification methods [7].

As indicated by many existing studies on associative classification, learning an optimum set of features for classification is very expensive and cannot be scalable. Thus, we adopt a greedy approach.

A local shapelet f is said to *cover* a training example t if f matches t and the classes in f and t are identical. We maintain the set of training examples that are covered by some local shapelets selected as features during the feature selection process.

The feature selection step in EDSC works as follows. We rank all local shapelets in utility, and take the one of the highest utility, denoted by f_1, as a feature. All training examples covered by f_1 are marked. We consider the remaining local shapelets that can cover at least some training examples that are not marked as "covered" yet, and iteratively select the one of the highest utility. The iteration continues until either a required number of features are extracted or at least a certain percentage of training examples are covered.

The selected local shapelets can be used for early classification immediately. When we scan a time series only, we try to match the time series with the selected local shapelets. Once a match is found, a prediction is made.

5.2 Cost Analysis In the feature learning step, we consider all the subsequences of length between $minL$ and $maxL$ as the local shapelets. For each local shapelet, we learn the distance threshold. Suppose we have N time series in the training data set and the average length is L. The total number of length k subsequences is $(L - k + 1)N$. For a length k local shapelet, the cost of computing its BMD-list against the training data set in a straightforward way is $O(kN(L - k + 1))$. To compute the BMD-lists of all length k local

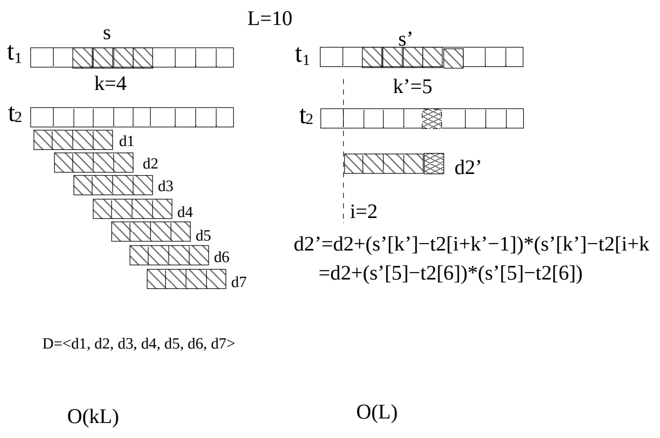

L=10
s
t1
k=4
t2
d1
d2
d3
d4
d5
d6
d7
D=<d1, d2, d3, d4, d5, d6, d7>
O(kL)
s'
t1
k'=5
t2
d2'
i=2
d2'=d2+(s'[k']−t2[i+k'−1])*(s'[k']−t2[i+k
=d2+(s'[5]−t2[6])*(s'[5]−t2[6])
O(L)

Figure 4: Sharing in computing BMDs .

shapelets, the cost is $O(k(L-k+1)^2N^2)$. For all the local shapelet between length $minL$ to $maxL$, the time cost of computing the BMD-lists is $O(\sum_{k=minL}^{maxL} k(L-k+1)^2N^2) = O(N^2L^4)$.

Given a BMD-list, to learn the distance threshold, we need either $O(N)$ for the CHE method or $O(N^2)$ for the KDE method. For all the local shapelets between length $minL$ to $maxL$, the time cost to learn the distance thresholds is $O(\sum_{k=minL}^{maxL}(L-k+1)N^2) = O(N^2L^2)$ for the CHE method and $O(N^3L^2)$ for the KDE method.

For the feature selection step, the complexity is bounded by sorting all the local shapelets extracted, which is $O(L^2Nlog(NL))$.

The major computational cost of EDSC comes from two aspects. First, we consider a large number of local shapelets. Second, similar to [16], a computational bottleneck is computing the BMD-lists.

5.3 A Speed-up Technique When computing the BMD-lists, we can share the computation among different local shapelets.

Given a time series t, for the subsequences of t starting from the same position, we can use them as a group to compute their BMD-lists in a batch. In Figure 4, we illustrate the idea of sharing computation by storing additional information of the matching distances.

