Title Page

Noname manuscript No.
(will be inserted by the editor)

RHC: Non-parametric Cluster-based Data Reduction for efficient k-NN Classification

Stefanos Ougiaroglou · Georgios Evangelidis

Received: date / Accepted: date

Abstract Although the k-NN classifier is a popular classification method, it suffers from the high computational cost and storage requirements it involves. This paper proposes two effective cluster-based data reduction algorithms for efficient k-NN classification. Both have low preprocessing cost and can achieve high data reduction rates while maintaining k-NN classification accuracy at high levels. The first proposed algorithm is called Reduction through Homogeneous Clusters (RHC) and is based on a fast pre-processing clustering procedure that creates homogeneous clusters. The centroids of these clusters constitute the reduced training set. The second proposed algorithm is a dynamic version of RHC that retains all its properties and, in addition, it can manage datasets that cannot fit in main memory and is appropriate for dynamic environments where new training data are gradually available. Experimental results, based on fourteen datasets, illustrate that both algorithms are faster and achieve higher reduction rates than four known methods, while maintaining high classification accuracy.

Keywords k-NN Classification · Clustering · Data reduction · Prototypes

S. Ougiaroglou is supported by a scholarship from the state scholarships foundation of Greece (I.K.Y.)

Stefanos Ougiaroglou (Corresponding author)
Dept. of Applied Informatics, School of Information Sciences, University of Tel.: +30-2310-891820
Fax: +30-2310-891290
E-mail: stoug@uom.gr

Georgios Evangelidis
Dept. of Applied Informatics, School of Information Sciences, University of Macedonia, Thessaloniki, E-mail: gevan@uom.gr

1 Introduction

Classification is an important data mining task that tries to assign new data to a set of predefined classes on the basis of the available training data, i.e., a set of already classified instances (or items) [24, 26]. A typical classification example is to assign an email to either class "spam" or class "non-spam".

The k-Nearest Neighbour (k-NN) classifier [12, 11] is an extensively used lazy classification algorithm. It is a simple and easy to implement classifier and can be exploited in many domains. The k-NN classifier does not build any classification model. The available training data is considered to be the classification model. Particularly, the algorithm classifies a new item x by searching in the training set for the k nearest items (neighbours) to x according to a distance metric (e.g., Eucledian distance). Then, x is assigned to the most common class determined via a majority vote of the retrieved k nearest neighbours. Ties are resolved either randomly or by the single nearest neighbour. The k-NN classifier is analytically tractable and for $k = 1$ and unlimited items the error rate is asymptotically never worse than twice the minimum possible, which is the Bayes rate [11].

Although the k-NN classifier is considered to be an effective algorithm, it has two major weaknesses that render its use irrelevant for large datasets: (i) it involves high computational cost, since all distances between the new, unclassified item and the training data must be estimated, and, (ii) storage requirements are high since it has to maintain the training data. These weaknesses have constituted an active research field over the last decades.

Multi-attribute indexing methods [38] can deal with the first weakness for datasets with moderate dimen-

Noname manuscript No.
(will be inserted by the editor)

RHC: Non-parametric Cluster-based Data Reduction for efficient k-NN Classification

the date of receipt and acceptance should be inserted later

Abstract Although the k-NN classifier is a popular classification method, it suffers from the high computational cost and storage requirements it involves. This paper proposes two effective cluster-based data reduction algorithms for efficient k-NN classification. Both have low preprocessing cost and can achieve high data reduction rates while maintaining k-NN classification accuracy at high levels. The first proposed algorithm is called Reduction through Homogeneous Clusters (RHC) and is based on a fast pre-processing clustering procedure that creates homogeneous clusters. The centroids of these clusters constitute the reduced training set. The second proposed algorithm is a dynamic version of RHC that retains all its properties and, in addition, it can manage datasets that cannot fit in main memory and is appropriate for dynamic environments where new training data are gradually available. Experimental results, based on fourteen datasets, illustrate that both algorithms are faster and achieve higher reduction rates than four known methods, while maintaining high classification accuracy.

Keywords k-NN Classification · Clustering · Data reduction · Prototypes

1 Introduction

Classification is an important data mining task that tries to assign new data to a set of predefined classes on the basis of the available training data, i.e., a set of already classified instances (or items) [24,26]. A typical classification example is to assign an email to either class "spam" or class "non-spam".

Address(es) of author(s) should be given

The k-Nearest Neighbour (k-NN) classifier [12,11] is an extensively used lazy classification algorithm. It is a simple and easy to implement classifier and can be exploited in many domains. The k-NN classifier does not build any classification model. The available training data is considered to be the classification model. Particularly, the algorithm classifies a new item x by searching in the training set for the k nearest items (neighbours) to x according to a distance metric (e.g., Eucledian distance). Then, x is assigned to the most common class determined via a majority vote of the retrieved k nearest neighbours. Ties are resolved either randomly or by the single nearest neighbour. The k-NN classifier is analytically tractable and for $k = 1$ and unlimited items the error rate is asymptotically never worse than twice the minimum possible, which is the Bayes rate [11].

Although the k-NN classifier is considered to be an effective algorithm, it has two major weaknesses that render its use irrelevant for large datasets: (i) it involves high computational cost, since all distances between the new, unclassified item and the training data must be estimated, and, (ii) storage requirements are high since it has to maintain the training data. These weaknesses have constituted an active research field over the last decades.

Multi-attribute indexing methods [38] can deal with the first weakness for datasets with moderate dimensionality (e.g., 2-10). In higher dimensions, the curse of dimensionality degrades their performance to the degree that sequential scans are more effective.

Data Reduction Techniques (DRTs)[1] [44,20,28,43, 47,27,23,7,31] can cope with both weaknesses since

[1] DRTs have two points of view: (i) item reduction, and, (ii) dimensionality reduction. We consider them from the first point of view.

they build a small representative set of the training data, which is usually called the condensing set. Applying the k-NN classifier using this small set, one has the benefit of much lower computational cost and storage requirements. A DRT is effective when accuracy does not degrade significantly. DRTs are distinguished into prototype selection [20] and prototype abstraction [44] algorithms. Prototype selection algorithms choose some training items as representatives (or prototypes) and place them into the condensing set. On the other hand, prototype abstraction approaches generate prototypes by summarizing similar training items.

DRTs can be compared to each other by using three evaluation criteria: (i) classification accuracy achieved by the k-NN classifier when run on the condensing set, (ii) reduction rate that indicates how much smaller the size of the condensing set is in comparison to the size of the training set (the higher the reduction rate, the faster the classification step), and (iii) preprocessing computational cost, that is, the cost to build the condensing set. Some DRTs can achieve good performance, but DRTs exhibit one or more of the following weaknesses:

1. They usually include a costly, time-consuming preprocessing step on the training set, which may be prohibitive for large datasets.
2. Many DRTs are parametric. The user has to provide the values for a number of parameters in advance. This usually involves tuning via an iterative execution of a trial-and-error procedure.
3. The resulting condensing set of many DRTs depends on the order of items in the training set. This means that some DRTs may build a different condensing set when processing the items of a specific training set in a different order.
4. Usually, there is a trade-off between data reduction and classification accuracy. Although some DRTs can achieve high reduction rates, the accuracy of the classifier is negatively affected. On the other hand, there are DRTs that produce condensing sets that achieve accuracies close to those achieved by the non-reduced training sets, but their reduction rates are not high.
5. Most DRTs are memory-based. This implies that the whole training set must reside in main memory. Thus, they are inappropriate for very large datasets that cannot fit into main memory or for devices with limited main memory (e.g., sensor devices).
6. Most DRTs cannot consider new training items after the construction of the condensing set. These DRTs are inappropriate for dynamic/streaming environments where new training items are gradually available.

