Positive Unlabeled Learning for Time Series Classification

Minh Nhut Nguyen, Xiao-Li Li, See-Kiong Ng
Institute for Infocomm Research, Singapore
{mnnguyen, xlli, skng}@i2r.a-star.edu.sg

Abstract

In many real-world applications of the time series classification problem, not only could the negative training instances be missing, the number of positive instances available for learning may also be rather limited. This has motivated the development of new classification algorithms that can learn from a small set *P* of labeled seed positive instances augmented with a set *U* of unlabeled instances (i.e. PU learning algorithms). However, existing PU learning algorithms for time series classification have less than satisfactory performance as they are unable to identify the class boundary between positive and negative instances accurately. In this paper, we propose a novel PU learning algorithm LCLC (Learning from Common Local Clusters) for time series classification. LCLC is designed to effectively identify the ground truths' positive and negative boundaries, resulting in more accurate classifiers than those constructed using existing methods. We have applied LCLC to classify time series data from different application domains; the experimental results demonstrate that LCLC outperforms existing methods significantly.

1 Introduction

Time series data are commonly found in many application domains including multimedia, medicine, aerospace, finance, manufacturing and entertainment (Olszewski 2001; Rath and Manmatha 2003; Xi, Keogh et al. 2006). A key task is to classify the time series data into predefined categories. For example, a typical application in manufacturing would be to classify the machines into normal or faulty statuses based on time series sensor readings.

Classification methods that simply apply traditional supervised learning techniques on the time series data rely on having large amounts of labeled examples for learning accurate classifiers. In practice, collecting and labeling large sets of training data could be very expensive and sometimes even impossible. As such, alternative learning techniques have been proposed to build classifiers from a small amount of labeled training data and a large set of unlabeled data which are typically available for learning. These methods include semi-supervised learning (Li and Zhou 2005; Chapelle, Scholkopf et al. 2006; Zhu 2008) and PU learning (Li and Liu 2003; Li, Liu et al. 2007; Li, Yu et al. 2009). While both approaches exploit the unlabelled data (*U*) to enhance the performance of their classifiers, they differ in the training data requirement as PU learning only requires positive data (*P*). As a result, PU learning is applicable in a wide range of application domains, e.g. text classification, biomedical informatics, pattern recognition, and recommendation system, as negative data are often unavailable in the real-world applications. However, due to the various challenges of time series classification such as high feature correlation (Wei and Keogh 2006), the application of PU learning to classify time series data has been relatively less explored than other data classification tasks.

Recently, Wei and Keogh (Wei and Keogh 2006) proposed a PU learning approach that iteratively expands the initial positive examples using those unlabelled data that are most similar to them, with the remaining unlabelled data being extracted as negative data. Unfortunately, without a good stopping criterion, the method often stops too early with an expansion of a small number of positives. The corresponding negatives being extracted are therefore not pure, resulting in classifiers with limited accuracy. To improve the algorithm, a more recent method (Ratanamahatana and Wanichsan 2008) attempted to propose a good stopping criterion by using the historical distances between candidate examples from *U* to the initial positive examples. Although more positive examples can be extracted, the method is unable to extract accurate positives and hence negatives from *U*. The resulting classifier therefore fails to identify the ground truth's boundary between positive and negative data. The experimental results reported showed that the classification results had high precision but low recall.

In this paper, we propose an effective technique called LCLC (Learning from Common Local Clusters) to tackle the challenge of constructing a robust classifier for time series classification using only limited labeled positive data. The proposed method first partitions the unlabeled set *U* into small unlabeled local clusters (*ULC*). It treats each cluster as an observed variable in which all the data belonging to the cluster share the same principal component and have the same class value. Our method will learn the local clusters' common principal features in order to choose *independent* and *relevant* features for classification. Next, LCLC automatically extracts the hidden positive clusters and negative clusters from *U*. Our cluster-based approach is much more reliable than existing PU learning methods that extract negative and positive instances point by point. We also propose a novel *cluster chain* approach which is critical for contributing to accurate cluster extraction.