Suppose we have a time series t_1 of length L, and we use its subsequence s and s' as two local shapelets. The length of s is 4 and the length of s' is 5. s and s' start from the same position in t_1. Let s be a prefix of s' and $len(s') = len(s) + 1$. When we compute the BMD of s against another length L time series t_2, we need to compute the Euclidean distance between s and every sliding window in t_2 of length $|s|$, and the time complexity is $O(L|s|)$. We store all the squared Euclidean distances between s and each sliding window in t_2 in a vector D. When we compute the BMD between s' and t_2, we only need cost $O(L)$ instead of $O(L * |s'|)$ by reusing D. This is because, to obtain the Euclidean distance between s' and a length 5 sliding window of t_2, we only need to update the distance between the last time point of s' and the last time point in a sliding window.

By using the above method, it is easy to see that we can compute the BMD-lists for all local shapelets from length 1 to L in time complexity $O(N^2L^3)$ instead of $O(N^2L^4)$ in the straightforward implementation.

Please note that the speedup method described here can handle time series without normalization only. It is an open challenging problem to speedup local shapelet finding for time series with normalization. We leave this for future work.

6 Experimental Results

We evaluate our methods on seven data sets from the UCR time series archive [6]. For comparisons, we conducted the experiments on the same data sets as in [15]. All the experimental results are obtained by using a PC computer with an AMD 2.2GHz CPU and 3GB main memory. The algorithms were implemented in C++ using Microsoft Visual Studio 2005.

6.1 Results Overview The results of the seven data sets are listed respectively in Table 1 (ECG), Table 2 (Gun Point), Table 3 (CBF), Table 4 (Synthetic Control), Table 5 (Wafer), Table 6 (OliveOil), and Table 7 (Two Patterns). For each data set, we also include the information about the size, dimensionality, and number of classes of the training data set and testing data set.

We compare four methods on each data set, namely, EDSC-CHE (EDSC with the CHE method), EDSC-KDE (EDSC with the KDE method), 1NN-Full (Full length 1NN with Euclidean distance) and ECTS (Early classifier for time series proposed in [15]). For EDSC-CHE and EDSC-KDE, we use a default rule which is the majority class. Without using the default rule, EDSC-CHE and EDSC-KDE may not cover all the time series to be classified. In the Tables 1–7, we also report the accuracy and coverage rate of the EDSC method without using the default rules, and they are referred as EDSC-CHE-NDefault and EDSC-KDE-NDefault, respectively.

All the results are obtained using the same parameter settings. For EDSC-CHE, we set $MinLen = 5$, $MaxLen = \frac{L}{2}$, and $k = 3$, where L is the full length of the time series. For EDSC-KDE, we set $MinLen = 5$, $MaxLen = \frac{L}{2}$, and $p = 95\%$. We observe that the results are insensitive to the parameter α in the weighted

ECG: 2 classes; 100 training inst.; 100 testing inst.; L=96				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	86.67%	90%	16.87/96	32
EDSC-CHE	82%	100%	24.78/96	32
EDSC-KDE-NDefault	90.70%	86%	20.34/96	29
EDSC-KDE	88%	100%	30.93/96	29
1NN-FUll	88%	100%	96/96	NA
ECTS	89%	100%	57.71/96	NA

Table 1: Results of ECG data set

CBF: 3 classes; 30 training inst.; 900 testing inst.; L=128				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	95.03%	89.44%	35.03/128	3
EDSC-CHE	87.89%	100%	44.84/128	3
EDSC-KDE-NDefault	94.94%	87.78%	35.12/128	3
EDSC-KDE	85.89%	100%	46.47/128	3
1NN-FUll	85.2%	100%	128/128	NA
ECTS	85.2%	100%	91.73/128	NA

Table 3: Results of CBF data set

Gun-Point: 2 classes; 50 training inst.; 150 testing inst.; L=150				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	97.18%	94.67%	64.75/150	8
EDSC-CHE	94.67%	100%	69.3/150	8
EDSC-KDE-NDefault	95.74%	94%	64.80/150	9
EDSC-KDE	94%	100%	69.97/150	9
Shapelets [16]	93.3%	100%	150/150	1
1NN-FUll	91.33%	100%	150/150	NA
ECTS	86.67%	100%	70.39/150	NA