To address the last two weaknesses one needs dynamic DRTs that are capable of updating their condensing set when additional training data segments become available after the construction of the condensing set and without requiring all previously used training items.

The aforementioned observations and the need for fast k-NN classification in large and high dimensional datasets constitute motivations of our work. In [35], we proposed an effective prototype abstraction algorithm. Our algorithm is able to cope with the first four weaknesses. Here, we extend our previous work by proposing a dynamic version of the algorithm that can cope with all weaknesses.

The complete contribution of our work is summarized as follows:

- We propose and evaluate a fast, non-parametric, independent of data order, and easy to implement prototype abstraction algorithm that achieves high reduction rates and accuracy measurements. The algorithm, which we call RHC (Reduction through Homogeneous Clusters) [35], is based on the well-known k-Means clustering algorithm [29,48], and thus, it can be easily integrated in many existing environments.
- We propose and evaluate a dynamic version of RHC (dRHC) that retains all the properties of RHC and, in addition, is capable of dynamically updating its condensing set. Thus, dRHC can deal with very large datasets and it is appropriate for dynamic/ streaming environments.

The rest of this paper is organized as follows. Section 2 briefly presents the related work. Section 3 considers in detail the proposed RHC algorithm and its dynamic variation. In Section 4, both algorithms are experimentally compared to known DRTs on fourteen datasets. The experimental results are validated by the Wilcoxon signed ranks test. Finally, Section 5 concludes the paper and provides pointers for future work.

2 Related work

A great number of DRTs have been proposed in the literature. Our purpose is not to extensively review all DRTs. We present in detail only the DRTs that we use in our experimental study in Section 4. For the interested reader, prototype abstraction and prototype selection algorithms are reviewed, categorized and compared to each other in [44] and [20]. Other relevant reviews can be found in [27,23,28,43,47,7,31].

The earliest and best known prototype selection algorithm is the Condensing Nearest Neighbour (CNN)

rule [25]. It tries to put only the, so called, close-class-border items in the condensing set. The basic idea behind CNN-rule is that, since the non-close-class-border (or central) items do not define decision boundaries, they can be removed without affecting accuracy. The algorithm uses two bins, S and T. Initially, a training item is placed in S while the remaining items are placed in T. Then, CNN-rule classifies the content of T using that of S by employing the 1-NN classifier. Whenever an item of T is misclassified, the assumption is that it lies in a close-class-border data area, and so, it is moved to S. When there is no move during a complete pass of T, the procedure terminates. The items that have been placed in S constitute the condensing set. Contrary to many other DRTs, CNN-rule automatically determines the size of the condensing set based on the level of noise in the training set and the number of classes (or, in other words, the number of boundaries). A weak point is that the resulting condensing set depends on the order of items in the training set.

Many other DRTs either extend CNN-rule or are based on the same idea, e.g., Reduced NN rule [22], Selective NN rule [37], Modified CNN rule [16], Generalized CNN rule [10], Fast CNN [4,5], Tomek's CNN rule [42] and the IBL algorithms [2,1]. However, CNN-rule continues to be the reference prototype selection algorithm and is used for comparison purposes in many research papers.

Prototype Selection by Clustering (PSC) [32,34,33,36] is a recently proposed prototype selection algorithm whose main goal is fast execution of the reduction procedure rather than high reduction rates. PSC applies the k-Means clustering algorithm [29,48] in order to find clusters in the training set. For the homogeneous clusters (i.e., clusters that contain only items of a specific class), it keeps only the nearest to the centroid training item. For each non-homogeneous cluster, it retains only the items that define the decision boundaries between different classes in the cluster. A disadvantage of PSC is that the user has to determine the number of clusters that will be created through a trial-and-error procedure and also the choice of the initial means affect the contents of the constructed condensing set.

Some other known cluster-based DRTs are the Self Generating Prototypes (SGP) [18], the Symbolic Nearest Mean Classifier (SNMC) [14], and the Generalized Modified Chang's Algorithm (GMCA) [30]. Contrary to PSC, they are prototype abstraction approaches. GMCA is based on the idea of the earliest and well-known prototype abstraction algorithm introduced by Chang [8].

Some prototype selection algorithms attempt to improve accuracy rather than achieve high reduction rates. This is achieved by removing noisy and close-class-border items and, thus, leaving smoother decision boundaries. These methods are called editing algorithms and, although they have a completely different motivation, they constitute a subcategory of prototype selection algorithms. The reduction rates of many DRTs depend mainly on the level of noise in the training data. Therefore, an editing routine should be applied before the application of the main reduction procedure [13,28,43].

The reference editing algorithm is ENN-rule [46]. It removes the irrelevant items by using the following rule: a training item is removed, if its class does not agree with the majority of its k nearest neighbours. Consequently, ENN-rule needs to compute all distances among the training items, i.e., $\frac{N*(N-1)}{2}$ distances. All k-NN [41], Repeated ENN-rule [41] and Multiedit [17] are well-known variations of ENN-rule. Multiedit usually constitutes a time consuming editing algorithm. We note that some algorithms are called hybrid because they integrate the idea of editing in their main reduction procedure (see [20,44] for details).

IB2 [2] is a one-pass version of CNN-rule that can dynamically update its condensing set. This means that new training items can be taken into account after the construction of the condensing set and without requiring the previously used training items. Because of this property and contrary to all other algorithms presented in this section, IB2 can manage very large datasets that cannot fit in main memory. Therefore, it can be characterized as a dynamic algorithm. IB2 begins by putting the first available training item in the condensing set. Then, whenever a new training item x is available, it is classified by employing the 1-NN classifier on the content of the current condensing set. if x is misclassified, it is put in the condensing set. Otherwise, it is removed. Since IB2 is an one-pass algorithm, it is very fast. In addition, like CNN-rule, IB2 determines the size of the condensing set automatically. The drawback is that the resulting condensing set highly depends on the order that the training items arrive. We note that IB3 [2] is a variation of IB2 that integrates an editing mechanism in its reduction procedure. Thus, IB3 is a hybrid approach.

