# Ensemble Based Positive Unlabeled Learning for Time Series Classification

Minh Nhut Nguyen, Xiao-Li Li, and See-Kiong Ng

Institute for Infocomm Research, Singapore
{mnnguyen,xlli,skng}@i2r.a-star.edu.sg

**Abstract.** Many real-world applications in time series classification fall into the class of positive and unlabeled (PU) learning. Furthermore, in many of these applications, not only are the negative examples absent, the positive examples available for learning can also be rather limited. As such, several PU learning algorithms for time series classification have recently been developed to learn from a small set *P* of labeled seed positive examples augmented with a set *U* of unlabeled examples. The key to these algorithms is to accurately identify the likely positive and negative examples from *U*, but it has remained a challenge, especially for those uncertain examples located near the class boundary. This paper presents a novel ensemble based approach that restarts the detection phase several times to probabilistically label these uncertain examples more robustly so that a reliable classifier can be built from the limited positive training examples. Experimental results on time series data from different domains demonstrate that the new method outperforms existing state-of-the art methods significantly.

**Keywords:** Ensemble based system, positive and unlabeled learning, time series classification.

## 1 Introduction

Many real-world data mining application domains, such as aerospace, finance, manufacturing, multimedia and entertainment, involve time series classification [1-3]. For example, a typical aircraft health monitoring application in aerospace would be to classify the states of the airplane engines into either the normal or faulty states based on time series sensor readings from multiple sensors (e.g. vibration and temperature sensors) attached to the aircraft. Most of classification methods directly apply traditional supervised learning techniques that rely on large amounts of labeled examples from predefined classes for learning. In practice, this paradigm is not practical because collecting and labeling large sets of data for training are often very expensive if not impossible.

Researchers have proposed alternative learning techniques to build classifiers from a small amount of labeled training data enhanced by a larger set of unlabeled data that are typically easy to collect. These methods include semi-supervised learning [4-6] and Positive Unlabeled learning (PU learning) [7-13]. While both approaches exploit

S.-g. Lee et al. (Eds.): DASFAA 2012, Part I, LNCS 7238, pp. 243–257, 2012.

the unlabeled data (*U*) to enhance the performance of their classifiers, they differ in their training data requirements: PU learning only requires positive data (*P*) whereas semi-supervised learning still requires both positive and negative training data. PU learning is therefore applicable in a wide range of application domains, such as text classification, medical informatics, pattern recognition, bioinformatics and recommendation system, where negative data are often unavailable. However, the applications of PU learning to classify time series data have been relatively less explored due to specific challenges of time series classification such as high feature correlation [14]. As far as we know, there are only 3 research works that applied PU learning approaches for time series data classification.

The pioneering work, proposed by Wei and Keogh [14], iteratively expands the positive set from the initial positive examples using the unlabeled data that are most similar to them in terms of Euclidean distance, with the remaining unlabeled data being extracted as negative data. The method is highly dependent on having a good stopping criterion; otherwise, early stopping will result in an expansion of only a small number of positives, with highly noisy negatives. To improve the algorithm, a more recent work [15] attempted to propose a good stopping criterion by using the historical distances (in this case, dynamic time warping distance) between candidate examples from *U* to the initial positive examples. Although the refinement has enabled more positive examples to be extracted, it is still unable to identify *accurate* positives (and hence negatives) from *U*, especially when the actual positives and negatives in *U* are severely unbalanced. The experimental results reported showed high precision but very low recall for classification.

More recently, to tackle the challenge of constructing accurate boundary between positive and negative data in *U*, we proposed a new PU approach called LCLC for time series classification [16]. Unlike the previous methods, LCLC adopts a cluster-based approach instead of instance-based approach. First, the unlabeled set *U* is partitioned into small unlabeled local clusters (*ULCs*) using the *K-means* algorithm [17]. All the examples within an individual cluster will be assigned a same label as either *LP* (Likely Positive) or *LN* (Likely Negative). The local clusters are also exploited for more robust feature selection for classification. A cluster chaining approach is then applied to extract the boundary positive and negative clusters (*ULCs*) to estimate the decision boundary between the actual positives and negatives in *U*. LCLC has been demonstrated to perform much better than the first two PU learning methods, as it can identify the boundary positive and negative clusters from *U* more accurately. While LCLC's cluster-based approach (i.e. all the instances within an individual cluster will be assigned the same label) is more robust than traditional instance-based approach, in practice, not all the examples within the individual local clusters will actually be from same class. This means that some instances within each cluster may be misclassified. When these misclassified examples (especially when they are in the boundary clusters) are used to build the final 1-NN classifier (i.e. classification based on the top one nearest neighbor; Keogh et al [18] performed a comprehensive empirical evaluation on the current state-of-the-arts which shows 1-NN to be the best technique), the performance of overall LCLC algorithm is less satisfactory than expected. In Section 2, we will go into the further details of the LCLC algorithm as well as the reason that LCLC algorithm generates false positives/negatives.