The rest of the paper is organized as follows. First, we provide an overview on the related works of PU learning in Section 2. We then present our proposed LCLC algorithm and its experimental results in Sections 3 and 4 respectively. Extensive experiments on time series data across diverse fields show that classifiers built using LCLC can accurately identify the ground truths' positive and negative boundaries, thereby significantly outperforming the existing state-of-the-art PU learning methods for time series classification. Finally, Section 5 concludes the paper.

2 Related Work

As mentioned, traditional supervised learning methods rely on having sufficient positive and negative training examples available for learning. In practice, the negative examples can often be limited or unavailable. This has motivated the development of the model of *learning from positive and unlabeled examples*, or PU learning, where *P* denotes a set of positive examples, and *U* a set of unlabeled examples (which contains both hidden positive and hidden negative instances). The PU learning problem is therefore to build a classifier using *P* and *U* in the absence of negative examples to classify the data in a future test data *T*.

PU learning has been investigated by numerous researchers in the past decade or so. A study of PAC learning for the PU setting under the statistical query model was first given in (Denis 1998). An extension of positive naive Bayes learning for probabilistic problems could be found in (Borja, Pedro et al. 2007). (Liu, Lee et al. 2002) reported the sample complexity result and showed how the problem may be solved. Subsequently, a number of practical algorithms (Liu, Lee et al. 2002; Li and Liu 2003) were proposed. They generally follow a two-step strategy: (i) identifying a set of *reliable negative* documents *RN* from the unlabeled set *U*, and then (ii) building a classifier with the positive set *P*, the reliable negative set *RN* and the new unlabelled set *U*' (*U*'=*U*-*RN*), using EM, or running SVM iteratively. Please refer to (Liu 2007) for the details of more recently developed PU learning algorithms.

To our best knowledge, thus far only two PU learning approaches (Wei and Keogh 2006; Ratanamahatana and Wanichsan 2008) have been proposed for classifying time series data. However, as we have discussed in the previous section, the current methods are unable to accurately extract the reliable positive and negative examples from *U*, resulting in inaccurate classifiers. Given that time series data are commonly found in many application domains, we are thus motivated to explore devising better PU learning techniques for time series data classification.

3 The Proposed Technique

In this section, we present our proposed LCLC algorithm. Our method is designed to address two specific issues in PU learning for time series classification: how to select independent and relevant features from the time series data, and how to accurately extract reliable positive and negatives from the given unlabelled data. We design our algorithm to build robust classifiers with a small number of positives, even in the extreme scenario of having only one seed positive example.

3.1 Local Clustering and Feature Selection

Clearly, the single seed positive example set is too small to represent the positive feature space. We need to include additional high-confidence positive examples from *U* to construct an initial positive set *P*. In this paper, we adopt Wei's method (Wei and Keogh 2006) for this task. Given the seed positive *s*, we add the next most confident positive instances from *U* until the stopping criterion is reached, namely, there is a drop of the *minimal nearest neighbor distance*. Wei's method uses this early stopping criterion because if a negative example were to be mistakenly added into *P*, there is a high chance that we will keep adding more negative examples as the negative space is much denser than the positive space (Wei and Keogh 2006). 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 positive set *P* with very high precision. In other words, we can obtain a "pure" positive set *P* that is reasonably bigger than the original one seed positive example set to work with.

Next, we partition the remaining unlabeled data *U* ($U = U - (P - \{s\})$) into small local clusters ULC_i (i=1, 2, …, K) using *K-means* clustering method (Kanungo, Mount et al. 2002). Each local cluster ULC_i is treated as an observed variable of the time series data, and we assume that all the instances belonging to a local cluster share the same principal component and have the same class label. We use these local clusters for two purposes: 1) Instead of extracting individual likely positive and negative examples as in previous PU learning methods, we extract the likely positives and negatives based on the clusters (See Subsection 3.2). 2) Given that time series data have high feature correlation, we can exploit the clusters for feature selection, choosing a subset of common principal features from the raw feature set that better captures the underlying characteristics of the time series data set.