The family of Reduction by Space Partitioning algorithms (RSP) [39] constitutes a popular set of three prototype abstraction algorithms known as RSP1, RSP2 and RSP3. They can be characterized as extensions of the Chen and Jozwik algorithm (CJA) [9]. The latter, initially retrieves items A and B that define the dataset's diameter, i.e., they are the most distant items in the training set. Then, it divides the training set into two subsets. One subset contains the items nearest to A, while the other the ones nearest to B. CJA continues by dividing subsets that contain items of more than one

classes (non-homogeneous subsets). It divides the non-homogeneous subset with the largest diameter first. If all subsets are homogeneous, CJA divides the homogeneous subsets by also using the criterion of the largest diameter. Divisions are repetitively executed until the number of created subsets is equal to a user-defined threshold. Finally, for each created subset S, CJA computes a mean item (centroid) by averaging the items in S. This centroid is labelled by the most common class in S and is placed in the condensing set as prototype.

Contrary to CJA, RSP1 computes as many centroids as the number of different classes in each subset. Obviously, it builds a larger condensing set than CJA. However, it can achieve higher accuracy. RSP1 and RSP2 differ to each other on how they choose the next subset that should be split. RSP1 uses the criterion of the larger diameter while RSP2 uses the criterion of overlapping degree [39]. RSP3 does not depend on the criterion used. It continues splitting the non-homogeneous subsets and terminates when all of them become homogeneous. RSP3 is the only RSP algorithm (CJA included) that is non-parametric. It automatically determines the size of the condensing set. In addition, all RSP algorithms (CJA included) are independent on the order of items in the training set.

3 Proposed Methods

3.1 Reduction through Homogeneous Clusters

The Reduction through Homogeneous Clusters (RHC) algorithm is a prototype abstraction non-parametric algorithm. It is based on a simple idea that recursively applies the well-known k-Means clustering algorithm [2] [29, 48]. Particularly, RHC keeps on constructing clusters until all of them are homogeneous, i.e., they contain items only of a specific class.

Initially, RHC considers the whole training set as a non-homogeneous cluster. The algorithm begins by computing the mean for each class by averaging the attribute values of the corresponding training items. Therefore, for a dataset with n classes, the algorithm computes n class means. Then, RHC executes k-Means clustering using the n aforementioned class means as initial means and builds n clusters. For each homogeneous cluster, its mean (centroid) is placed in the condensing set as prototype. For each non-homogeneous cluster, the above procedure is applied recursively. The algorithm stops when all clusters are homogeneous. In the end, the condensing set contains all the centroids of the homogeneous clusters. Note that using the class means as initial means for k-Means clustering, the number of clusters is determined automatically.

The mean item m for a cluster or class C is computed by averaging the n attribute values of items x_i, $i = 1, 2 \ldots |C|$ that belong to C. More formally, the j^{th} attribute $m.d_j$ of m is estimated as follows:

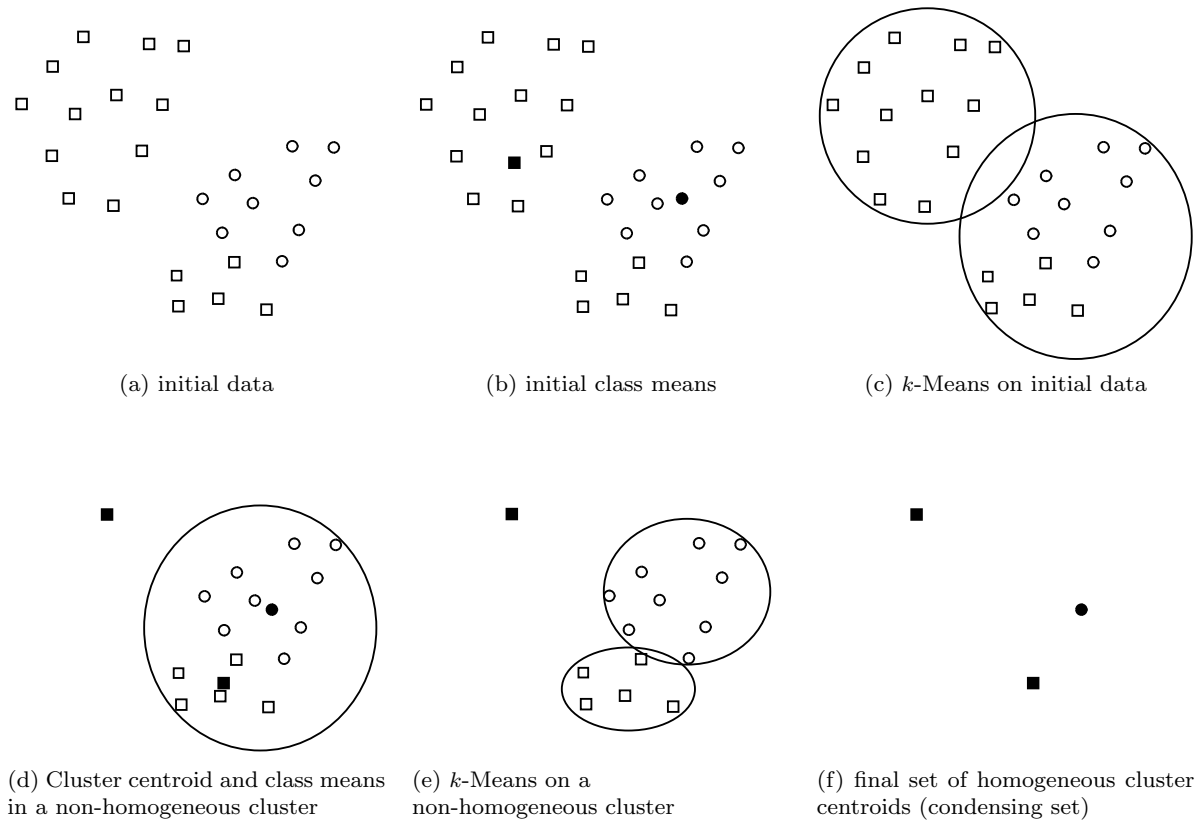
$$m.d_j = \frac{1}{|C|} \sum_{x_i \in C} x_i.d_j, j = 1, 2, \ldots, n$$

Figure 1 presents a two-dimensional example of RHC execution. Suppose that a dataset contains twenty six items of two classes: squares and circles (Figure 1(a)). RHC computes a class mean for the squares and a class mean for the circles (Figure 1(b)). Then, k-means clustering uses the two class-means as initial means and constructs two clusters. One cluster contains only squares while the other cluster contains items of both classes (Figure 1(c)). For the homogeneous cluster, RHC stores the cluster centroid in the condensing set as a prototype of class square (Figures 1(d)). For the items of the non homogeneous cluster, RHC recursively builds two homogeneous clusters (Figures 1(d,e)). Consequently, two more prototypes are stored in the condensing set. Thus, the final condensing set contains three prototypes instead of the twenty six items of the initial training set (Figure 1(f)).

Obviously, RHC generates many prototypes for close-class-border data areas and few prototypes for the "central" class data areas. Therefore, the more the classes and the noise in the data, the more borders exist, and thus, the lower reduction rate is achieved. Moreover, by using the class means as initial means for the k-Means clustering, RHC increases the probability of quickly finding large homogeneous clusters and achieving a high reduction rate (the larger the homogeneous clusters constructed, the higher the reduction rate achieved).

