Applying prototype selection and abstraction algorithms for efficient time-series classification

Stefanos Ougiaroglou, Leonidas Karamitopoulos, Christos Tatoglou, Georgios Evangelidis, and Dimitris A. Dervos

Abstract A widely used time series classification method is the single nearest neighbour. It has been adopted in many time series classification systems because of its simplicity and effectiveness. However, the efficiency of the classification process depends on the size of the training set as well as on data dimensionality. Although many speed-up methods for fast time series classification have been proposed and are available in the literature, state-of-the-art, non-parametric prototype selection and abstraction data reduction techniques have not been exploited on time series data. In this work, we present an experimental study where known prototype selection and abstraction algorithms are evaluated both on original data and a dimensionally reduced representation form of the same data from seven popular time series datasets. The experimental results demonstrate that prototype selection and abstraction algorithms, even when applied on dimensionally reduced data, can effectively reduce the computational cost of the classification process and the storage requirements for the training data, and, in some cases, improve classification accuracy.

1 Introduction

Classification methods based on similarity search have been proven to be effective for time series data analysis. More specifically, the one-Nearest Neighbour (1-NN) classifier is a widely-used method. It works by assigning to an unclassified time

Stefanos Ougiaroglou, Georgios Evangelidis
Department of Applied Informatics, School of Information Sciences, University of Macedonia,
156 Egnatia St, GR-54006, Thessaloniki, Greece
e-mail: stoug@uom.gr, gevan@uom.gr

Leonidas Karamitopoulos, Christos Tatoglou, Dimitris A. Dervos
Information Technology Department, Alexander TEI of Thessaloniki,
GR-57400 Sindos, Greece
e-mail: lkaramit@otenet.gr, xtatty@gmail.com, dad@it.teithe.gr

series the class label of its most similar training time series. The main drawback of similarity-based classifiers is that all similarities between an unclassified time series item and the training time series items must be computed. For large and high dimensional time series training sets, the high computational cost involved renders the application of such classifiers prohibitive. Time series classification performance can be improved through indexing, representation and/or data reduction.

Indexing accelerates classification, but works well only in low dimensionality spaces. Thus, one must first use a dimensionality reduction technique to acquire a representation of the original data in lower dimensions. A representation may be considered as a transformation technique that maps a time series from the original space to a feature space, retaining the most important features. There have been several time series representations proposed in the literature, mainly for the purpose of reducing the intrinsically high dimensionality of time series [12].

The main goal of data reduction is to reduce the computational cost of the k-NN classifier and the storage requirements of the training set. Data Reduction Techniques (DRTs)[1] [31, 14, 20, 30, 33, 18, 16, 7, 21] build a small representative set of the initial training data. This set is called the condensing set and has the benefits of low computational cost and storage requirements while keeping the accuracy at high levels. DRT algorithms may be grouped into two categories: (i) Prototype Selection (PS) [14], and, (ii) Prototype Abstraction (PA) (or generation) [31]. Both categories share the same motivation. However, they differ on the way the condensing set is constructed. PS algorithms select some training items and use them as representatives, whereas, PA algorithms generate new item representatives by summarizing on similar training items.

Data reduction has recently been exploited for fast time series classification. More specifically, [8] and [34] propose PS algorithms for speeding-up 1-NN time series classification. The disadvantage of these methods is that they are parametric. The user must define the size of the condensing set by trial-and-error.

The present work has been motivated by the following two observations: (a) to the best of our knowledge, state-of-the-art non-parametric PS and PA algorithms have not been evaluated neither on original time series nor on their reduced dimensionality representations, and, (b) PA algorithms that we have proposed (RHC [24, 23], AIB2 [25, 22]) have not been evaluated on time series data. The contribution of this paper is the experimental evaluation of two PS algorithms, namely, CNN-rule [17] and IB2 [3, 2], and three PA algorithms, namely, RSP3 [28], RHC [24] and AIB2 [25, 22]. The algorithms are evaluated both against original time series datasets and their reduced dimensionality representations.

Our study adopts the Piecewise Aggregate Approximation (PAA) [19, 35] time series representation method. The goal is to investigate the degree to which classification accuracy gets affected when applying data reduction on dimensionally reduced time series. PAA is an effective and very simple dimensionality reduction technique that segments a time series into h consecutive sections of equal-width and

[1] One can claim that dimensionality reduction is also data reduction. However, we consider DRTs only from the item reduction point of view.

calculates the corresponding mean for each section. The series of these means is the new representation of the original data.

The rest of the paper is organized as follows. Section 2 discusses the details of the five aforementioned DRTs. Section 3 presents the experimental setup and the results obtained, and Section 4 concludes the paper.