In this paper, we propose a novel ensemble based approach En-LCLC (Ensemble based Learning from Common Local Clusters) to overcome the drawbacks of the LCLC algorithm. Our proposed En-LCLC method adopts an ensemble-based strategy by performing the LCLC algorithm multiple times on different cluster settings to obtain multiple diverse classifiers. We can then assign each instance with a "soft" probabilistic confidence score based on its overall classification results that could better indicate each instance's class label. Based on the probabilistic scores, we also identify and remove potential noisy instances which could confuse our classifier. An Adaptive Fuzzy Nearest Neighbor (AFNN) classifier is then constructed based on the clean set of "softly labeled" positive and negative instances identified.

The rest of the paper is organized as follows. We provide an overview on the LCLC algorithm in Section 2. We then present our proposed En-LCLC algorithm in Section 3. Results from extensive experiments on time series data across diverse fields reported in Section 4 show that the classifiers built using En-LCLC algorithm can indeed identify the ground truth's positive and negative boundaries more accurately, leading to improvements in classification accuracy. Finally, Section 5 concludes the paper.

## 2 LCLC Algorithm and Its Weakness

In this section, we describe the LCLC method proposed in [16] in further details. As mentioned earlier, the first step of LCLC algorithm groups the unlabeled data $U$ into local clusters and selects independent and relevant features based on these clusters. Subsequently, LCLC algorithm extracts reliable negative set $RN$ from $U$, with the remaining clusters belonging to $U$-$RN$ regarded as ambiguous clusters ($AMBI$). Finally, LCLC determines likely positive clusters $LP$ and negative clusters $LN$ from $AMBI$ using cluster chaining, and the final classifier is built using all the extracted positives and negatives from $U$.

Algorithm 1 shows the main steps of the LCLC. Steps 1 and 2 perform the local clustering and feature selection. In Step 1, LCLC partitions the unlabeled data $U$ into small local clusters $ULC_i$ ($i$=1, 2, …, $K$) using *K-means* clustering method. Each local cluster $ULC_i$ is then treated as an observed variable of the time series data, and it assumes that all the instances belonging to a local cluster share the same principal component and have the same class label.

In Step 2, the *Clever-Cluster* method [19] is then used to select $K$ common feature subset from the positive set $P$ and a partitioned coherent unlabeled clusters $ULC_i$. It first computes the principal components for each time series observations which are the positive set $P$ and unlabeled clusters $ULC_i$. Descriptive common principal components are then computed across all these principal components and used to select $K$ highest mutual information features. Interested readers could find more details in [16]. The intuition for such selection is based on the observation that a well-selected subset of the common principal features can capture the underlying characteristics of the time series dataset to enable accurate extraction of the remaining *hidden* positives/negatives from $U$.

Step 3 identifies the Reliable Negative set *RN* from *U* based on the similarities between the local clusters $ULC_i$ to the initial positive cluster *P*. In this step, LCLC computes the Euclidean distance of each $ULC_i$ from the positive set *P* using common principal features extracted in Step 2. After that, it extracts those local clusters which are farthest away from *P* and store them into *RN*. The size of *RN* is set to contain about a half of the local clusters, while the other half is considered as ambiguous clusters *AMBI* in Step 4.

**Algorithm 1.** LCLC algorithm
**input**: Initial positive data *P*, Unlabeled dataset *U*, number of clusters *K*

1. *K-ULCs* ← Partition *U* into *K* local clusters using *K-means*;
2. Select *K* features from the raw feature set ← *Clever-Cluster*(*P*, *K-ULCs*);
3. Extract Reliable Negative Examples *RN* from the Unlabeled dataset *U*;
4. Define the ambiguous clusters *AMBI* = *U* – *RN*;
5. Identify likely positive clusters *LP* and likely negative clusters *LN* from the AMBI clusters using cluster chaining for boundary decision;
6. Build a 1-NN classifier using *P* together with *LP* as a positive training set, and *RN* together with *LN* as a negative training set.

By now (after Step 4), LCLC algorithm has obtained a positive data *P* and reliable negative data *RN* that can be used to further extract the *likely* positive clusters *LP* and the *likely* negative clusters *LN* from the ambiguous clusters *AMBI* which are near the positive and negative boundary. Step 5 performs a novel *cluster chaining* method to label these boundary clusters. The basic idea of cluster chaining is to build cluster chains starting from the positive *P*, going through one or more *AMBI* clusters, and finally stopping at a reliable negative cluster in *RN*. Figure 1 illustrates the scenario where there are two reliable negative clusters *RN* far away from the positive cluster *P*, and 4 *AMBI* clusters located between the positive cluster (*P*) and negative clusters (*RN*). Two cluster chains have been built here. For each cluster chain, LCLC finds the *breaking link* (decision boundary) with *maximal distance* between the clusters that separates the cluster chain into two sub-chains. All the *AMBI* clusters within the sub-chain that contains *P* will be regarded as likely positive clusters and stored into *LP,* while the *AMBI* clusters within the other sub-chain that includes *RN* are regarded as likely negative clusters and stored into *LN*.