Algorithm 1 shows a non-recursive RHC implementation. It uses a queue data structure, $Queue$, to hold unprocessed clusters. Initially, the whole training set (TS) constitutes an unprocessed cluster and is put in $Queue$ (line 3). At each repeat-until iteration, RHC dequeues cluster C from the head of $Queue$ (line 7) and checks whether C is homogeneous or not. If it is (line 8), its centroid is placed in the condensing ser (CS) as a prototype (line 10) and its items are removed. Otherwise, RHC computes a list of class means (M), one for each of the distinct classes that exist in C (lines 13–16). Then, RHC calls k-Means, with parameters the current non homogeneous cluster C and the list of the initial class means M to be used as initial means. The result is a new set of unprocessed clusters ($NewClusters$) (line 17) all of which are put into $Queue$ (lines 18–20). The repeat-until loop continues until $Queue$ becomes empty

[2] One should not confuse k of k-NN with k of k-Means.



(a) initial data (b) initial class means (c) k-Means on initial data

(d) Cluster centroid and class means in a non-homogeneous cluster (e) k-Means on a non-homogeneous cluster (f) final set of homogeneous cluster centroids (condensing set)

Fig. 1 Data abstraction using RHC

(line 22), i.e., there are no more non-homogeneous clusters.

In effect, RHC combines the idea of RSP3 [39] with that of PSC [32,34,33]. It retains their advantages and avoids their weak points. Let's recall that PSC is a fast and parametric algorithm, while RSP3 is a non-parametric algorithm that involves high pre-processing cost due to the procedure for finding the most distant items in each subset. Thus, RSP3 is inappropriate for large datasets. Contrary to PSC, RHC is a non-parametric algorithm. Contrary to RSP3, RHC is fast since it is based on a version of k-Means that is sped-up by the class mean initializations. Note that we have adopted the full cluster consolidation for the stopping condition of k-Means clustering (no item re-assignment during a complete pass of data). RHC could have become even faster had we used a more efficient stopping condition. In addition, contrary to CNN-rule [25], IB2 [2] and many other methods, the effectiveness of RHC does not depend on the order of training items.

3.2 Dynamic RHC

Like most DRTs, RHC is a memory based technique. This implies that the whole training set must be resident in main memory. RHC cannot manage datasets that cannot fit in main memory. Therefore, it cannot be executed on a device with limited main memory, without transferring data to a server over a network for processing. This is a costly and time-consuming procedure.

In addition, RHC cannot handle new training items, i.e., it cannot update its condensing set in a dynamic manner. Suppose that RHC is executed over a dataset D and builds a condensing set. Then, suppose that a data segment S with new training items is available and should be taken into consideration. For the construction of an updated condensing set, RHC must be executed from scratch over the complete dataset $D \cup S$. This procedure must be repeated whenever a new data segment is available. In such a dynamic environment, all the items of the training set, or at least a recent set of them, must always be available. In other words, storage requirements remain high.

Dynamic RHC (dRHC) is a dynamic variation of RHC. It retains all the advantages of RHC discussed in Subsection 3.1. But, it can cope with the weak points of RHC by considering the available data in the form of data segments. On one hand, if the dataset does not fit in main memory, it is divided into data segments ap-

Algorithm 1 RHC

Input: TS
Output: CS

1: {**Stage 1: Queue Initialization**}
2: $Queue \leftarrow \varnothing$
3: Enqueue($Queue$, TS)
4: {**Stage 2: Construction of the condensing set**}
5: $CS \leftarrow \varnothing$
6: **repeat**
7: $C \leftarrow$ Dequeue($Queue$)
8: **if** C is homogeneous **then**
9: $r \leftarrow$ mean of C
10: $CS \leftarrow CS \cup \{r\}$
11: **else**
12: $M \leftarrow \varnothing$ {M is the set of class means}
13: **for** each class L in C **do**
14: $m_L \leftarrow$ mean of L
15: $M \leftarrow M \cup \{m_L\}$
16: **end for**
17: $NewClusters \leftarrow K$-MEANS(C, M)
18: **for** each cluster $NC \in NewClusters$ **do**
19: Enqueue($Queue$, NC)
20: **end for**
21: **end if**
22: **until** IsEmpty($Queue$)
23: **return** CS

propriate for the available main memory. On the other hand, in dynamic and/or streaming environments, since training items arrive continuously, they can be considered as data segments. In this case, the concept of data segment is implemented by using a buffer, where the new training items are stored. When the buffer is full, dRHC is run over it. Then, the training items stored in the buffer are removed and the buffer is ready to receive new training items.

The execution of dRHC has two phases: (i) *initial condensing set (CS) construction*, and, (ii) *condensing set (CS) update*. The *initial CS construction* phase is executed only once, while the *CS update* phase is executed for each arriving data segment. Figure 2 depicts the complete procedure of dRHC. The algorithm begins with the *initial CS construction* phase on the available training set (TS Data Seg. 1). The procedure is almost similar to RHC execution (see Algorithm 1). The only difference between RHC and the first phase of dRHC is that the latter, for each generated prototype, stores a weight value as an extra attribute. This value is the number of the training items that were clustered together and are represented by the specific prototype in the condensing set.

The *CS update* phase is also based on the concept of cluster homogeneity and the data weights. In particular, while the original RHC and the *initial CS construction* phase of dRHC begin with a single cluster that contains all the items of the training set, the *CS update* phase of dRHC uses the prototypes of the existing condensing set and a data segment to construct a set of initial clusters and then proceeds similarly to RHC.

Algorithm 2 presents the *CS update* phase of dRHC. It takes an already constructed condensing set ($oldCS$) and a new data segment ($dataSeg$) and returns an updated condensing set ($newCS$). The algorithm begins by building the queue of unprocessed clusters. First, it initializes as many clusters, as the number of prototypes in $oldCS$ (lines 3–6). Then, each item x of $dataSeg$ is assigned to one of these clusters (lines 7–11). Finally, the clusters are enqueued to the queue data structure (lines 12–14).

The algorithm proceeds to generate $newCS$ in a way analogous to RHC, but taking into account the weight values. For a homogeneous cluster, the prototype stored in $newCS$ is the weighted mean (centroid) of the cluster (lines 19–22). Similarly, for a non-homogeneous cluster C, each class mean is computed as the weighted mean of the class items (lines 24–29). These class means play the role of the initial means in the call to k-Means for that cluster (line 30). Note that in the case of dRHC, we use a version of k-Means that also takes into account the item weights in the determination of the cluster centroids. For a cluster (or class) C, each vector attribute d_j, $j = 1, 2, \ldots, n$ of its centroid m_C (lines 20, 26) is estimated as follows:

$$m_C.d_j = \frac{\sum_{x_i \in C} x_i.d_j \times x_i.weight}{\sum_{x_i \in C} x_i.weight}$$

Old prototypes (from $oldCS$) usually have weights that are greater than one and have higher influence in the computation of a new weighted class mean or cluster centroid than any item of a new data segment, whose weight is one.