We find that this novel cluster-based approach of feature selection together with the robust cluster-based extraction of likely positives and negatives from the unlabelled test set help to build more accurate classifiers. Recall that the similarity between two time series data records can be effectively measured by comparing their principal components of the corresponding features (Yoon, Yang et al. 2005; Hyunjin and Cyrus 2006). A well-selected subset of the common principal features can capture the underlying characteristics of the time series data set to enable accurate extraction of the remaining *hidden* positives/negatives from *U*, subsequently identifying the boundary between positive and negative data. In this work, we use the *Clever-Cluster* method (Yoon, Yang et al. 2005) for selecting a common feature subset from the positive set *P* (a "pure" positive local cluster generated using Wei's method) and a partitioned coherent unlabelled cluster ULC_i as follows.

Let X_i be the correlation matrix of the ith local cluster ULC_i (i=1, 2, …, K) and X_{K+1} be the correlation matrix of positive cluster P. We first compute the principal components (PCs) matrix L_i for all correlation matrixes ($X_1, X_2, \ldots, X_{K+1}$) by applying *Singular Value Decomposition* (SVD):

$$SVD(X_i) = A_i \Lambda A_i^T, \; (i=1, 2, \ldots, K+1) \tag{1}$$

Suppose we have n raw features in our time series data. Although there are n PCs for each cluster, for feature selection, only the first p ($p < n$) PCs are taken into consideration. Each cluster can thus be represented as a $p \times n$ matrix in which the p rows are its first p principal components and columns represent the feature indexes. In our implementation, p is chosen at 80%*n as recommended in (Yoon, Yang et al. 2005). Let the integration matrix H of these clusters be

$$H = \sum_{i=1}^{K+1} L_i^T L_i \tag{2}$$

The descriptive common principal components ($DCPC$s) that agree most closely with *all* the local clusters are then defined by the eigenvectors of H, i.e.

$$SVD(H) = SVD\left(\sum_{i=1}^{K+1} L_i^T L_i\right) = V \Lambda V^T \tag{3}$$

where the rows of V are the eigenvectors of the integration matrix H and the first p of them define the $DCPC$s for our K+1 clusters. Λ is a diagonal matrix whose diagonal elements are the eigenvalues of H. They describe the total discrepancy between $DCPC$s and PCs. Just like the PC matrix of a local unlabelled cluster and positive cluster, the p rows of this $DCPC$ matrix are the first principal components, while the columns still represent the n raw feature indexes, and the (i,j)th element is a $DCPC$ loading of the jth variable to the ith $DCPC$. The correspondence to the original raw features is still retained in the columns of the $DCPC$s matrix with each raw feature now represented as a column vector of its $DCPC$ loadings. Algorithm 1 below provides the detailed steps for selecting such a subset of common principal features.

Algorithm 1. **Local clustering and feature selection**
Input: one initial seed positive s, unlabelled dataset U, number of clusters K

```
1. Use Wei's method to get an initial positive set $P$;
2. Partition the remaining unlabeled data $U$ - ($P$ - {$s$}) into
   $K$ unlabeled local clusters using Kmeans clustering;
3. Compute principal components for positive cluster $P$,
   and each unlabelled cluster $ULC_i$ into a principal matrix
   $L_i$ for ($i$=1, .. , $K$+1);
4. $H = \sum_{i=1}^{K+1} L_i^T L_i$
5. Compute the descriptive common principal components
   (DCPCs) on matrix $H$;
6. Compute raw feature vector of its DCPC loadings;
7. Perform K-means clustering algorithm to cluster the raw
   feature vectors (DCPC loadings) into $K$ feature groups
8. Select $K$ features ($v_1, v_2, \ldots, v_K$) from the raw feature set
```