2 Data Reduction Techniques

In this section, we present the five DRTs used in our experimentation. They are based on a simple idea: data items that do not represent decision boundaries between classes are useless for the classification process. Therefore, they can be discarded. The idea is that the k-NN classifier achieves similar accuracy using either the training set or the condensing set. However, condensing set scanning is more efficient than training set scanning. Consequently, DRTs try to select or generate a sufficient number of items that lie in data areas close to decision boundaries. The DRTs we deal with in this section are non-parametric. They automatically determine the size of the condensing set based on the level of noise and the number of classes in the data (the more the classes, the more boundaries exist and, thus, the more items get selected or generated). Therefore, expensive trial-end-error procedures for parameter tuning are avoided.

2.1 Prototype Selection algorithms

2.1.1 Hart's Condensing Nearest Neighbour rule (CNN-rule)

CNN-rule [17] is the earliest and the best known PS algorithm. It uses two sets, CS and TS. Initially, a training item is placed in CS, while all the other training items are placed in TS. Then, CNN-rule tries to classify the content of TS by using the 1-Nearest Neighbour (1-NN) classifier on the content of CS. When an item is misclassified, it is considered to lie in a data area close to decision boundaries. Thus, it is transferred from TS to CS. The algorithm terminates when there are no transfers from TS to CS during a complete pass of TS. The final instance of set CS constitutes the condensing set.

Algorithm 1 presents the pseudo-code of CNN rule: it starts with a training set TS and returns a condensing set CS. Initially, CS has only a training item (lines 1 and 2). Then, for each training item $x \in TS$ (line 5), the algorithm retrieves and examines the class label of its nearest neighbour (line 6). If the class label of x differs from that of its nearest neighbour (line 7), x is moved to CS (lines 8–9). The repeat-until loop terminates when there are no more TS items to migrate to CS (lines 4,10,13).

The multiple passes on data ensure that the remaining (discarded) items in TS can be correctly classified by applying the 1-NN classifier on the condensing set. The

Algorithm 1 CNN-rule

Input: TS
Output: CS

```
1: CS ← ∅
2: pick an item of TS and move it to CS
3: repeat
4:     stop ← TRUE
5:     for each x ∈ TS do
6:         NN ← Nearest Neighbour of x in CS
7:         if NN_class ≠ x_class then
8:             CS ← CS ∪ {x}
9:             TS ← TS − {x}
10:            stop ← FALSE
11:        end if
12:    end for
13: until stop == TRUE
14: discard TS
15: return CS
```

algorithm is based on the following simple idea: items that are correctly classified by 1-NN, are considered to lie in a central-class data area and thus, they are ignored.On the other hand, items that are misclassified, are considered to lie in a close-class-border data area, and thus, they are placed in the condensing set. The weak point of the CNN-rule is that the resulting condensing set depends on the ordering by which training set items are considered. This means that different condensing sets may be constructed by considering the same training set data in a different order.

There are many other condensing algorithms that either extend the CNN-rule, or they are based on the same idea. Some of the these algorithms are the Reduced Nearest Neighbour (RNN) rule [15], the Selective Nearest Neighbour (SNN) rule [27], the Modified CNN rule [11], the Generalized CNN rule [10], the Fast CNN algorithms [4, 5], Tomek's CNN rule [29], the Patterns with Ordered Projection (POP) algorithm [26, 1], the recently proposed Template Reduction for k-NN (TRkNN) [13] and the IB2 algorithm [3, 2].

2.1.2 IB2

IB2 belongs to the well-known family of Instance-Based Learning (IBL) algorithms [3, 2] and is based on the CNN-rule. In effect, IB2 constitutes a simple one pass variation of the CNN-rule. Algorithm 2 presents IB2 in pseudo-code. Each training item $x \in TS$ is classified using 1-NN classifier on the current CS (line 4). If x is classified correctly, it is discarded (line 8). Otherwise, x is transferred to CS (line 6).

Contrary to the CNN-rule, IB2 does not ensure that all discarded items can be correctly classified by the final version of the condensing set. However, since it is a one-pass algorithm, it is very fast, i.e., it involves low preprocessing computational

Algorithm 2 IB2

Input: TS
Output: CS

1: $CS \leftarrow \varnothing$
2: an item is chosen at random to migrate from TS to CS
3: **for** each $x \in TS$ **do**
4: $NN \leftarrow$ Nearest Neighbour of x in CS
5: **if** $NN_{class} \neq x_{class}$ **then**
6: $CS \leftarrow CS \cup \{x\}$
7: **end if**
8: $TS \leftarrow TS - x$
9: **end for**
10: **return** CS

cost. In addition, IB2 builds its condensing set incrementally. New training items can be taken into consideration after the creation of the condensing set. Therefore, IB2 is appropriate for dynamic/streaming environments whereby new training items arrive in an one-by-one fashion. Certainly, IB2 can not deal with data streams with concept drift [32]. IBL-DS [6] adopts the idea of the family of IBL algorithms and can deal with such data. It is worth mentioning that, contrary to the CNN-rule and to many other DRTs, IB2 does not require that all training data reside in main memory. Therefore, it can be applied in devices whose memory is insufficient for storing all the training data. Of course, like the CNN-rule, IB2 is a data ordering dependent algorithm.