Figure 3 presents an example of the *CS update* phase. Suppose that an already constructed condensing set is available (Figure 3(a)). It contains the prototypes generated in the example of Figure 1 with the corresponding weight values. Moreover, suppose that a new data segment with seven new training item arrives (Figure 3(b)). Certainly, their initial weight is equal to one. Initially, dRHC assigns each new item to the cluster of the nearest prototype (Figure 3(c)). Since, no item was assigned to cluster B, the corresponding prototype is not modified. Although new items were assigned to cluster A, A remains homogeneous. Therefore, the weighted mean (centroid) of A is computed and placed in the con-

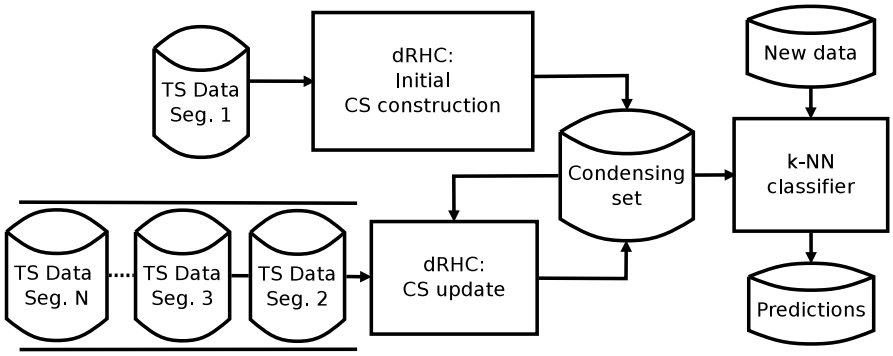


Fig. 2 Classification procedure with dRHC

Algorithm 2 dRHC: CS update phase

Input: $oldCS$, $dataSeg$
Output: $newCS$

{**Stage 1: Queue Initialization**}
$Queue \leftarrow \varnothing$
$CList \leftarrow \varnothing$ {empty list of clusters}
for each prototype $m \in oldCS$ **do**
  add new cluster $C = \{m\}$ in $CList$
end for
for each item $x \in dataSeg$ **do**
  $x.weight = 1$
  find $C_x \in CList$ with the nearest to x mean
  $C_x \leftarrow C_x \cup \{x\}$ {do not recompute mean of C_x}
end for
for each cluster C in $CList$ **do**
  Enqueue($Queue$, C)
end for
{**Stage 2: Construction of newCS**}
$newCS \leftarrow \varnothing$
repeat
  $C \leftarrow$ Dequeue($Queue$)
  if C is homogeneous **then**
    $m \leftarrow$ weighted mean of C
    $m.weight \leftarrow \sum_{x_i \in C} x_i.weight$
    $newCS \leftarrow newCS \cup \{m\}$
  else
    $M \leftarrow \varnothing$ {M is the set of weighted class means}
    for each class L in C **do**
      $m_L \leftarrow$ weighted mean of L
      $m_L.weight \leftarrow \sum_{x_i \in L} x_i.weight$
      $M \leftarrow M \cup \{m_L\}$
    end for
    $NewClusters \leftarrow K$-MEANS(C, M)
    for each cluster $NC \in NewClusters$ **do**
      Enqueue($Queue$, NC)
    end for
  end if
until IsEmpty($Queue$)
return $newCS$

densing set along with its new weight. In effect, the old prototype slightly "moves" towards the new items (Figure 3(d)). Cluster C is non-homogeneous. This means that at least one new prototype will be generated. A weighted class mean is computed for each class in C (Figure 3(d)) and k-Means is executed. The result is the construction of two homogeneous clusters (Figure 3(e)). The weighted mean of each cluster is computed and placed in the condensing set along with its weight (Figure 3(f)).

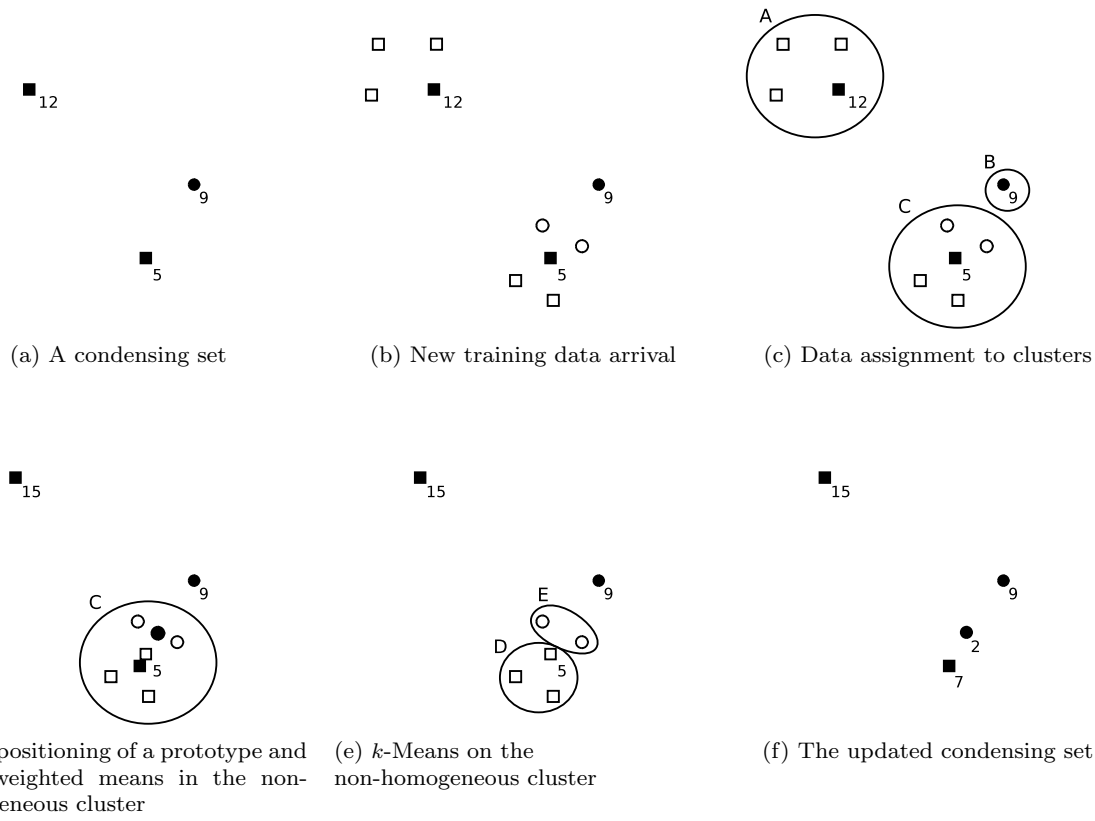
Considering dRHC, we realize that the *initial CS construction* phase is more expensive than an execution of the *CS update* phase. This is because the *initial CS construction* phase begins from scratch without already constructed clusters. It begins by considering the whole dataset as an unprocessed cluster and a small number of class means as initial means for the k-means clustering. Thus, to obtain homogeneous clusters a high number of k-means executions is needed. In contrast, the *CS update* phase begins by assigning new data to already constructed clusters. Now, the probability of having a homogeneous cluster after the new data assignment is high. Of course, the probability of getting homogeneous clusters depends on the level of noise in the data.