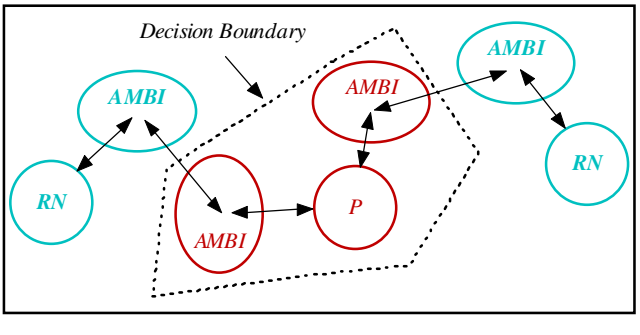


**Fig. 1.** Cluster chaining for boundary decision

Finally, LCLC uses *P* together with *LP* as a positive training set, and *RN* together with *LN* as a negative training set to build the final 1-NN classifier for time series classification.

Although LCLC works better than the existing PU learning methods identifying the boundary clusters more accurately, we observed that it still has two drawbacks. Firstly, it assumes that *all the instances belonging to a local cluster have the same class label*. Clustering ensures that *most* of the examples within a same cluster belong to the same class, but some of the examples within same cluster could belong to other classes. By assigning the same label to all the examples within each cluster, LCLC will misclassify some examples (typically minority class examples within individual clusters), ultimately affecting the performance of the classifier trained on these mis-assigned examples. The errors introduced will be especially costly for those examples located in the boundary clusters between positive and negative classes.

Secondly, as LCLC algorithm uses *K-means* algorithm to perform clustering, it will generate different clusters based on *K* randomly initialized centroids. It is highly possible that the examples near the positive and negative boundary will be grouped into different clusters and assigned with different labels each time we perform LCLC algorithm. This source of random errors can introduce further limitations in the overall performance of LCLC.

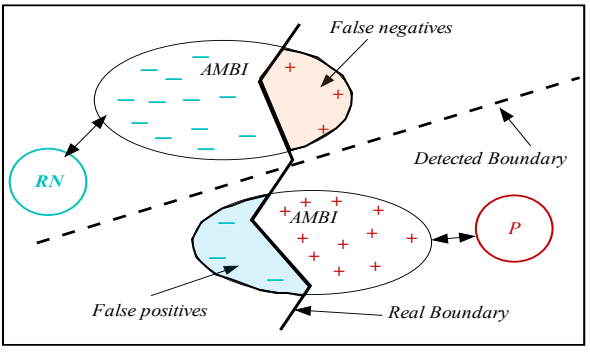


**Fig. 2.** Drawback of the LCLC on fixed clustering assignment to every instance

Figure 2 depicts the scenario of having possible misclassified instances in the local clusters near the real positive and negative boundaries. We can see that some instances may be clustered wrongly and some assigned with wrong labels based on its container cluster's label. The region represented by dashed green (orange) areas shows the set of false negative (positive) examples. The probability to be misclassified is high for those instances that are close to the class boundaries, leading to decreased accuracy of the final classification. This motivates us to propose a more robust approach to address the issue.

## 3 The Proposed Technique En-LCLC

In this section, we present our proposed En-LCLC algorithm (Ensemble based Learning from Common Local Clusters) to overcome the drawbacks in the original LCLC. We propose an ensemble based approach that restarts LCLC algorithm multiple times to assign labels for each instance in the unlabeled data. We then generate an integrated "soft" probabilistic label to the examples based on their classification results from diverse classifiers. Following that, we filter and remove uncertain instances. We then design an Adaptive Fuzzy Nearest Neighbor (AFNN) classifier to train on the enhanced dataset that have been assigned with clean soft labels to better reveal the ground truths' positive and negative boundaries.

### 3.1 Probabilistic Soft Labeling Using Diverse Classifiers

We make use of the randomness of clusters generated by *K-means* clustering used in the LCLC algorithm to create diverse classifiers. We perform $n$ times LCLC algorithm in which each *K-means* clustering with random initialization will produce different cluster settings for constructing cluster chains and deciding the boundary clusters. Each time corresponds to a different classifier setting. We use the probability distribution of the $n$ labels generated by $n$ diverse classifiers for each example as its "soft" labels. By integrating them together probabilistically, we minimize the potential bias of individual LCLC prediction and the expected errors in the extracted likely positive/negative examples by our ensemble-based approach can be expected to be reduced.