2.2 Prototype Abstraction algorithms

2.2.1 Abstraction IB2 (AIB2)

The AIB2 algorithm constitutes a PA variation of IB2. Therefore, it inherits all the aforementioned properties of IB2. The idea behind AIB2 is quite simple: prototypes should be at the center of the data area they represent. Therefore, the correctly classified items are not ignored. In effect, they contribute to the final condensing set by repositioning their nearest prototype. This is achieved by adopting the concept of prototype weight. Each prototype is characterized by a weight value. It denotes the number of items it represents.

Algorithm 3 presents the pseudo code of the algorithm. Initially, the condensing set (CS) has only one item whose weight is initialized to one (lines 1–3). For each training item x, AIB2 retrieves from the current CS its nearest prototype nn (line 5). If x is misclassified, it is placed in CS and its weight is initialized to one (lines 6–8). Otherwise, the attributes of nn are updated by taking into account its current weight and the attributes of x. In effect, nn "moves" towards x (lines 10–12). Finally, the weight of NN is increased by one (line 13) and x is removed (line 15).

Algorithm 3 AIB2

Input: TS
Output: CS

$CS \leftarrow \varnothing$
move a random item y from TS to CS
$y_{weight} \leftarrow 1$
for each $x \in TS$ **do**
  $nn \leftarrow$ Nearest Neighbour of x in CS
  if $nn_{class} \neq x_{class}$ **then**
    $x_{weight} \leftarrow 1$
    $CS \leftarrow CS \cup \{x\}$
  else
    for each attribute $attr(i)$ **do**
      $nn_{attr(i)} \leftarrow \frac{nn_{attr(i)} \times nn_{weight} + x_{attr(i)}}{nn_{weight}+1}$
    end for
    $nn_{weight} \leftarrow nn_{weight} + 1$
  end if
  $TS \leftarrow TS - \{x\}$
end for
return CS

AIB2 aims at improving the efficiency of IB2 by building a condensing set with better prototypes than IB2. Each prototype lies close to the center of the data area it represents. Therefore AIB2 is able to achieve higher classification accuracy. Moreover, the repositioned prototypes reduce the items placed in the final condensing set. Hence, AIB2 can achieve higher reduction rates and even lower preprocessing cost than IB2.

2.2.2 RSP3

The RSP3 algorithm belongs to the popular family of Reduction by Space Partitioning (RSP) algorithms [28]. This family includes three PA algorithms. All of them are based on the idea of the early PA algorithm of Chen and Jozwik [9]. Chen and Jozwik's Algorithm (CJA) works as follows: First, the most distant items A and B of the training set that define its diameter are retrieved. Then, the training set is divided into two subsets, S_A and S_B. S_A includes training items that lie closer to A, whereas, S_B includes training items that lie closer to B. Then, CJA proceeds by selecting to divide subsets that include items of more than one classes (non-homogeneous subsets). The subset with the largest diameter is divided first. If all subsets are homogeneous, CJA divides the largest homogeneous subset. This procedure continues until the number of subsets becomes equal to a user specified value. In the end, for each subset S, CJA averages the items in S and creates a mean item that is assigned the label of the majority class in S. The created mean items constitute the final condensing set.

Certainly, the mean item m of each subset S, is computed by averaging the t attribute values of items x_i, $i = 1,2\ldots|S|$ that belong to S. Therefore, the t average attributes $m.d_j$ of m are estimated as follows:

$$m.d_j = \frac{1}{|S|}\sum_{x_i \in S} x_i.d_j, j = 1,2,\ldots,n$$

Algorithm 4 lists in pseudo-code a possible implementation of RSP3. It accepts a training set (TS) and the number of prototypes n that will be generated. The algorithm uses a data structure to store the created subsets. Initially, the entire TS is stored in S (line 2). Then, the non-homogeneous subset C with the largest diameter is divided into two subsets (lines 4,8). If all subsets are homogeneous, CJA divides the homogeneous subset C with the largerst diameter (lines 5–7). Both subsets are added to S, while C is removed (lines 9–11). The procedure for constructing subsets continues until n subsets have been created (line 3). The last step is the mean computation (or prototype generation) for each subset and its inclusion in the condensing set (CS) (lines 13–18).