Table 2: Results of Gun-Point data set

Syn.:6 classes; 300 training inst.; 300 testing inst.; L=60				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	94.49%	90.67%	30.62/60	38
EDSC-CHE	87.66%	100%	33.36/60	38
EDSC-KDE-NDefault	97.06%	90.67%	30.43/60	39
EDSC-KDE	90.33%	100%	33.19/60	39
1NN-FUll	88%	100%	60/60	NA
ECTS	89%	100%	53.98/60	NA

Table 4: Results of Synthetic Control data set

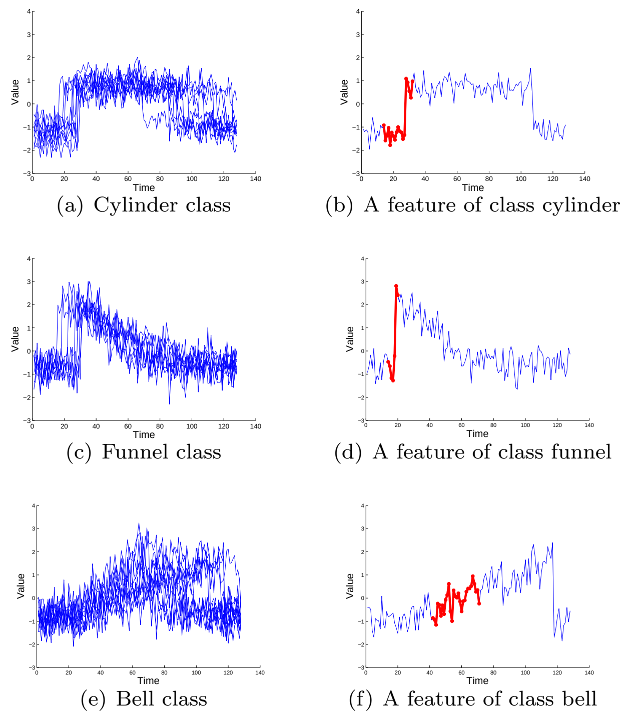
recall (Equation 5.7) when α is small. Limited by space, we omit the details here, and by default set $\alpha = 3$.

In Figure 5, we summarize the performance of EDSC-CHE, EDSC-KDE, ECTS and full 1NN in terms of classification accuracy and average prediction length. EDSC-CHE and EDSC-KDE are always earlier than ECTS, and sometimes significantly earlier, such as on the ECG, CBF and Synthetic control data sets. EDSC-CHE and EDSC-KDE have similar performance in terms of earliness. For the classification accuracy, generally, EDSC-KDE is more accurate than EDSC-CHE. It is shown that the distance threshold learning quality of EDSC-CHE is not as good as EDSC-KDE.

The above results on the benchmark data sets show that EDSC-CHE and EDSC-KDE can achieve competitive classification accuracies with great earliness.

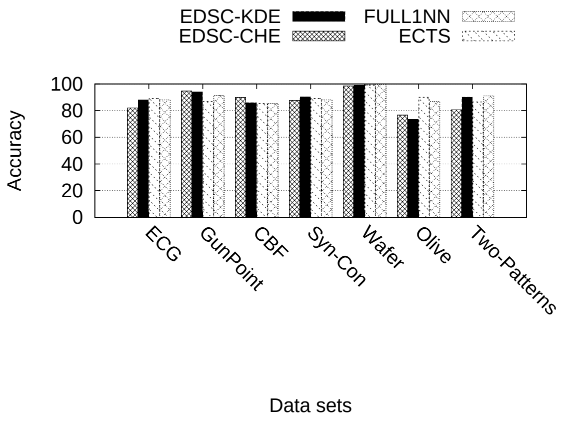
6.2 Interpretability of Features In this section, we exam the interpretability of the learned features using data sets CBF and Gun-Point as examples.