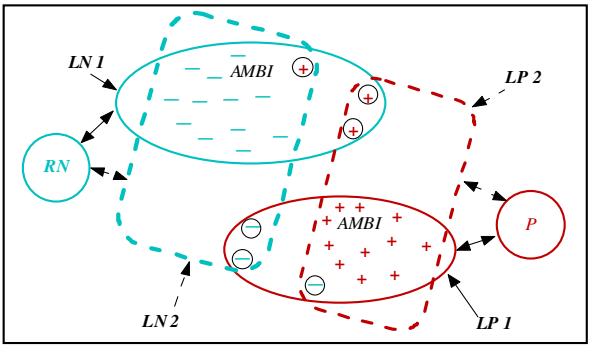


**Fig. 3.** An example of En-LCLC by repeating LCLC on two different clustering configurations

Figure 3 illustrates a scenario in which we have two classifiers with different cluster settings (i.e. $n$=2). In the first cluster setting for classifier 1, *K-means* clustering has generated two ambiguous clusters denoted by *LP1* and *LN1*. A cluster chain is constructed from *P* through *LP1*, *LN1*, and finishes at *RN*, and LCLC eventually determined all the examples in *LP1* as positive and all the examples in *LN1* as negative class. Note that there are three positive and three negative instances that

are misclassified in this case. In the second cluster setting for classifier 2, *K-means* clustering generated two ambiguous clusters, denoted by *LP2* and *LN2*; and the corresponding cluster chain goes through *P, LP2*, *LN2*, and ends at *RN*. The eventual LCLC labels for the examples in *LP2* are positives and the examples in *LN2* are negatives. In this case, only 1 positive and 1 negative are misclassified. With a big *n*, we can integrate the results together probabilistically and give the misclassified examples low confidence scores as they will be given inconsistent labels using different classifiers.

Algorithm 2 shows the details to generate the confidence score for each example in the unlabeled set *U-P* (*AMBI* clusters and *RN* clusters). Note that we have given a confidence score of 1 for all the positive examples in *P*.

**Algorithm 2.** Confidence score generation
**Input**: one initial seed positive *s*, unlabeled dataset *U*, number of iterations *n*

1. Use Wei's method to get an initial positive set *P*;
2. Repeat step 3 to step 5 *n* times
3. Partition the remaining unlabeled data *U - P* into *K* unlabeled local clusters using *K-means* clustering with random initialization;
4. Perform LCLC (cluster chain breaking) to extract the reliable negative clusters *RN*, likely positive clusters *LP*, likely negative clusters *LN*;
5. Compute the weights for each instance using equation (1) and (3);
6. Compute the normalized confidence score for each instance using equation (4) and (5);

We need to cater for cases where only a small number of positives are available for learning, even in the extreme scenario of having only one seed positive example. As such, similar with LCLC, in the first step, we adopt Wei's method [14] for this task as follows. Given the positive seed *s*, we add the next most confident positive instances from *U* until the stopping criterion is reached, that is, when there is a drop of the *minimal nearest neighbor distance*. Wei's method uses this early stopping criterion because when a negative example is mistakenly added into *P*, there is a high chance that we will keep adding more negative examples, for the negative space is expected to be much denser than the positive space [14]. While this method tends to provide an early stop instead of proceeding to find the actual boundary between the positives and negatives, we observe that it is useful for constructing a robust positive set *P* with very high precision. In other words, we can obtain a "pure" positive set *P* which is still reasonably bigger than the original one seed positive example set to work with.

We repeat *n* times Steps 3 to 5 with different initializations for *K-means* clustering to create an ensemble system with *n* diverse classifiers. In Step 4, instead of assigning a "hard" label to all the instances within a cluster, we assign a "soft" probabilistic label to each instance according to its contribution to the container cluster. In particular, given an instance $x_i$, its probability $P_{x_i}$ to be labeled as its belonging cluster $C_{x_i}$'s label is defined using Gaussian distribution as:

$$P_{x_i} = \exp\left(-\frac{1}{2}\frac{dist(centroid(C_{x_i}), x_i)^2}{\sigma^2}\right) \tag{1}$$

where *centroid* ( $C_{x_i}$ ) is the cluster $C_{x_i}$ 'centroid, $\sigma$ is the *width* of the Gaussian distribution which is defined as the standard deviation of distances between all the instances in $C_{x_i}$ to *centroid*( $C_{x_i}$ ), i.e.

$$\sigma = \sqrt{\frac{1}{|C_{x_i}|}\sum_{i=1}^{|C_{x_i}|}\left[dist\left(centroid(C_{x_i}), x_i\right) - mean(dist)\right]^2}, x_i \in C_{x_i} \tag{2}$$

The probability $P_{x_i}$ denotes the possibility that an instance $x_i$ has its container cluster's label. If it is near its cluster centroid *centroid*( $C_{x_i}$ ), then it has a higher probability to belong to cluster's label; otherwise, it will be given a lower probability.