We should mention that dRHC creates different condensing sets by examining the data segments in different order. However, the order of data into each data segment is irrelevant. Finally, we note that although dRHC can deal with fast and large data streams, it does not take into account the phenomenon of concept drift [45] that may exist in data streams. IBL-DS [6] is a DRT that can deal with this phenomenon.

4 Performance Evaluation

4.1 Experimental setup

The proposed algorithms were evaluated using fourteen datasets distributed by the KEEL dataset repos-



(a) A condensing set (b) New training data arrival (c) Data assignment to clusters

(d) Repositioning of a prototype and class weighted means in the non-homogeneous cluster (e) k-Means on the non-homogeneous cluster (f) The updated condensing set

Fig. 3 Data abstraction using dRHC (CS update phase)

itory[3] [3]. They are summarized in Table 1. For comparison purposes, we implemented three prototype selection algorithms, namely, CNN-rule [25], IB2 [2,1], and PSC [32,33,36], and a prototype abstraction approach, namely, RSP3 [39]. We selected the aforementioned methods because: (i) CNN-rule and RSP3 are popular algorithms that are usually used in many research papers for comparison purposes, (ii) IB2, PSC and RHC have the same goal, that is, fast execution of the reduction procedure (or, low preprocessing cost), (iii) like RSP3 and PSC, RHC is based on the idea of homogeneity, and, (iv) IB2 is a fast DRT that dynamically builds its condensing set and, thus, it is appropriate to be compared with dRHC.

All algorithm implementations were written in C and the Euclidean distance was adopted as the distance metric. The thirteen datasets (except KDD) were used without data normalization. The MGT, LS, TXR and ECL datasets are distributed sorted on the class label and this affects the methods that depend on the order of data. Consequently, we randomized the order of the data items for these datasets. With the exception of the KDD dataset, no other data transformation was performed. All experiments were conducted without previous knowledge about the datasets such as data distribution, level of noise, etc.

For each dataset and algorithm, we report three average measurements obtained via five-fold cross-validation: (i) Accuracy, (ii) Reduction Rate, and, (iii) Preprocessing Cost in terms of distance computations. We report the classification accuracy of k-NN for $k = 1$ (1-NN). For all datasets (except KDD), we used the five already constructed pairs of training/testing sets hosted by the KEEL repository.

The original form of the KDD dataset contains 41 attributes. However, for simplifying our experimentation procedure, we removed the three nominal and the two fixed-value attributes that exist in the dataset. In addition, many data items are duplicates. The KDD dataset contains 494,020 items, but only 141,481 of them are unique. Thus, we removed all duplicate items. Actually, duplicates are useless especially in the context of 1-NN classification. They do not influence classification accuracy and negatively affect computational cost. Note that removal of duplicates is a common process[4]

[3] `http://sci2s.ugr.es/keel/datasets.php`

[4] Many datasets distributed by the KEEL repository have been preprocessed to remove duplicates

Table 1 Datasets description

Dataset	Size	Attributes	Classes	Memory/Buffer size
Letter Recognition (LR)	20000	16	26	2000
Magic G. Telescope (MGT)	19020	10	2	1902
Pen-Digits (PD)	10992	16	10	1000
Landsat Satellite (LS)	6435	36	6	572
Shuttle (SH)	58000	9	7	1856
Texture (TXR)	5500	40	11	440
Phoneme (PH)	5404	5	2	500
KddCup (KDD)	494020/141481	36	23	1000
Balance (BL)	625	4	3	100
Banana (BN)	5300	2	2	530
Ecoli (ECL)	336	7	8	200
Yeast (YS)	1484	8	10	396
Twonorm (TN)	7400	20	2	592
MONK 2 (MN2)	432	6	2	115

and has been adopted by the DRT presented in [49]. Furthermore, the attribute value ranges of the KDD dataset vary extremely. Therefore, we decided to normalized them to the range [0-1]. Then, we randomized the transformed KDD dataset and divided it into the appropriate pairs for training/testing sets.

Although duplicates are useless during classification, they influence the construction of the condensing set. In particular, CNN-rule and IB2 build the same condensing set regardless the number of duplicates in the training set but with higher preprocessing cost. In contrast, the condensing sets built by RSP3, PSC, RHC, dRHC are influenced by duplicates because they contribute to the estimation of the mean items. For that reason, our experimental study also includes the original form of the KDD dataset(without discarding the duplicates).

In addition, we wanted to evaluate the proposed algorithms on noise-free data. Thus, we ran our tests twice using an edited and a non-edited version of the training set. For editing purposes, we implemented ENN-rule [46] and based on [47,21,31], we set $k = 3$. Certainly, we did not edit the testing portions of each fold. The KDD, BL, ECL and YS datasets contain some rare or weak classes. ENN-rule eliminates some of these. It is worth mentioning that the execution of ENN-rule over the KDD dataset is an extremely time consuming procedure. It computes $\frac{113,185\times113,184}{2}\times5$ folds $\simeq 32$ billion distances. The SH dataset also contains rare classes. However, editing did not eliminate any rare class and, thus, we decided to include the editing procedure for that dataset in our experimentation.

Apart from PSC, all algorithms are non-parametric. For the parameter c of PSC (number of clusters built), we run experiments by building $c = r\times j$, $j = 2, 4, \cdots, 10$, clusters, where r is the number of discrete classes in the data, as Lopez et al. did in their experiments [32]. Hence, we built five PSC based classifiers for each dataset.

The complete procedure that we followed during our experiments is shown in Figure 4. In addition to using condensing sets, we also measured the performance of the 1-NN classifier on the initial and edited training sets.

Contrary to all other methods, dRHC considers data in segments. To obtain data in a such form, we split the training sets of the datasets into segments. The last column of Table 1 shows the size of the data segment. Note that in some datasets, the last data segment may not be complete. The size of the data segment corresponds to either the size of the available main memory (scenario of limited main memory) or the size of the data buffer (scenario of streaming/dynamic environments). The purpose of the experiment was not to test the method in a real life situation regarding memory sizes, but to assess the performance of classification on condensing sets that are constructed in a dynamic manner. Therefore, the sizes of the data segments used do not correspond to actual memory sizes. Of course, actual memory sizes can be used in real life situations.