The CBF data set has three classes, namely cylinder, bell, and funnel. In Figures 6.2(a), (c), and (e), we plot the profiles of the three classes of the CBF data set in the left column. Both EDSC-CHE and EDSC-KDE extract 3 local shapelets on this data set, 1 feature for each class. The features extracted by the two methods are very similar. We only show the features from EDSC-CHE here (Figures 6.2(b), (d), and (f)). We highlight the feature in the time series which it comes from. The



(a) Cylinder class (b) A feature of class cylinder
(c) Funnel class (d) A feature of class funnel
(e) Bell class (f) A feature of class bell

Figure 6: Selected features on the CBF Data Set



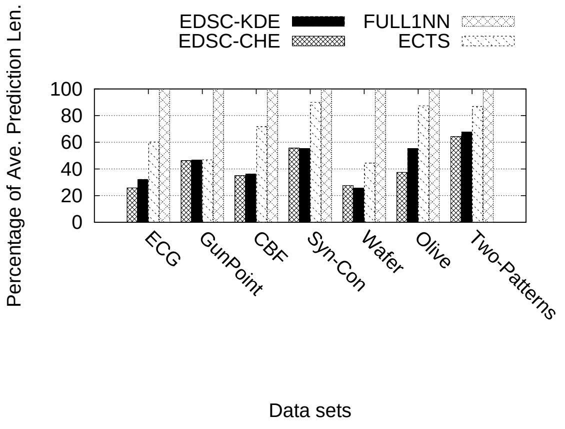


Figure 5: Comparison among EDSC, ECTS and Full-1NN

Wafer: 2 classes; 1000 training inst.; 6174 testing inst.; L=152				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	98.82%	99.51%	41.39/152	62
EDSC-CHE	98.49%	100%	41.93/152	62
EDSC-KDE-NDefault	99.15%	99.51%	38.42/152	52
EDSC-KDE	98.87%	100%	38.97/152	52
1NN-FUll	99.55%	100%	152/152	NA
ECTS	99.08%	100%	67.39/152	NA

Table 5: Results of Wafer data set

Two P.: 4 classes; 1000 training inst.; 4000 testing inst.; L=128				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover.	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	84.75%	93.75%	79.29/128	305
EDSC-CHE	80.6%	100%	82.33/128	305
EDSC-KDE-NDefault	94.94%	93.33%	83.62/128	279
EDSC-KDE	90%	100%	86.58/128	279
1NN-FUll	91%	100%	128	NA
ECTS	86.48%	100%	111.10/128	NA

Table 7: Results of Two Patterns data set

OliveOil: 4 classes; 30 training inst.; 30 testing inst.; L=570				
EDSC-CHE: $MinLen = 5; MaxLen = L/2; k = 3$ EDSC-KDE: $MinLen = 5; MaxLen = L/2; p = 95\%$				
	Accu.	Cover	Ave.Len.	♯Fs.
EDSC-CHE-NDefault	85.16%	90%	174.07/570	13
EDSC-CHE	76.67%	100%	213.48/570	13
EDSC-KDE-NDefault	95.45%	73%	223.18/570	13
EDSC-KDE	73.33%	100%	315.67/570	13
1NN-FUll	86.7%	100%	570/570	NA
ECTS	90%	100%	497.83 /570	NA

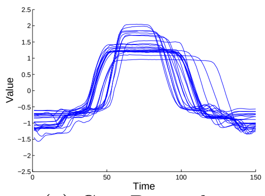
Table 6: Results of OliveOil data set

three classes are quite similar at the beginning. The features learned by EDSC-CHE represent the characteristics of each class, and lay in the early phases of the time series when the classes start to separate from each other.

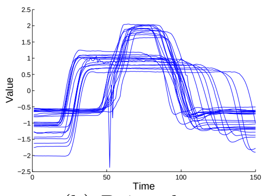
Let us take the Gun-Point data set as another example, which contains two classes, the Gun-Draw class and the Point class [10]. Figures 7(a) and (b), the profiles of the Gun-Draw class (left) and the Point class (right) are plotted, respectively. For the Gun-Draw class, the actors "draw a replicate gun from a hip-mounted holster, point it at a target for approximately one second, then return the gun to the holster" [10]. For the Point class, "The actors have their hands by their sides. They point with their index fingers to a target for approximately one second, and then return their hands to their sides" [10]. The Gun-Draw class is different from the Point class by two actions, "draw a gun from a holster", and "return the gun to the holster" [10].

EDSC-CHE learns 4 features for each class. In Figure 7(c) and (d), we plot the feature with the highest utility score for each class, respectively. For the GUN-Draw class, the feature captures the region of "draw a gun from the holster". It is the action to distinguish the two classes and it is an earlier feature than "return the gun to the holster". For the Point class, the feature happens to belong to an unexpected signal. By plotting the best matches of the time series for this feature (Figure 7(e)), we can see the feature represents the later moment of the "lifting the arm". The top features learned by the EDSC-KDE method are quite similar to the two features plotted, and thus are omitted here for the sake of space.