The probability is converted as a weight for each instance depending on whether it is extracted into the positive or negative class:

$$w_{x_i}^j = \begin{cases} P_{x_i} & \text{if its container cluster is } P \text{ or } LP \\ -P_{x_i} & \text{if its container cluster is } LN \text{ or } RN \end{cases} \tag{3}$$

Since we perform LCLC $n$ times, we use $j$ ($1 \le j \le n$) to indicate that the weight is given in $j$-th iteration of En-LCLC algorithm.

Step 6 computes the confidence score that an instance is extracted as either positive or negative, which is simply defined as follows:

$$\lambda_i = \sum_{j=1}^{n} w_{x_i}^j \tag{4}$$

Equation (4) basically sums all the weights over $n$ iterations into a consolidated score. An instance will get a bigger positive (negative) value if it is extracted as a positive (negative) consistently. On the other hand, if it is assigned a near zero value (no matter positive or negative), it is an unreliable instance which may not be very useful for further classification step.

We compute the confidence score of each instance normalized to the range of [-1,1] as follows:

$$\lambda_{nor,i} = \frac{\lambda_i - \frac{1}{2}(\max(\lambda_i) + \min(\lambda_i))}{\frac{1}{2}(\max(\lambda_i) - \min(\lambda_i))} \tag{5}$$

Each instance $x_i$ in *U-P* will have a confidence score $\lambda_{nor,i}$ that indicates its propensity to be a positive or negative instance. The instances in *U-P* (all with normalized confidence scores) together with *P* (all with a fixed confidence score of 1) may serve as the training set *TRAIN* for learning a classifier.

Note that the training data *TRAIN* may still contain some uncertain instances which have low confidence scores. Before building our final classifier, we perform

noise-filtering pass [20] to further remove the possibly incorrectly labeled instances. To do so, for each example $x_i$, we define its adaptive neighborhood $N(x_i)$ which consists of its *minimal* number of nearest neighbors whose total confidence score is larger or equal to 1 (which indicates we have enough information from the neighbors to make accurate decision). If $sign(\lambda_{nor,i}) \neq sign(\sum_{j=1}^{|N(x_i)|} \lambda_{nor,j})$, or the instance will be misclassified by its nearest neighbors within its adaptive neighborhood (label inconsistent), then we will remove the confusing instance from our training set *TRAIN*.

**Algorithm 3.** Adaptive Noise-filtering procedure
**Input**: Training set *TRAIN* with normalized confidence score $\lambda_{nor,j}$ for each instance in *TRAIN*

1. Sort the instances in *TRAIN* by their normalized confidence score in the decreasing order;
2. **For** all $x_i \in TRAIN$ do
3. $K=1$;
4. Find the nearest neighbor $x_j$, and add it to $N(x_i)$;
5. **While** $\sum_{j=1}^{K} | \lambda_{nor,j} | < 1$
6. $K=K+1$;
7. Find $K$th nearest neighbor $x_j$ and add it to $N(x_i)$;
8. **If** $sign(\lambda_{nor,i}) \neq sign(\sum_{j=1}^{|N(x_i)|} \lambda_{nor,j})$
9. $removalflag(x_i)=1$;
10. **For** all $x_i \in TRAIN$ do
11. **If** ($removalflag\ (x_i) == 1$)
12. $TRAIN = TRAIN - \{ x_i \}$;

Algorithm 3 shows the detailed step of this noise filtering process. For each example in training set, Steps 3-7 find its adaptive neighborhoods and Steps 8-9 detect if it is the noisy instance. Finally, we remove all these noisy instances from the training set in Steps 10-12.

### 3.2 Combining Classifiers Using Adaptive Fuzzy Nearest Neighbor Method

Traditional time series data classification often employed the k-nearest neighbor (KNN, especially 1-NN) method to build final classifier based on the given "hard" labeled training data [18]. Our approach is able to generate more refined "soft" labeled training data with confidence scores. Instead of applying a fixed threshold (which is difficult to determine for arbitrary datasets) on the confidence scores to provide "hard" labeling of the training data to enable employing the conventional KNN approaches, we will build an adaptive fuzzy nearest neighbor classifier as it can effectively make use of the normalized confidence scores of the training instances.

The detailed algorithm for building our Adaptive Fuzzy Nearest Neighbor (AFNN) classifier is shown in Algorithm 4. Steps 1-10 classify all the test instances based on their prediction values (Step 7), which are the accumulated normalized confidence scores of the nearest neighbors from the adaptive neighborhood. Steps 8-10 classify a test instance based on the sign of its prediction value.