Algorithm 4 CJA

Input: TS, n
Output: CS

```
1:  S ← ∅
2:  add(S, TS)
3:  for i = 2 to n do
4:      C ← select the non-homogeneous subset ∈ S with the largest diameter
5:      if C == ∅ {All subsets are homogeneous} then
6:          C ← select the homogeneous subset ∈ S with the largest diameter
7:      end if
8:      (S_x, S_y) ← divide C into two subsets
9:      add(S, S_x)
10:     add(S, S_y)
11:     remove(S, C)
12: end for
13: CS ← ∅
14: for each subset T ∈ S do
15:     r ← compute the mean item by averaging the items in T
16:     r.label ← find the most common class label in T
17:     CS ← CS ∪ {r}
18: end for
19: return CS
```

CJA selects the next subset that will be divided by examining its diameter. The idea is that a subset with a large diameter probably includes more training items. Therefore, if this subset is divided first, a higher reduction rate will be achieved. A desirable property is that CJA builds the same condensing subset regardless of the ordering of the data in the training set. However, it has two weak points. The first is that the algorithm is parametric. The user has to specify the number of prototypes.

This usually involves tuning via a costly trial-end-error procedure. In certain domains, this property may be desirable, since it allows one to control the size of the condensing subset. However, it prohibits the automatic determination of the size of the condensing subset in accordance with the nature of the available data. This constitutes a drawback for the algorithm, in general. The second weakness is that the items that do not belong to the most common class of the subset are not represented in the condensing set. Since the mean item of each subset is labeled by the most common class, items that belong to other classes are practically ignored.

RSP1 deals with the second drawback. More specifically, RSP1 computes as many mean items as the number of different classes in each subset. Therefore, it averages the items that belong to each class in the subset. Of course, RSP1 builds larger CSs than CJA. However, it attempts to improve accuracy since it takes into account all training items.

RSP1 and RSP2 differ on how they select the next subset to be divided. Similar to CJA, RSP1 uses the subset diameter as the splitting criterion. In contrast, RSP2 uses as its splitting criterion the highest overlapping degree. This criterion assumes that items that belong to a specific class lie as close to each other as possible while items that belong to different classes lie as far as possible. According to [28], it is better to divide the subset with the highest overlapping degree. The overlapping degree of a subset is the ratio of the average distance between items belonging to different classes and the average distance between items that belong to the same class.

RSP3 adopts the concept of homogeneity. A subset is homogeneous when it includes items of only a specific class. The algorithm continues dividing the created subsets until all of them become homogeneous. RSP3 can use either the largest diameter or the highest overlapping degree as spiting criterion. Actually, since all non-homogeneous subsets are divided, the choice of splitting criterion becomes an issue of secondary importance.

Algorithm 5 lists the pseudo-code of RSP3. It utilizes a data structure S to hold the unprocessed subsets. Initially, the whole training set (TS) is an unprocessed subset and is put in S (line 2). At each repeat-until iteration, RSP3 selects the subset C with the highest splitting criterion value (line 5) and checks if C is homogeneous or not. If it is homogeneous, the mean item is computed by averaging the items in C and is placed in the condensing set (CS) as a prototype (lines 6–9). Otherwise, C is divided into two subsets D_1 and D_2 (line 11) in the CJA fashion. These new subsets are added to S and C is removed from S (lines 12–15). The repeat-until loop continues until S becomes empty (line 22), i.e., all subsets are homogeneous.

Certainly, RSP3 generates more prototypes for the "close border" data areas and fewer for the "central" data areas. RSP3 is the only non-parametric algorithm of the RSP family (CJA included). It is worth mentioning that like CJA, RSP1 and RSP2, the condensing set built by RSP3 does not depend on the data order in the training set. The procedure for finding the most distant items in each subset is quite expensive. Thus, the main drawback of RSP3 is that it usually involves high preprocessing computation cost. In cases of large datasets, this drawback may evan render its execution prohibitive.

Algorithm 5 RSP3

Input: TS
Output: CS

```
 1: S ← ∅
 2: add(S, TS)
 3: CS ← ∅
 4: repeat
 5:     C ← select the subset ∈ S with the highest splitting criterion value
 6:     if C is homogeneous then
 7:         r ← calculate the mean item by averaging the items in C
 8:         r.label ← class of items in C
 9:         CS ← CS ∪ {r}
10:     else
11:         (D1, D2) ← divide C into two subsets
12:         add(S, D1)
13:         add(S, D2)
14:         remove(S, C)
15:     end if
16: until IsEmpty(S)
17: return CS
```

2.2.3 Reduction through Homogeneous Clusters (RHC)

RHC [24, 23] is also based on the concept of homogeneity. Initially, the whole training set is considered as a non-homogeneous cluster C. RHC begins by computing a mean item for each class (class-mean) in C. Then, it applies k-means clustering on C using the class means as initial means. The clustering procedure builds as many clusters as the number of classes in C. The aforementioned clustering procedure is applied recursively on each non-homogeneous cluster. In the end, the means of the homogeneous clusters are stored in the condensing set as prototypes.