Note that we have used *K-means* clustering algorithm again to cluster the raw feature vectors ($DCPC$ loadings) into K feature groups in which the related raw features are grouped together. Since *K-means* could reach local minima, we repeat the *K-means* clustering 10 times and choose the one with the minimum sum of Euclidean distances between each cluster's centroid and its component feature vectors within the cluster as the best clustering result. Finally, the K highest mutual information features (Cai, He et al. 2005) are selected from these clusters as our final feature set. Specifically, for each cluster, we choose the one feature closest to the centroid of that cluster. The intuition behind such selection is based on the observation that features with similar patterns of loading values will be highly correlated and have high mutual information (Yoon, Yang et al. 2005).

3.2 Building the Final Classifier

There have been many algorithms proposed for time series classification; most of them are traditional supervised learning based techniques. Keogh *et al* (Keogh and Kasetty 2003) performed a comprehensive empirical evaluation on the current state-of-the-arts. Interestingly, the simple technique K-NN (specifically, 1-NN, i.e. classification based on the top one nearest neighbor) with Euclidean distance was shown to be the best technique. These supervised learning methods, including 1-NN, require both positive and negative training sets for learning. In the PU learning setting, there is therefore the need to extract a *reliable negative* set RN from the unlabeled set U as we only have positive set P.

Extracting Reliable Negative set *RN*

It is important to extract a RN according to the following two criteria: (1) The RN should cover as many diverse negative instances as possible; and (2) the negative data extracted from the unlabeled set U should be reasonably reliable or pure, i.e., with no or very few positive instances (i.e. false negatives). The presence of *false negatives* in RN will hurt the performance of the subsequent steps of extracting the likely positives and the likely negatives. The final step to build 1-NN classifier is also very sensitive to noise.

Algorithm 2. **Extracting Reliable Negative Examples**
Input: positive data P, K unlabeled local clusters ULC_i

```
1.      $RN = \emptyset$, $AMBI = \emptyset$;
2.      For $i$=1 to $K$
3.   Compute the distance between local cluster $ULC_i$ to $P$;
4.      Sort $d(ULC_i, P)$ ($i$=1, 2, …, $K$) in a decreasing
        order;
5.      $d_{Median}$= the median distance of $d(ULC_i, P)$ ($i$=1,
        2,…, $K$);
6.      For $i$=1 to $K$
7.   If ($d(ULC_i, P) > d_{Median}$)
8.      $RN = RN \cup ULC_i$;
9.   Else
10.     $AMBI = AMBI \cup ULC_i$;
```

Algorithm 2 above shows the steps to extract the reliable negative set RN. In step 1, both RN (which is used to store the *reliable* negative data) and $AMBI$ (which is used to store the *ambiguous* data), are initialized as empty set. In steps 2 and 3, we compute the distance of each ULC_i (i=1, 2, …, K) from our positive data P using the features selected (see previous Section 3.1). The distance between ULC_i and P are defined as follows:

$$d(UCL_i, P) = \min(dist(x_{UCL}, x_P)), \quad \forall x_{UCL} \in UCL_i, \forall x_P \in P \quad (4)$$

Note that $d(ULC_i, P)$ is the minimum distance between all instances in ULC_i and all the instances in P. Let $ULC_i = (a_1, a_2, \ldots, a_K)$ and $P=(b_1, b_2, \ldots, b_K)$. The dist($x_{ULC}$, x_P) is computed using Euclidean distance between each pair of corresponding elements based on the K common selected features:

$$dist(x_{ULC}, x_P) = \sqrt{\sum_{i=1}^{K}(a_i - b_i)^2} \quad (5)$$

Steps 4 and 5 then sort all the distances $d(ULC_i, P)$ in decreasing order and compute the median distance d_{Median}. Finally, in steps 6 to 10, we extract those local clusters which are farthest away from P and store them into RN. The size of RN is set to contain a half of the local clusters, while the other half is considered as ambiguous clusters *AMBI* to be processed further in the next phase.