4.2 Experimental measurements

The results of our experiments are presented in Tables 2–4 and Tables 5–7 for the non-edited and the edited datasets, respectively. Best measurements are in

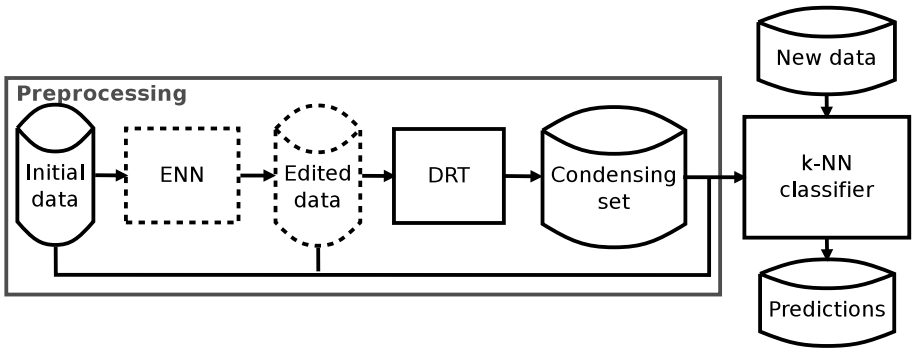


Fig. 4 Experimental classification procedure

bold. Preprocessing Cost measurements are in million distance computations (M). For reference, in Tables 2 and 5, we report the accuracy values of the conventional k-NN classifier (the 1-NN classifier applied on the non-edited training set). In addition, we present the measurements of ENN-rule. The latter reveal the level of noise in the datasets. We note that, in Table 7, preprocessing cost measurements of CNN, IB2, RSP3, PSC, RHC and dRHC algorithms do not include the cost of editing. In these cases, the total preprocessing cost can be computed by adding the preprocessing cost measurements of the ENN column. In addition, reduction rates presented in Table 6 correspond to the total reduction rate: editing and data reduction. We note that the measurements of dRHC are estimated after the arrival of all data segments.

An immediate observation is that RHC and dRHC have low preprocessing cost. In almost all cases, the preprocessing cost of dRHC is lower than that of IB2 and PSC, whose major goal is to reduce the preprocessing cost.

Almost in all cases, RHC and dRHC achieve the highest reduction rates. This means that the 1-NN classifier executes faster when using a condensing set generated by these algorithms. Our measurements confirm that RSP3 is a time-consuming approach that achieves low reduction rates. However, in many cases, RSP3 has the highest accuracy, which is very close to the one measured for the conventional 1-NN classifier. RHC and dRHC appear to be more accurate than IB2 and PSC and as accurate as CNN-rule. RHC and dRHC achieved the highest accuracies in five datasets (see the BN and MN2 datasets in Table 2 and the BN, TN and MN2 datasets in Table 5). In the case of the MN2 dataset, RHC and dRHC are more accurate even than the conventional 1-NN and ENN-rule. Although the rest accuracy measurements of RHC and dRHC are not as good as those of RSP3, they are close enough.

Although IB2 is an one-pass version of CNN-rule, it achieved higher reduction rates with much lower preprocessing cost. However, CNN-rule seems to be more accurate than IB2. It is worth mentioning that the measurements for both algorithms would be different had we examined the same data in a different order.

Concerning the differences between Tables 2–4 and Tables 5–7 (non-edited vs edited data), we observe that except the MGT, BN and TN datasets the reported accuracies are almost similar. On the other hand, all DRTs that ran over edited data, executed faster and generated smaller condensing sets. In the cases of the MGT, LS, PH and BN datasets, ENN-rule removed many irrelevant items ($> 9\%$) and so, the reduction rate differences are obvious. It is worth noting that in the case of edited data, dRHC, RHC and IB2 generate their condensing set by calculating an extremely low number of distances (see Table 7). In addition, we observe that when RHC and dRHC are executed on edited data they are able to create extremely small condensing sets (see Table 6). They contain less than 5% of items of the initial training set on average. Considering the differences between the performance of RHC and dRHC, we conclude that dRHC may be characterized as a slightly better approach than RHC.

We can make a final comment concerning the average measurements (last rows in Tables 2–7): Almost in all cases, RHC and dRHC appear to build the smallest condensing sets with the lowest preprocessing cost, and a high classification accuracy, similar to that of CNN-rule.

4.3 Complementary experiments

As we have already mentioned, in the context of 1-NN classification, duplicates do not influence classification accuracy but affect the computational cost. However, they influence the data reduction procedures. Consequently, we wanted to measure the performance of DRTs

Table 2 Experimental Results on non-edited datasets: Accuracy (%)

Dataset	1-NN	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	95.83	94.98	92.84	91.98	**95.43**	82.73	85.65	87.14	87.73	88.67	93.59	93.93
MGT	78.14	80.44	74.54	71.97	**74.69**	63.51	63.95	63.95	64.28	64.24	71.97	72.97
PD	99.35	99.30	**98.68**	98.04	99.05	95.73	96.64	96.26	96.90	96.93	98.30	98.49
LS	90.60	90.29	88.21	86.87	**90.57**	82.42	83.29	83.93	83.90	84.32	88.95	88.50
SH	99.82	99.79	**99.76**	99.73	99.75	99.67	98.24	97.93	98.82	95.96	98.09	99.65
TXR	99.02	98.64	97.16	96.35	**98.29**	96.13	94.96	94.84	94.46	94.78	97.04	97.60
PH	90.10	88.14	**87.82**	85.57	86.94	71.41	75.19	75.17	74.70	75.63	85.59	85.38
KDD	99.71	-	**99.66**	99.48	99.60	95.50	96.18	96.68	96.89	96.95	99.39	99.42
BL	78.4	-	70.88	70.72	**73.28**	65.92	66.40	70.88	68.00	68.32	68.64	70.56
BN	86.91	89.36	**85.62**	83.81	84.00	57.60	58.00	56.87	57.49	58.70	83.28	82.79
ECL	79.78	-	72.05	66.97	**73.53**	57.16	63.39	66.97	68.16	66.37	68.76	69.35
YS	52.02	-	49.06	46.02	**50.47**	46.03	45.01	47.84	46.77	47.71	48.85	48.38
TN	94.88	95.69	92.00	89.15	92.68	78.74	79.08	79.78	80.49	80.12	88.69	**93.08**
MN2	90.51	89.58	95.84	93.75	91.22	94.43	95.14	90.06	92.58	93.52	94.68	**97.68**
Avg	88.22	92.62	86.01	84.32	**86.39**	77.64	78.65	79.16	79.37	79.44	84.70	85.56

Table 3 Experimental Results on non-edited datasets: Reduction Rate (%)