Algorithm 6 lists the pseudo-code of RHC. It utilizes a queue data structure, Q, to store clusters. Initially, the training data (TS) is considered as an unprocessed cluster. Therefore, it is placed in Q (line 3). At each one iteration, the algorithm examines the head C of Q (line 7). Then it checks whether C is a homogeneous cluster or not. If it is homogeneous (line 8), the mean of C is placed in the condensing set (CS) (line 9) and its items are removed. If C is non-homogeneous, the algorithm computes the class means (M): one class mean for each of the classes in C (lines 13–16). Then, RHC calls k-means clustering, with parameters C and M. k-means produces a new set of clusters ($clusters$) (line 17) that are enqueued into Q (lines 18–20). The algorithm stops iterating when Q becomes empty (line 22), i.e., all clusters are homogeneous.

Like RSP3, RHC builds many prototypes for close-class-border data areas and fewer for the non-close-class-border data areas. By using the class means as initial means for the k-means clustering, the algorithm attempts to quickly find homogeneous clusters and achieve high reduction rates (the larger the clusters built, the higher the reduction rates achieved). Moreover, since RHC is based on k-means clustering, it is very fast and can easily be integrated into much of the existing soft-

Algorithm 6 RHC

Input: TS
Output: CS

```
1: Q ← ∅
2: Enqueue(Q, TS)
3: CS ← ∅
4: repeat
5:     C ← Dequeue(Q)
6:     if C is homogeneous then
7:         r ← mean of C
8:         CS ← CS ∪ {r}
9:     else
10:        M ← ∅
11:        for each class L in C do
12:            m_L ← mean of L
13:            M ← M ∪ {m_L}
14:        end for
15:        clusters ← K-MEANS(C, M)
16:        for each cluster C ∈ clusters do
17:            Enqueue(Q, C)
18:        end for
19:    end if
20: until IsEmpty(Queue)
21: return CS
```

ware. In addition, RHC is independent on the ordering of the data in the training set. The results of the experimental study in [24, 23] indicate that RHC achieves higher reduction rates (smaller CSs) and is faster than RSP3 and CNN-rule, while accuracy remains high.

3 Experimental Study

3.1 Experimental setup

The five DRT algorithms presented were evaluated on seven known time series datasets distributed by the UCR time-series classification/clustering website[2]. Table 1 summarizes on the datasets used. All datasets are available in a training/testing form. We merged the training and testing parts and then we randomized the resulting datasets. No other data transformation was performed. All algorithms were coded in C and as a similarity measure we used the Euclidean distance.

We report on the experiment we conducted with a certain value for the parameter of the PAA representation. We applied the PAA representation on time series by setting the number of dimensions equal to twelve (h=12). Most of the research work

[2] http://www.cs.ucr.edu/∼eamonn/time_series_data/

Table 1: Time-series datasets description

Time-series dataset	Size (time-series)	Length (Attr.)	Classes
Synthetic Control (SC)	600	60	6
Face All (FA)	2250	131	14
Two-Patterns (TP)	5000	128	4
Yoga (YG)	3300	426	2
Wafer (WF)	7164	152	2
Sweadish Leaf (SL)	1125	128	15
CBF	930	128	3

provides experimental results with values of h ranging from 2 to 20. We found that lower values of h have a negative effect on classification accuracy, whereas, higher values produce time series that cannot be efficiently indexed by multi-dimensional indexing methods. Hence, we decided to use h=12.

All experiments were run twice, once on the original time series and once on their 12-dimensional representations. We wanted to test how the combination of data reduction and dimensionality reduction affects the performance of 1-NN classification.

We evaluated the five DRTs by estimating four measurements, namely, accuracy (ACC), classification cost (CC), reduction rate (RR), and, preprocessing cost (PC). Cost measurements were estimated by counting the distance computations multiplied by the number of time series attributes (time series length). Of course, RR and CC measurements relate to each other: the lower the RR, the higher is the CC. However, CC measurements can express the cost introduced by the dimensionality of data. We report on the average values of these measurements obtained via five-cross-fold validation.

3.2 Comparisons

Tables 2 and 3 present the experimental measurements. Table 2 presents the results obtained on the original datasets while table 3 presents the results obtained on the 12-dimensional representations of the datasets we got after applying PAA on them. Both tables include the measurements obtained by applying the 1-NN classifier on the non-reduced data (conventional 1-NN). Each table cell includes the four measurements obtained by first applying a DRT on the original or 12-dimensional time series datasets (preprocessing step) and then by using 1-NN on the resulting condensing set (classification step). The cost measurements are in million (M) distance computations. The PC measurements do not include the small cost overhead introduced by PAA execution.