Identifying Likely Positive Clusters *LP* and Likely Negative Clusters *LN*

Since we now have a positive data P and reliable negative data RN, we should be able to build a classifier using any supervised learning method. However, the number of initial positive training examples in cluster P is often insufficient, and the reliable negative clusters in RN extracted in the previous step is still far away from the boundary between the actual positive and negative data. To build a robust classifier, an important next step in our LCLC algorithm is to further extract the *likely* positive clusters LP and the *likely* negative clusters LN from instances in the ambiguous clusters *AMBI* which are near the positive and negative boundary.

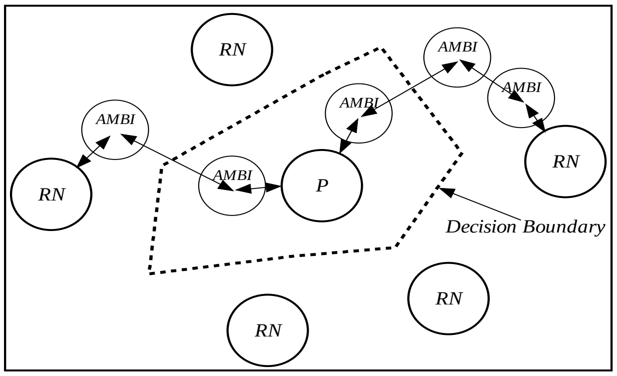

Figure 1. Cluster chain for boundary decision

We propose a novel idea to build *cluster chains.* Each cluster chain starts from the positive P, goes through one or more *AMBI* clusters, and finally stops at one reliable negative cluster in RN. Figure 1 illustrates the scenario where we have 5 five reliable negative clusters which are far away from the positive cluster, and 5 *AMBI* clusters which are located between the positive cluster and negative clusters. We show two possible cluster chains: the chain on the left-hand side consists of 4 clusters (including two *AMBI* clusters) while the chain on the right-hand side consists of 5 clusters (including three *AMBI* clusters). For each cluster chain, we find the *breaking link* (decision boundary) with *maximal distance* between the clusters and separate the cluster chain into two sub-chains. All the *AMBI* clusters within a sub-chain including P will be regarded as likely positive clusters and stored into LP, while all the *AMBI* clusters within a sub-chain including RN will be regarded as likely negative clusters and stored into LN.

We use different measures to evaluate the distances between different types of clusters. The distance between a *AMBI* cluster C_{AMBI} and the positive cluster P or a negative cluster C_{RN} ($C_{RN} \in RN$) is defined as the minimum distance from the centroid of the *AMBI* cluster to every instance of cluster P or C_{RN}, i.e.

$$d(C_{AMBI}, P) = min(dist(centroid_{AMBI}, x_p)), \; \forall x_p \in P \quad (6)$$

$$d(C_{AMBI}, C_{RN}) = min(dist(centroid_{AMBI}, x_n)), \; \forall x_n \in C_{RN} \quad (7)$$

where $centroid_{AMBI}$ is the centroid of the *AMBI* cluster. On the other hand, the distance between two *AMBI* clusters $C_{AMBI,A}$ and $C_{AMBI,B}$ is computed based on the distance between their two centroids $centroid_{AMBI,A}$ and $centroid_{AMBI,B}$:

$$d(C_{AMBI,A}, C_{AMBI,B}) = dist(centroid_{AMBI,A}, centroid_{AMBI,B}) \quad (8)$$

Note that we have used centroids instead of directly using the examples/points to compute the distance between two *AMBI* clusters. This is because compared with the relatively more reliable positive cluster P and negative clusters in RN, the *AMBI* clusters are likely to contain noisy examples which could result in inaccurate distance measure if we used individual examples. We therefore use centroids to minimize the effect of possible noisy examples.