**Algorithm 4.** Adaptive Fuzzy Nearest Neighbor
**Input**: Training set *TRAIN*, Test set *TEST*, $\lambda_{nor,j}$ for each instance $x_i$ in *TRAIN*

1. **For** all $x_i \in TEST$ do
2. $K=1$;
3. Find the nearest neighbor $x_j$, $x_j \in TRAIN$;
4. **While** $\sum_{j=1}^{K} | \lambda_{nor,j} | < 1$
5. $K=K+1$;
6. Find $K$th nearest neighbor $x_j$, $x_j \in TRAIN$;
7. Compute the prediction value: $Pre(x_i) = \sum_{j=1}^{K} \lambda_{nor,j}$;
8. **If** $Pre(x_i) \geq 0$
9. Label $x_i$ as positive;
10. **Else** Label $x_i$ as negative.

## 4 Empirical Evaluation

We compare our proposed technique En-LCLC algorithm against three existing state-of-the-art PU learning methods for time series classification, namely, Wei's method [14], Ratanamahatana's method (denoted as Ratana's method) [15] as well as the LCLC method [16].

### 4.1 Experimental Data, Settings and Evaluation Metric

Similar to the experiments reported in [14] and [16], we have performed our empirical evaluation on the five diverse time series datasets across different fields from [21] and the UCR Time Series Data Mining archive [22] to facilitate comparison. The details of the datasets are shown in Table 1.

**Table 1.** Datasets used in the evaluation experiments

| Name | Training set | | Testing set | | Num of Features |
|---|---|---|---|---|---|
| | Positive | Negative | Positive | Negative | |
| **ECG** | 208 | 602 | 312 | 904 | 86 |
| **Word Spotting** | 109 | 796 | 109 | 796 | 272 |
| **Wafer** | 381 | 3201 | 381 | 3201 | 152 |
| **Yoga** | 156 | 150 | 156 | 150 | 428 |
| **CBF** | 155 | 310 | 155 | 310 | 128 |

In our empirical evaluation, we repeated the experiments performed in [14] and [16] by randomly selecting just one seed example from the positive training class for the learning phase (i.e. seed $s$ in Algorithm 2), with the rest of the training set (both positives and negatives) treated as unlabeled data (ignoring their labels; $U$ in Algorithm 2).

We repeat our experiments 10 times with different initial seed positive instances for each dataset and report the average values of the 10 results. Since our proposed En-LCLC performs $n$ diverse classifiers to generate the confidence scores for each instance, we set $n = 15$ for this work. We will also evaluate $n$'s sensitivity later.

We use the F-measure to evaluate the performance of the four PU learning techniques. The F-measure is the harmonic mean of precision ($p$) and recall ($r$), and it is defined as $F=2*p*r/(p+r)$. In other words, the F-measure reflects an average effect of both precision and recall. F-measure is large only when both precision and recall are good. This is suitable for our purpose to accurately classify the positive and negative time series data. Having either too small a precision or too small a recall is unacceptable and would be reflected by a low F-measure.

### 4.2 Experimental Results

Table 2 shows the overall classification results of all the four techniques. The results showed that both LCLC [16] and En-LCLC performed much better than the other two earlier methods for time series classification, namely, Wei's method [14], and Ratana's method [15]. Our proposed En-LCLC produced the best classification results across all the 5 datasets, achieving F-measures of 86.9%, 76.2%, 77.5%, 89.1% and 81.6%, which are 0.2%, 3.5%, 5.1%, 3.7% and 11.5% higher than the second best results from LCLC. On average (last row), En-LCLC was able to achieve 82.3% F-measure, which is 4.8% higher than the second best LCLC method in terms of F-measure, indicating that En-LCLC is indeed well-designed for time series data classification.

**Table 2.** Overall performance of various techniques

| Dataset | Wei's method | Ratana's method | LCLC | En-LCLC |
|---|---|---|---|---|
| **ECG** | 0.405 | 0.840 | 0.867 | **0.869** |
| **Word Spotting** | 0.279 | 0.637 | 0.727 | **0.762** |
| **Wafer** | 0.433 | 0.080 | 0.724 | **0.775** |
| **Yoga** | 0.466 | 0.626 | 0.854 | **0.891** |
| **CBF** | 0.201 | 0.309 | 0.701 | **0.816** |
| **Average** | 0.357 | 0.498 | 0.775 | **0.823** |

Recall that En-LCLC has two key steps for our classification task: (1) using "soft" Adaptive Fuzzy Nearest Neighbor classifier replace the "hard" label 1-NN classifier, and (2) performing a noise filtering before building our final classifier. To determine their individual effects on our classification performance, we also test the En-LCLC algorithm without these key steps. The results are shown in Table 3.