It is noted that 1-NN classification on the 12-dimensional datasets is very fast. In most cases, the preprocessing and classification cost are extremely low, while classification accuracy remains at high, acceptable levels. Therefore, a first conclusion is

Table 2: Experimental results on original datasets

Dataset		Original dimensionality					
		Conv. 1-NN	**CNN**	**IB2**	**RSP3**	**RHC**	**AIB2**
SC	Acc (%):	91.67	90.17	89.00	98.33	98.67	**99.83**
	CC (M):	3.46	0.67	0.53	1.38	**0.09**	0.34
	RR (%):	-	80.50	84.67	60.08	**97.29**	90.13
	PC (M):	-	7.77	1.31	16.22	2.39	**1.14**
FA	Acc (%):	95.07	91.60	91.02	**95.46**	93.02	92.94
	CC (M):	106.11	19.87	18.38	51.65	**12.93**	16.08
	RR (%):	-	81.28	82.68	51.32	**87.81**	84.84
	PC (M):	-	216.36	48.96	533.70	140.41	**43.27**
TP	Acc (%):	**98.50**	94.68	93.60	98.10	93.72	97.06
	CC (M):	512.00	85.66	76.83	243.51	**55.50**	61.88
	RR (%):	-	83.27	85.00	52.44	**89.16**	87.92
	PC (M):	-	1169.75	205.95	2085.42	**150.49**	177.88
YG	Acc (%):	**93.76**	91.58	89.55	92.85	90.94	90.49
	CC (M):	742.26	138.56	108.92	229.82	**93.85**	100.26
	RR (%):	-	81.33	85.33	69.04	**87.36**	86.49
	PC (M):	-	1854.74	254.41	4072.30	**162.61**	240.73
WF	Acc (%):	**99.87**	99.69	99.62	99.82	99.55	99.65
	CC (M):	1248.30	13.59	11.72	26.88	**9.37**	9.71
	RR (%):	-	98.91	99.06	97.85	**99.25**	99.22
	PC (M):	-	165.88	31.42	7196.75	63.69	**25.78**
SL	Acc (%):	52.36	49.87	48.18	52.00	**52.80**	51.56
	CC (M):	25.92	15.94	14.80	19.00	**12.80**	14.65
	RR (%):	-	38.51	42.89	26.69	**50.60**	43.49
	PC (M):	-	112.17	31.39	1537.07	57.01	**31.02**
CBF	Acc (%):	98.39	98.17	97.63	**99.78**	98.60	99.68
	CC (M):	17.71	1.29	1.15	1.97	**0.40**	0.59
	RR (%):	-	92.74	93.49	88.87	**97.74**	96.67
	PC (M):	-	15.06	3.50	78.48	7.26	**2.01**
Avg	Acc (%):	89.94	87.97	86.94	**90.91**	89.62	90.17
	CC (M):	379.40	39.37	33.19	82.03	**26.42**	29.07
	RR (%):	-	79.51	81.87	63.76	**87.03**	84.11
	PC (M):	-	505.96	82.42	2217.13	83.37	7**4.55**

that one can obtain efficient time series classifiers by combining prototype selection or abstraction algorithms with time-series dimensionality reduction representations.

It is worth mentioning that the three PA algorithms, RSP3, RHC and AIB2, achieved higher classification accuracy than the conventional 1-NN. In the case of SC dataset, accuracy improvement was very high. Almost in all cases, RSP3 achieved the highest accuracy. However, it is the slowest method in terms of both preprocessing and classification (RSP3 had the lowest reduction rates). The high PC measurements are attributed to the costly procedure for finding the most distant items in each created subset (see Subsection 2.2 or [28] for details).

RHC, AIB2 and IB2 had much lower preprocessing cost than the other two methods. This happened because IB2 and AIB2 are one-pass algorithms and RHC is based on a version of *k*-means that is sped-up by the class mean initializations (see

Table 3: Experimental results on datasets with 12 dimensions

Dataset		12 dimensions					
		Conv. 1-NN	**CNN**	**IB2**	**RSP3**	**RHC**	**AIB2**
SC	Acc (%):	98.50	97.00	95.83	**98.83**	98.17	98.50
	CC (M):	0.69	0.06	0.05	0.12	**0.03**	**0.03**
	RR (%):	-	90.75	93.13	82.96	**95.75**	95.13
	PC (M):	-	0.89	0.13	3.45	0.52	**0.10**
FA	Acc (%):	**87.91**	83.78	82.31	87.07	84.49	84.36
	CC (M):	9.72	2.89	2.53	4.80	**2.08**	2.22
	RR (%):	-	70.23	74.01	50.58	**78.59**	77.21
	PC (M):	-	30.36	5.95	50.91	13.16	**5.30**
TP	Acc (%):	**97.56**	93.52	91.38	96.66	94.34	94.48
	CC (M):	48.00	8.22	6.86	20.42	6.69	**5.39**
	RR (%):	-	82.89	85.72	57.45	86.06	**88.77**
	PC (M):	-	103.86	17.34	196.00	17.63	**14.56**
YG	Acc (%):	**92.36**	90.39	88.03	91.03	90.03	89.67
	CC (M):	20.91	4.41	3.50	6.71	3.13	**3.12**
	RR (%):	-	78.91	83.26	67.90	85.02	**85.06**
	PC (M):	-	52.23	8.04	110.56	**4.26**	7.30
WF	Acc (%):	**99.79**	99.62	99.51	99.40	99.25	99.50
	CC (M):	98.55	1.21	1.01	1.86	1.01	**0.99**
	RR (%):	-	98.77	98.97	98.11	98.97	**99.00**
	PC (M):	-	15.63	2.57	495.63	4.64	**2.44**
SL	Acc (%):	**52.62**	49.07	48.62	51.20	51.20	49.78
	CC (M):	2.43	1.54	1.37	1.78	**1.32**	1.35
	RR (%):	-	36.76	43.67	26.69	**45.69**	44.40
	PC (M):	-	11.33	2.86	56.00	4.99	**2.84**
CBF	Acc (%):	**100.00**	99.57	99.35	99.68	99.57	99.46
	CC (M):	1.66	0.06	0.06	0.12	0.04	**0.04**
	RR (%):	-	96.34	96.56	92.63	97.47	**97.55**
	PC (M):	-	0.66	0.19	7.32	0.70	**0.14**
Avg	Acc (%):	**89.82**	87.57	86.43	89.12	88.15	87.96
	CC (M):	25.99	2.63	2.20	5.12	2.04	**1.88**
	RR (%):	-	79.24	82.19	68.05	**83.94**	83.87
	PC (M):	-	30.71	5.30	131.44	6.56	**4.67**