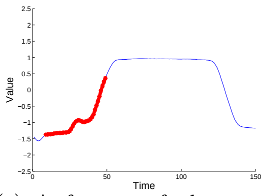
The above two examples on CBF and Gun-Point data sets demonstrate that the features learned by our methods can capture the early characteristics of



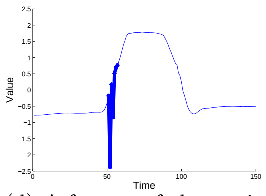
(a) Gun-Draw class



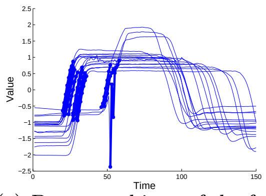
(b) Point class



(c) A feature of class gun-draw



(d) A feature of class point



(e) Best matchings of the feature in Figure 7(d)

Figure 7: Selected Features on the Gun-Point Data Set

different classes.

6.3 Sensitivity of Parameters The previous results are all generated by setting $k = 3$ in EDSC-CHE and $p = 95\%$ in EDSC-KDE. In learning the distance threshold, parameters k (in Definition 6) and p (in Definition 5) control the degree of distinctiveness of learned features. In this subsection, we use the ECG data set as an example to show the effects of the parameters. The results on the ECG data with different values of p and k are shown in Figures 8 and 9, respectively.

In EDSC-KDE, when we increase p, the features we learned should be more distinctive and the accuracy classified by the features is expected to increase. In Figure 8, the accuracy of EDSC-KDE-NDefault generally increases from 80% to 91% in the range of $p = 50\%$ to $p = 95\%$ and decreases after $p = 99\%$. When we set $p = 100\%$, no feature satisfies this probability threshold and the accuracy of EDSC-KDE-NDefault is 0. By examining the selected features, we found that when $p = 99\%$, some features with very small recalls are selected and lead to wrong classification due to overfitting.

When p increases, the coverage rate of EDSC-KDE-NDefault decreases, since the distance thresholds learned tends to be smaller. There is a trade-off between the accuracy and the coverage when p increases. By using a default classification rule, EDSC-KDE reaches the highest accuracy of 88% when $p = 95\%$ as the best balance between accuracy and coverage. When $p = 90\%$, the EDSC-KDE has a similar accuracy as 87%. On the other data sets, we also observe that the best results usually appear when setting $90\% \leq p \leq 95\%$, and the results are stable in this range.

Figure 9 plots the results on ECG data set using EDSC-CHE with various values of k. When k increases, the accuracy of EDSC-CHE-NDefault increases and the coverage rate decreases. When $k = 2.5$, EDSC-CHE reaches the best accuracy as 85% due to a good balance between the accuracy and coverage of EDSC-CHE-NDefault . On the other data sets, we observe that the best results usually happen when $2.5 \leq k \leq 3.5$. The average prediction length increases as $k > 1$ increases.

6.4 Efficiency Table 8 compares the training time using the straightforward implementations of EDSC-CHE and EDSC-KDE, as well as the implementations using the techniques in Section 5.3 (EDSC-CHE(Impr.) and EDSC-KDE(Impr.)). In the straightforward implementations, when computing BMD-lists we also incorporate the early stopping techniques proposed in [16].

EDSC-CHE is faster than EDSC-KDE. The technique discussed in Section 5.3 can significantly reduce the training time.

7 Conclusions

To tackle the problem of extracting interpretable features for early classification on time series, we develop the notion of local shapelets and the EDSC method. Our experimental results clearly show that the local shapelets extracted by our methods are highly interpretable and can achieve effective early classification. As future work, we plan to improve the effectiveness and efficiency of feature selection and early classification, and explore other types of features.

References

[1] A. O. Allen. *Probability, Statistics, and Queuing Theory with Computer Science Applications.* Acadimic Press, Inc., San Diago, CA, 1990.

[2] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *Computer Communication Review*, 36(2):23–26, 2006.