**Table 3.** En-LCLC with 1-NN and without noise filtering

| Dataset | ECG | Word Spotting | Wafer | Yoga | CBF |
|---|---|---|---|---|---|
| **En-LCLC w 1-NN** | 0.867 | 0.736 | 0.745 | 0.861 | 0.754 |
| **En-LCLC w/o noise filtering** | 0.868 | 0.749 | 0.764 | 0.879 | 0.792 |
| **En-LCLC** | **0.869** | **0.762** | **0.775** | **0.891** | **0.816** |

Without using "soft" AFNN classifier, we use 1-NN by converting the confidence scores into "hard" labels using the following procedure: if the confidence score of an unlabeled instance is larger than 0, then it is regarded as a positive training example; otherwise, it is treated as a negative training example. We observe from Table 3 that En-LCLC with 1-NN is on average 1.8% worse than our proposed En-LCLC. It is important to note that En-LCLC with 1-NN has already benefited from the more accurate confidence scores obtained from our ensemble-based strategy. Similarly, by including noise filtering, En-LCLC is able to perform 1.22% higher on average, indicating that removing those potentially noisy examples using the confidence scores can contribute to enhance the classification performance.

Table 4 compares the performance of the LCLC and En-LCLC techniques for extracting positives and negatives from unlabeled data. In the table, *Ext_P* (*Ext_N*) represents the number of positives (negatives) extracted. *Error_P* and *Error_N* represent the error rate of positive and negative extraction respectively. Compared with LCLC, En-LCLC made 0.40%, 6.80%, 5.60%, 1.90% and 1.90% less errors for positive extraction over the 5 datasets. It also made 0.10%, 1.10%, 0.50%, 4.10% and 5.90% less errors for negative extraction over 5 datasets. As these instances are located near the positive and negative boundary, the reduction in false positives and false negatives helped improve En-LCLC's eventual accuracy.

**Table 4.** Extraction comparison between LCLC and En-LCLC

| Dataset | | ECG | Word Spotting | Wafer | Yoga | CBF |
|---|---|---|---|---|---|---|
| **LCLC** | *Ext_P* | 234.3 | 139.4 | 269.7 | 166.8 | 106.6 |
| | *Error_P* | 16.1% | 36.3% | 16.6% | 17.3% | 12% |
| | *Ext_N* | 575.7 | 765.6 | 3312.3 | 139.2 | 358.4 |
| | *Error_N* | 2% | 2.6% | 4.7% | 12.9% | 17.1% |
| **En-LCLC** | *Ext_P* | 233 | 134.8 | 271.4 | 165.7 | 118.6 |
| | *Error_P* | 15.7% | 29.5% | 11% | 15.4% | 10.1% |
| | *Ext_N* | 574.3 | 745.6 | 3299.8 | 134.8 | 321 |
| | *Error_N* | 1.9% | 1.5% | 4.2% | 8.8% | 11.2% |

We now analyze the efficiency of En-LCLC algorithm. The main steps for our algorithm include *K-means* clustering to partition the unlabeled data, compute the confidence score for each example, perform noise filtering, and construct Adaptive Fuzzy Nearest Neighbor (AFNN) classifier. These steps can be performed using efficient algorithms implemented in linear time. Note that our ensemble-based

approach will not increase the overall time complexity although it performs $n$ times classification ($n$ is typically a relatively small integer). Our proposed algorithm is therefore not less efficient than the other methods. Figure 4 shows that our proposed En-LCLC method can scale well over all the datasets (we implemented our En-LCLC algorithm using Matlab and running on the Desktop Intel Pentium IV core 2 Duo 3.0 GHz CPU using Microsoft Windows XP with 4.0 GB memory).

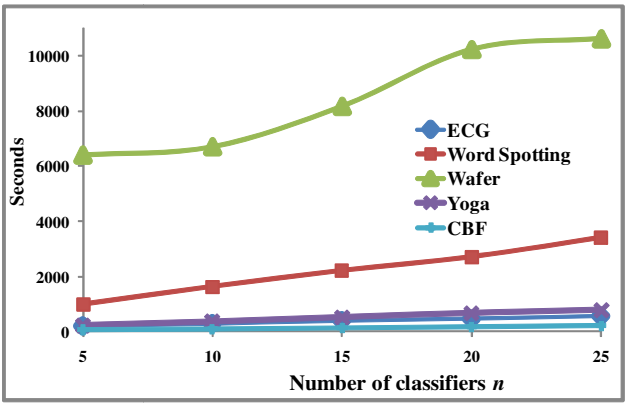


**Fig. 4.** Efficiency analysis of the number of classifiers $n$

Finally, we evaluate the effect of parameter $n$, the number of classifiers, which is the only one parameter of our method. In this paper, we have set $n$ = 15. We repeated our experiments with different values of $n$ from 5 to 25, with a step of 5. Figure 5 shows the sensitivity of En-LCLC with different values of $n$. We can see that with the increasing value of $n$ from small values, the F-measure also increased for 4 datasets (interestingly, the parameter $n$ does not affect the performance of ECG data much). When $n$>= 15, the performance has reached steady status, without much changes, suggesting that we do not need a very large $n$ to benefit from ensemble based strategy.