Algorithm 3. Identifying likely positive clusters LP and likely negative clusters LN
Input: ambiguous clusters *AMBI*, positive cluster P, reliable negative clusters set in RN

```
1.  $LP = \emptyset$; $LN = \emptyset$;
2.  $i = 0$;
3.  While ($AMBI != \emptyset$)
4.      Find the nearest AMBI cluster $C_{AMBI,A}$ to $P$ and add
        $C_{AMBI,A}$ to cluster-chain$_i$ ;
5.      While $C_{AMBI,A} \notin RN$
6.          Find the nearest cluster $C_{AMBI,B}$ (from $AMBI \cup$
            $RN$) to $C_{AMBI,A}$ and add $C_{AMBI,B}$ to cluster-chain$_i$;
7.          $C_{AMBI,A} = C_{AMBI,B}$;
8.      $i = i + 1$;
9.  Loop for all the cluster-chain$_i$
10.     breaking link$_i$ ($C_m$, $C_{m+1}$)= the link with maximal dis-
        tance between the clusters in cluster-chain$_i$
11.     Separate the cluster chain cluster-chain$_i$ into two
        sub-chains;
12.     Move all the ambiguous clusters from the cluster-
        chain$_i$ with $P$ into likely positive clusters $LP$;
13.     Move all the ambiguous clusters from the cluster-
        chain$_i$ without P (or stops with a negative cluster in
        $RN$) into likely negative clusters $LN$;
```

Algorithm 3 above shows the steps to identify the likely positive clusters *LP* and negative clusters *LN*. After initializing *LP* and *LN* in step 1 as empty sets, we build the cluster chains from steps 2 to 8. Then, for each cluster-chain, we break it into two sub-chains by separating the link with maximal distance. In particular, steps 9 to 13 allocate the ambiguous clusters into positive clusters *LP* if they are located in the sub-chains that include *P*, and negative clusters *LN* if they are located in the sub-chains that include a negative cluster in *RN*. The breaking links serve as the decision boundaries for separating the positive and negative ambiguous clusters. This process (from steps 9 to 13) is repeated until we have separated all the cluster chains and all the ambiguous clusters will be put into either LP or LN.

We are now ready to build the classifier for time series classification: we use *P* together with *LP* as a positive training set, and *RN* together with *LN* as a negative training set to build our final 1-NN classifier.

4 Empirical Evaluation

In this section, we compare our proposed LCLC algorithm with two existing state-of-the-art semi-supervised methods for time series classification that we have mentioned earlier, namely, Wei's method (Wei and Keogh 2006), and Ratanamahatana's method (denoted as Ratana's method) (Ratanamahatana and Wanichsan 2008). We use the F-measure to evaluate the performance of the three 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 positives and negatives. Having either too small a precision or too small a recall is unacceptable and would be reflected by a low F-measure.

To facilitate comparison, similar to the experiments reported in (Wei and Keogh 2006), we have performed our empirical evaluation on the five diverse time series datasets across different fields from (Wei 2007) and the UCR Time Series Data Mining archive (Keogh 2008). 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
	Positive	Negative	Positive	Negative	Features
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 randomly select just one seed instance from the positive class for the learning phase, and treat it as our only initial positive seed data; the rest of training data are treated as unlabeled data (ignoring their labels) for training. Given the random nature of choosing the initial positive examples and *K-means* clustering, we repeat our experiments 30 times and report the average values of the 30 results.

Table 2 shows the classification results of the three techniques. LCLC produces the best results consistently across all the data sets, achieving F-measures of 86.7%, 72.7%, 72.4%, 85.4% and 70.1%, which are 2.7%, 9.0%, 29.1%, 22.8% and 39.2% higher than the second best results (all from Ratana's method).