[3] A. Bregón, M. A. Simón, J. J. Rodríguez, C. J. Alonso, B. P. Junquera, and I. Moro. Early fault classification in dynamic systems using case-based reasoning. In *CAEPIA*, pages 211–220, 2005.

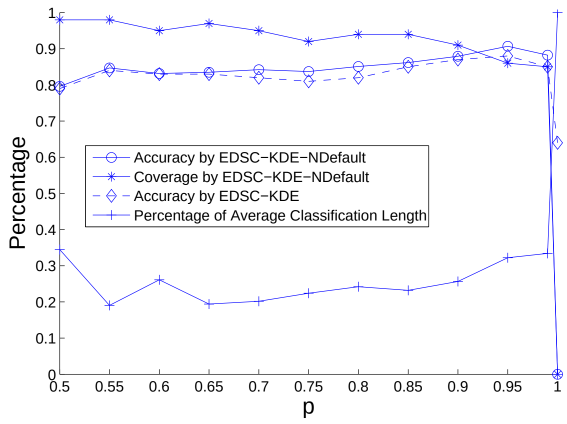


Figure 8: Results on ECG data set by varying p

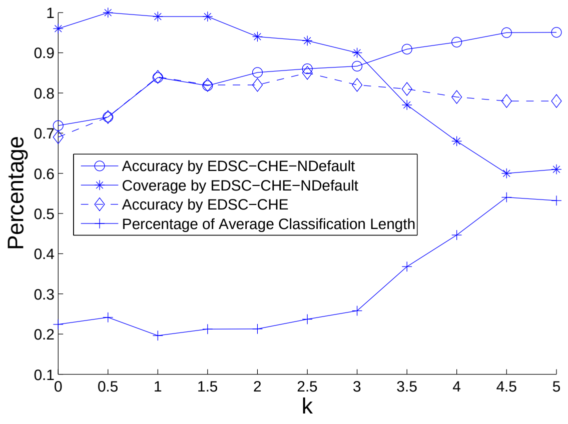


Figure 9: Results on ECG data set by varying k

Data set	# samples	Length	EDSC-CHE	EDSC-CHE(Impr.)	EDSC-KDE	EDSC-KDE(Impr.)
CBF	30	128	37.93 sec	9.52 sec	41.48 sec	13.05 sec
ECG	100	96	123.42 sec	26.05 sec	137.45 sec	40.17 sec
SynCon	300	60	252.83 sec	62.20 sec	332.60 sec	140.798 sec
GUN Point	50	150	165.13 sec	25.6 sec	170.21 sec	30.38 sec
OliveOil	30	570	6729.26 sec	1707.63 sec	6700.1 sec	1728.91 sec
TwoPattern	1000	128	37249.4 sec	10876.6 sec	40925.3 sec	14258.1 sec
Wafer	1000	152	73944.4 sec	10841.5 sec	103298 sec	40082.2 sec

Table 8: Training time comparison

[4] J. J. R. Diez, C. A. González, and H. Boström. Boosting interval based literals: variable length and early classification. *Intell. Data Anal.*, 5(3):245–262, 2001.

[5] M. P. Griffin and J. R. Moorman. Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. *PEDIATRICS*, 107(1):97–104, 2001.

[6] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification and clustering homepage: `http://www.cs.ucr.edu/~eamonn/time_series_data/`, 2006.

[7] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD '98*.

[8] M. D. Marzio and C. C. Taylor. Kernel density classification and boosting: an l2 analysis. *Statistics and Computing*, 15(2):113–123, 2005.

[9] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[10] C. A. Ratanamahatana and E. J. Keogh. Making time-series classification more accurate using learned constraints. In *SDM '04*.

[11] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 1992.

[12] A.S. Weigend and N.A. Gershenfeld, eds. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley.

[13] Z. Xing, J. Pei, G. Dong, and P. S. Yu. Mining sequence classifiers for early prediction. In *SDM'08: Proceedings of the 2008 SIAM international conference on data mining*, pages 644–655, 2008.

[14] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations*, Volume 12, Issue 1, pages 40-48, June 2010, ACM Press.

[15] Z. Xing, J. Pei, and P. S. Yu. Early classification on time series: A nearest neighbor approach. In *IJCAI'09*.

[16] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *KDD '09*.