Subsection 2.2 or [24] for details). In addition, RHC builds the smallest CSs. In all cases, RHC achieved higher reduction rates than the other DRTs. Thus, the corresponding classifiers had the lowest classification costs.

The classification accuracy achieved by RHC was usually higher than IB2 and CNN-rule and as high as AIB2. In some cases, RHC and AIB2 were more accurate than RSP3. Considering the above, one may conclude that, since RHC ans AIB2 deal with all comparison criteria, they are efficient speed-up methods for time-series data. Finally, the experimental results illustrate that AIB2 is an efficient variation of IB2. In all cases, AIB2 achieves higher performance than IB2.

No DRT can be said to comprise the best speed-up choice. If classification accuracy is the most critical criterion, RSP3 may be preferable. On the other hand, if fast

classification and/or fast construction of the condensing set are more critical than accuracy, RHC or AIB2 may be a better choice.

4 Conclusions

Efficient time series classification is an open research issue that attracts the interest of the data mining community. This paper proposes the use of non-parametric state-of-the-art prototype selection and abstraction algorithms for efficient and effective time series classification.

The experimental study conducted demonstrates that by combining prototype selection or abstraction algorithms with dimensionality reduction, one can obtain accurate and very fast time series classifiers. In addition, the study reveals that prototype abstraction algorithms are preferable to prototype selection algorithms when applied on time series data. The prototype abstraction algorithms examined in the study can achieve even higher accuracy than the conventional 1-NN classifier.

References

1. Aguilar, J.S., Riquelme, J.C., Toro, M.: Data set editing by ordered projection. Intell. Data Anal. **5**(5), 405–417 (2001). URL http://dl.acm.org/citation.cfm?id=1294007.1294010
2. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. **36**(2), 267–287 (1992). DOI 10.1016/0020-7373(92)90018-G. URL http://dx.doi.org/10.1016/0020-7373(92)90018-G
3. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6**(1), 37–66 (1991). DOI 10.1023/A:1022689900470. URL http://dx.doi.org/10.1023/A:1022689900470
4. Angiulli, F.: Fast condensed nearest neighbor rule. In: Proceedings of the 22nd international conference on Machine learning, ICML '05, pp. 25–32. ACM, New York, NY, USA (2005)
5. Angiulli, F.: Fast nearest neighbor condensation for large data sets classification. IEEE Trans. on Knowl. and Data Eng. **19**(11), 1450–1464 (2007). DOI 10.1109/TKDE.2007.190645. URL http://dx.doi.org/10.1109/TKDE.2007.190645
6. Beringer, J., Hüllermeier, E.: Efficient instance-based learning on data streams. Intell. Data Anal. **11**(6), 627–650 (2007). URL http://dl.acm.org/citation.cfm?id=1368018.1368022
7. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Min. Knowl. Discov. **6**(2), 153–172 (2002). DOI 10.1023/A:1014043630878. URL http://dx.doi.org/10.1023/A:1014043630878
8. Buza, K., Nanopoulos, A., Schmidt-Thieme, L.: Insight: efficient and effective instance selection for time-series classification. In: Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part II, PAKDD'11, pp. 149–160. Springer-Verlag, Berlin, Heidelberg (2011)
9. Chen, C.H., Jóźwik, A.: A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recogn. Lett. **17**(8), 819–823 (1996). DOI 10.1016/0167-8655(96)00041-4. URL http://dx.doi.org/10.1016/0167-8655(96)00041-4