LCLC works much better than the existing methods because it focuses on effectively extracting the positive data *P*, reliable negative data *RN*, as well as the likely positive (negative) data *LP* (*LN*) from *U*, so that the resulting classifier is near the optimal positive and negative boundary. On the other hand, both Wei's method and Ratana's method suffer from the early stopping strategy. As a result, their identified positives are so few that their corresponding negatives still contain a large number of hidden positives that eventually result in bad classifiers.

Table 2. Overall performance of various techniques

Dataset	ECG	Word Spotting	Wafer	Yoga	CBF
Wei's method	0.405	0.279	0.433	0.466	0.201
Ratana's method	0.840	0.637	0.080	0.626	0.309
LCLC	0.867	0.727	0.724	0.854	0.701

Table 3. LCLC without feature selection (FS) or cluster chain (CC)

Dataset	ECG	Word Spotting	Wafer	Yoga	CBF
LCLC wo FS	0.631	0.608	0.637	0.808	0.599
LCLC wo CC	0.781	0.52	0.32	0.699	0.586
LCLC	0.867	0.727	0.724	0.854	0.701

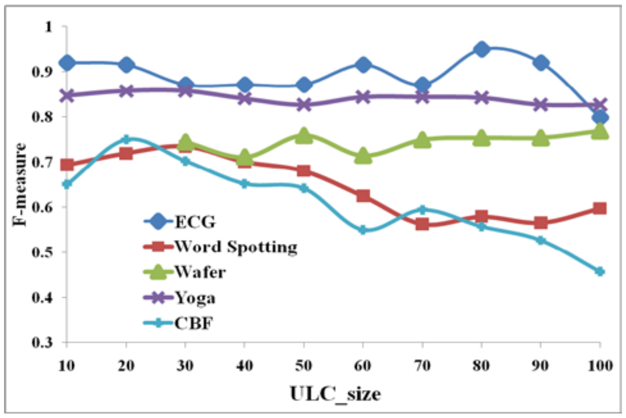

Figure 2. Sensitivity of the size of local clusters (*ULC_size*)

Recall that the cluster-based feature selection and the cluster-chain approach are two key steps for our LCLC algorithm. To determine their individual effects on our classification performance, we also test the LCLC algorithm without these key steps. The results are shown in Table 3. Without feature selection, LCLC performed poorly on all the datasets: on most of the data the classification performance dropped by 10% in F-measure (except the Yoga dataset which resulted in only a 4.6% drop). To see the effect without cluster chaining, we perform the following procedure: we use *P* and *RN* as the positive and negative to train a 1-NN classifier, and use it to extract the ambiguous clusters (*AMBI*) to construct *LP* and *LN*. If an AMBI cluster is closer to *P*, we will add it into *LP*; otherwise, it will be added into *LN*. Similarly, we train the final 1-NN classifier with *P*+*LP* as positives and *RN*+*LN* as negatives. The results shown in Table 3 show that the corresponding drop in F-measure ranges from 8.6% to 40.4%, depending on the data sets. These results verify that these two innovative LCLC steps played crucial roles for classifying the time series data.

Finally, we evaluate the effect of *K*, the one parameter of our method which is used as the number of local clusters. In this paper, we have set *K* = Size(*U*)/*ULC_size*, where *ULC_size* is a predefined number that decides the size of the unlabelled clusters. We repeat our experiments with different values of *ULC_size* from 10 to 100. Figure 2 shows the sensitivity of LCLC with different values of *K*. We can see that the parameter *ULC_size* does not affect the F-measure for all the datasets when *ULC_size* ranges from 20 to 60.

5 Conclusions

Time series classification is an important data mining task for many real-world applications. Given that it is often tedious and costly to hand-label large amounts of training data, we must consider a setting in which not only the negative training examples are absent, but the number of positive examples available for learning can also be rather limited. There are relatively few existing semi-supervised learning methods for such scenarios. In this work, we have proposed a novel PU technique LCLC that can effectively extract positive and negative data from the unlabelled set *U* for training robust time series classifiers.