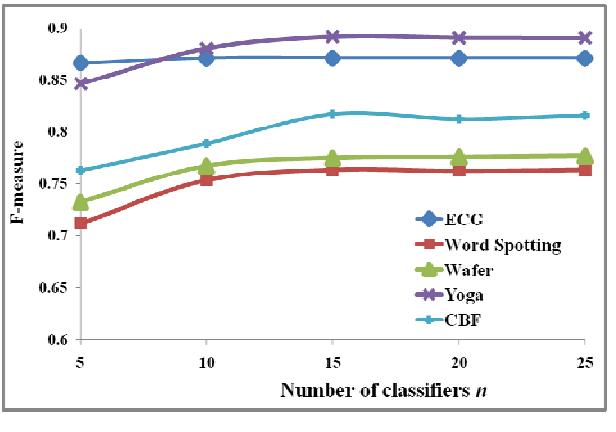


**Fig. 5.** Sensitivity analysis of the number of classifiers $n$

## 5 Conclusions

Time series classification has been applied in many real-world applications across different domains. In this paper, we study the positive unlabeled learning method as it eliminates the tedious and costly process to hand-label large amounts of training data. We proposed a novel En-LCLC algorithm (Ensemble based Learning from Common Local Clusters) to overcome the drawbacks of the existing state-of-the-art LCLC algorithm. Our proposed En-LCLC algorithm adopts an ensemble-based strategy which performs LCLC algorithm multiple times to minimize the potential bias of individual LCLC prediction. As shown in our experiments, the error rates in the extracted likely positive/negative examples has been effectively reduced. We designed an Adaptive Fuzzy Nearest Neighbor classifier to exploit the "soft" confidence scores obtained from the diverse classifiers. We also made use of the confidence scores to remove unreliable examples from the training dataset. The experimental results show that our proposed method performed much better than existing methods over multiple time series data from different domains.

## References

[1] Olszewski, R.T.: Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data, PhD thesis, Carnegie Mellon University, Pittsburgh, PA (2001)
[2] Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II-521–II-527 (2003)
[3] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd International Conference on Machine Learning. ACM, Pittsburgh (2006)
[4] Chapelle, O., Scholkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press (2006) (in Press)
[5] Li, M., Zhou, Z.-H.: SETRED: Self-Training with Editing. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 611–621. Springer, Heidelberg (2005)
[6] Zhu, X.: Semi-supervised learning literature survey, Technical report, no.1530, Computer Sciences, University of Wisconsin-Madison (2008)
[7] Liu, T., Du, X., Xu, Y., Li, M.-H., Wang, X.: Partially Supervised Text Classification with Multi-Level Examples. In: AAAI (2011)
[8] Gabriel Pui Cheong, F., Yu, J.X., Hongjun, L., Yu, P.S.: Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering 18, 6–20 (2006)
[9] Li, X., Liu, B.: Learning to classify texts using positive and unlabeled data. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc., Acapulco (2003)
[10] Li, X., Liu, B., Ng, S.-K.: Learning to Identify Unexpected Instances in the Test Set. In: Proceedings of Twentieth International Joint Conference on Artificial Intelligence, India (IJCAI 2007), pp. 2802–2807 (2007)
[11] Li, X., Yu, P., Liu, B., Ng, S.-K.: Positive Unlabeled Learning for Data Stream Classification. In: SDM, pp. 257–268 (2009)

[12] Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially Supervised Classification of Text Documents. In: ICML (2002)
[13] Elkan, C., Noto, K.: Learning Classifiers from Only Positive and Unlabeled Data. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2008)
[14] Wei, L., Keogh, E.: Semi-supervised time series classification. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, Philadelphia (2006)
[15] Ratanamahatana, C., Wanichsan, D.: Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification. In: Lee, R. (ed.) Soft. Eng., Arti. Intel., Net. & Para./Distri. Comp. SCI, vol. 149, pp. 1–14. Springer, Heidelberg (2008)
[16] Nguyen, M.N., Li, X., Ng, S.-K.: Positive Unlabeled Learning for Time Series Classification. In: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI (2011)
[17] Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 881–892 (2002)
[18] Keogh, E., Kasetty, S.: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. Data Mining and Knowledge Discovery 7, 349–371 (2003)
[19] Yoon, H., Yang, K., Shahabi, C.: Feature subset selection and feature ranking for multivariate time series. IEEE Transactions on Knowledge and Data Engineering 17, 1186–1198 (2005)
[20] Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man and Cybernetics 2, 408–421 (1972)
[21] Wei, L.: Self Training dataset (2007), http://alumni.cs.ucr.edu/~wli/selfTraining/
[22] Keogh, E.: The UCR Time Series Classification/Clustering Homepage (2008), http://www.cs.ucr.edu/~eamonn/time_series_data/