10. Chou, C.H., Kuo, B.H., Chang, F.: The generalized condensed nearest neighbor rule as a data reduction method. In: Proceedings of the 18th International Conference on Pattern Recognition - Volume 02, ICPR '06, pp. 556–559. IEEE Computer Society, Washington, DC, USA (2006). DOI 10.1109/ICPR.2006.1119. URL http://dx.doi.org/10.1109/ICPR.2006.1119
11. Devi, V.S., Murty, M.N.: An incremental prototype set building technique. Pattern Recognition **35**(2), 505–513 (2002)
12. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endow. **1**(2), 1542–1552 (2008). URL http://dl.acm.org/citation.cfm?id=1454159.1454226
13. Fayed, H.A., Atiya, A.F.: A novel template reduction approach for the k-nearest neighbor method. Trans. Neur. Netw. **20**(5), 890–896 (2009). DOI 10.1109/TNN.2009.2018547. URL http://dx.doi.org/10.1109/TNN.2009.2018547
14. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012). DOI 10.1109/TPAMI.2011.142. URL http://dx.doi.org/10.1109/TPAMI.2011.142
15. Gates, G.W.: The reduced nearest neighbor rule. ieee transactions on information theory. IEEE Transactions on Information Theory **18**(3), 431–433 (1972)
16. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms ii. results and comments. In: Artificial Intelligence and Soft Computing - ICAISC 2004, *LNCS*, vol. 3070, pp. 580–585. Springer Berlin / Heidelberg (2004)
17. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory **14**(3), 515–516 (1968)
18. Jankowski, N., Grochowski, M.: Comparison of instances seletion algorithms i. algorithms survey. In: Artificial Intelligence and Soft Computing - ICAISC 2004, *LNCS*, vol. 3070, pp. 598–603. Springer Berlin / Heidelberg (2004)
19. Keogh, E.J., Pazzani, M.J.: A simple dimensionality reduction technique for fast similarity search in large time series databases. In: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, PADKK '00, pp. 122–133. Springer-Verlag, London, UK, UK (2000)
20. Lozano, M.: Data Reduction Techniques in Classification processes (Phd Thesis). Universitat Jaume I (2007)
21. Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Kittler, J.: A review of instance selection methods. Artif. Intell. Rev. **34**(2), 133–143 (2010). DOI 10.1007/s10462-010-9165-y. URL http://dx.doi.org/10.1007/s10462-010-9165-y
22. Ougiaroglou, S., Evangelidis, G.: Efficient data abstraction using weighted ib2 prototypes. Submitted, under review
23. Ougiaroglou, S., Evangelidis, G.: Rhc: Non-parametric cluster-based data reduction for efficient k-nn classification. Submitted, under review
24. Ougiaroglou, S., Evangelidis, G.: Efficient dataset size reduction by finding homogeneous clusters. In: Proceedings of the Fifth Balkan Conference in Informatics, BCI '12, pp. 168–173. ACM, New York, NY, USA (2012). DOI 10.1145/2371316.2371349. URL http://doi.acm.org/10.1145/2371316.2371349
25. Ougiaroglou, S., Evangelidis, G.: Aib2: An abstraction data reduction technique based on ib2. In: Proceedings of the 6th Balkan Conference in Informatics, BCI '13, pp. 13–16. ACM, New York, NY, USA (2013). DOI 10.1145/2490257.2490260. URL http://doi.acm.org/10.1145/2490257.2490260
26. Riquelme, J.C., Aguilar-Ruiz, J.S., Toro, M.: Finding representative patterns with ordered projections. Pattern Recognition **36**(4), 1009 – 1018 (2003). DOI http://dx.doi.org/10.1016/S0031-3203(02)00119-X. URL http://www.sciencedirect.com/science/article/pii/S003132030200119X
27. Ritter, G., Woodruff, H., Lowry, S., Isenhour, T.: An algorithm for a selective nearest neighbor decision rule. IEEE Trans. on Inf. Theory **21**(6), 665–669 (1975)

28. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition **37**(7), 1561–1564 (2004)
29. Tomek, I.: Two modifications of cnn. Systems, Man and Cybernetics, IEEE Transactions on **SMC-6**(11), 769–772 (1976). DOI 10.1109/TSMC.1976.4309452
30. Toussaint, G.: Proximity graphs for nearest neighbor decision rules: Recent progress. In: 34th Symposium on the INTERFACE, pp. 17–20 (2002)
31. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. Trans. Sys. Man Cyber Part C **42**(1), 86–100 (2012). DOI 10.1109/TSMCC.2010.2103939. URL http://dx.doi.org/10.1109/TSMCC.2010.2103939
32. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland (2004)
33. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-basedlearning algorithms. Mach. Learn. **38**(3), 257–286 (2000). DOI 10.1023/A:1007626913721. URL http://dx.doi.org/10.1023/A:1007626913721
34. 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, ICML '06, pp. 1033–1040. ACM, New York, NY, USA (2006). DOI 10.1145/1143844.1143974. URL http://doi.acm.org/10.1145/1143844.1143974
35. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. In: Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00, pp. 385–394. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000). URL http://dl.acm.org/citation.cfm?id=645926.671689