There are three key approaches that underlie LCLC's improved classification performance over existing methods. First, LCLC adopts a cluster-based method that is much more robust than current PU learning methods which are typically instance-based and hence more easily prone to noise in the time series data. Secondly, we have adopted a feature selection strategy that can take the characteristics of both positive and unlabelled clusters into consideration and thus enable LCLC to evaluate the similarities between the clusters/examples much more accurately. Finally, we have devised a novel cluster chaining approach to extract the boundary positive and negative clusters. All these approaches were important in facilitating LCLC in building robust classifiers for accurate time series classification, as demonstrated by the superior experimental results over existing methods.

We noticed that the *AMBI* clusters may contain some noise and uncertainty instances. In our future work, we intend to introduce a probabilistic representation for each ambiguous instance to increase the effectiveness of extracting *LP* and *LN* from these *AMBI* clusters.

Acknowledgement: We thank Eamonn and Wei gracefully share the datasets for our experimental evaluation.

References

Borja, C., L. Pedro, et al. (2007). "Learning Bayesian classifiers from positive and unlabeled examples." Pattern Recognition Letters. 28(16): 2375-2384.

Cai, D., X. He, et al. (2005). "Document clustering using locality preserving indexing." IEEE Transactions on Knowledge and Data Engineering 17(12): 1624-1637.

Chapelle, O., B. Scholkopf, et al. (2006). "Semi-Supervised Learning." MIT Press.

Denis, F. (1998). "PAC Learning from Positive Statistical Queries." ALT.

Hyunjin, Y. and S. Cyrus (2006). "Feature Subset Selection on Multivariate Time Series with Extremely Large Spatial Features." IEEE ICDMW'06.

Kanungo, T., D. M. Mount, et al. (2002). "An efficient k-means clustering algorithm: analysis and implementation." IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7): 881-892.

Keogh, E. (2008). "The UCR Time Series Classification/ Clustering Homepage" http://www.cs.ucr.edu/~eamonn/time_series_data/.

Keogh, E. and S. Kasetty (2003). "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration." Data Mining and Knowledge Discovery 7(4): 349-371.

Li, M. and Z.-H. Zhou (2005). "SETRED: Self-training with Editing." Advances in Knowledge Discovery and Data Mining, Springer Berlin / Heidelberg. 3518: 611-621.

Li, X. and B. Liu (2003). "Learning to classify texts using positive and unlabeled data." IJCAI.

Li, X., B. Liu, et al. (2007). "Learning to Identify Unexpected Instances in the Test Set." IJCAI.

Li, X., P. Yu, et al. (2009). "Positive Unlabeled Learning for Data Stream Classification." SDM, SIAM.

Liu, B. (2007). Web Data Mining, Springer.

Liu, B., W. S. Lee, et al. (2002). "Partially Supervised Classification of Text Documents." ICML.

Olszewski, R. T. (2001). "Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data." PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Ratanamahatana, C. and D. Wanichsan (2008). "Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification." Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Springer Berlin / Heidelberg. 149: 1-14.

Rath, T. M. and R. Manmatha (2003). "Word image matching using dynamic time warping." IEEE CVPR.

Wei, L. (2007). "Self Training dataset." http://alumni.cs.ucr.edu/~wli/selfTraining/.

Wei, L. and E. Keogh (2006). "Semi-supervised time series classification." ACM SIGKDD.

Xi, X., E. Keogh, et al. (2006). "Fast time series classification using numerosity reduction." Proceedings of the 23rd international conference on Machine learning. ACM.

Yoon, H., K. Yang, et al. (2005). "Feature subset selection and feature ranking for multivariate time series." IEEE Transactions on Knowledge and Data Engineering 17(9): 1186-1198.

Zhu, X. (2008). "Semi-supervised learning literature survey." Technical report, no. 1530, Computer Sciences, University of Wisconsin-Madison.