Dataset	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	4.33	83.54	85.66	61.98	81.40	79.76	79.46	79.88	79.90	88.08	**88.18**
MGT	20.08	60.08	70.60	53.70	70.71	71.05	71.58	71.81	71.60	73.76	**74.62**
PD	0.67	95.36	96.23	89.22	91.44	92.86	93.73	94.42	94.83	96.52	**97.23**
LS	9.07	80.22	84.62	73.19	84.67	84.79	84.84	84.93	84.95	**89.84**	88.35
SH	0.18	99.37	99.44	98.59	96.88	97.68	97.87	98.33	98.54	**99.55**	99.50
TXR	1.24	91.90	93.33	83.31	86.81	89.33	90.62	91.29	91.54	94.70	**94.95**
PH	11.25	76.04	80.85	69.94	81.31	81.56	81.32	81.39	81.54	80.71	**82.34**
KDD	-	99.12	**99.26**	98.54	99.13	99.09	99.09	99.09	99.07	99.19	99.22
BL	-	65.72	69.36	64.64	77.8	77.44	78.04	77.2	75.88	78.00	**78.12**
BN	11.53	77.44	83.27	75.21	85.59	85.70	85.77	**85.89**	85.81	79.68	82.41
ECL	-	59.55	68.77	52.27	74.50	72.19	71.08	67.88	65.65	67.58	**70.26**
YS	-	32.68	44.82	27.36	**55.32**	55.25	53.84	53.81	54.23	49.83	51.23
TN	3.61	82.09	88.25	84.56	95.73	94.85	94.57	94.78	94.98	**96.63**	95.37
MN2	2.08	87.23	91.68	61.33	45.31	49.02	61.16	57.34	60.23	96.47	**96.88**
Avg	6.40	77.88	82.58	70.99	80.47	80.76	81.64	81.29	81.34	85.04	**85.62**

using the original form of the KDD dataset that contains a high number of duplicates. Especially for dRHC that dynamically builds its condensing set based on a weight-based schema, this experiment is essential. Hence, we ran experiments using the original form of the KDD dataset (494020 items). By applying five-fold cross validation, we obtained the measurements presented in Table 8.

Applying the preprocessing of CNN-rule, RSP3 and PSC on such a large dataset is an extremely time consuming procedure. Especially the execution of RSP3 is prohibitive, due to the costly retrieval of the most distant points in each subset. We decided to exclude those DRTs from this phase of our experimentation because of their high preprocessing cost. Certainly, conventional 1-NN classification is also extremely time-consuming. It needs to compute over 39 billions distances for each fold, and thus, we did not consider it in this experiment.

The results shown in Table 8 are very similar to the ones presented in Tables 2–4 concerning the KDD dataset. Of course, since the original KDD dataset contains over 70% more items (duplicates), reduction rates and preprocessing cost measurements in Table 8 are higher than those in Tables 3 and 4. Furthermore, since

Table 4 Experimental Results on non-edited datasets: Preprocessing Cost (millions of distance computations)

Dataset	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	127.99	163.03	23.37	326.52	66.32	110.06	129.16	165.32	169.92	41.85	**19.57**
MGT	115.76	281.49	34.61	511.67	23.95	17.21	22.68	27.09	33.47	**4.08**	26.03
PD	38.65	11.75	1.78	86.66	6.52	15.93	28.48	35.23	36.97	2.88	**1.44**
LS	13.25	17.99	2.22	37.70	2.96	5.85	8.41	10.11	10.50	1.69	**1.53**
SH	1076.46	45.30	8.26	17410.18	127.20	54.07	148.35	222.77	252.61	16.83	**7.68**
TXR	9.68	5.65	0.84	27.63	3.15	7.90	10.71	14.49	16.76	3.63	**0.68**
PH	9.35	13.45	1.96	20.31	1.08	0.94	2.08	2.79	3.12	**0.66**	1.64
KDD	-	384.90	**55.58**	20278.87	212.23	575.80	1161.43	2054.23	1902.41	81.59	57.40
BL	-	0.21	0.04	0.3	0.08	0.12	0.16	0.18	0.24	0.05	**0.03**
BN	8.99	11.49	1.58	18.76	1.91	1.44	2.39	4.63	4.37	**0.56**	1.53
ECL	-	0.06	**0.003**	0.08	0.06	0.11	0.11	0.12	0.15	0.03	0.02
YS	-	1.41	**0.19**	2.12	0.70	1.17	1.64	1.94	1.99	0.84	0.31
TN	17.52	22.13	2.07	37.13	1.76	5.40	6.76	6.93	8.37	1.64	**0.70**
MN2	0.06	0.04	0.006	0.13	0.014	0.07	0.08	0.12	0.13	0.007	**0.004**
Avg	141.77	68.49	9.46	2768.43	32.00	56.86	108.75	181.85	174.36	11.17	**8.47**

Table 5 Experimental Results on edited datasets: Accuracy (%)

Dataset	1-NN	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	95.83	94.98	92.06	91.38	**94.61**	82.29	85.68	87.00	87.97	88.46	92.72	93.14
MGT	78.14	80.44	**79.26**	78.01	79.09	72.50	72.71	73.33	73.31	73.35	77.78	78.33
PD	99.35	99.30	98.60	98.17	**99.03**	97.30	97.04	97.11	97.29	97.11	98.45	98.57
LS	90.60	90.29	88.66	88.05	**89.90**	83.53	84.60	84.91	84.85	84.99	89.14	88.81
SH	99.82	99.79	**99.73**	99.72	99.67	99.56	98.40	98.53	98.82	98.41	99.58	99.62
TXR	99.02	98.64	96.93	95.75	**97.91**	96.15	95.46	95.26	94.91	95.67	97.11	97.38
PH	90.10	88.14	**86.88**	86.33	86.49	80.74	81.07	81.75	81.42	81.70	85.40	85.55
BN	86.91	89.36	88.87	88.68	88.64	81.98	81.51	82.26	80.68	80.79	88.09	**88.94**
TN	94.88	95.69	92.30	91.22	94.69	82.58	83.14	83.77	85.23	85.49	93.11	**95.45**
MN2	90.51	89.58	95.37	94.46	90.07	95.13	93.98	94.90	93.06	94.21	**96.75**	96.31
Avg	92.52	92.62	91.87	91.18	92.01	87.18	87.36	87.88	87.75	88.02	91.81	**92.21**

Table 6 Experimental Results on edited datasets: Reduction Rate (%)

Dataset	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	4.33	87.75	88.88	66.12	81.95	80.25	80.14	80.80	81.22	90.34	**91.00**
MGT	20.08	90.09	92.05	84.20	85.57	85.67	86.61	86.57	86.63	93.06	**93.40**
PD	0.67	96.44	97.00	90.41	91.95	93.50	94.22	95.11	95.70	97.19	**97.79**
LS	9.07	91.44	92.98	85.84	90.25	90.65	90.95	91.26	91.48	**95.09**	94.94
SH	0.18	99.58	99.61	98.88	97.10	97.89	98.04	98.55	98.68	**99.66**	99.65
TXR	1.24	93.45	94.32	85.00	87.82	90.50	91.76	92.60	92.42	95.58	**95.85**
PH	11.25	90.49	91.62	85.13	87.70	88.04	87.80	87.94	87.91	92.10	**92.43**
BN	11.53	95.31	95.87	93.72	95.66	95.78	96.02	**96.28**	96.40	95.66	95.87
TN	3.61	89.49	92.36	89.63	98.55	98.28	98.07	98.02	97.88	98.52	97.85
MN2	2.08	88.84	93.12	62.25	44.34	53.24	60.92	61.16	62.95	**97.05**	96.94
Avg	6.40	92.29	93.78	84.12	86.09	87.38	88.45	88.83	89.13	95.43	**95.57**

Table 7 Experimental Results on edited datasets: Preprocessing Cost (millions of distance computations)

Dataset	ENN	CNN	IB2	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC	dRHC
LR	127.99	112.20	18.35	300.51	55.13	94.76	127.84	138.41	178.45	31.05	**15.15**
MGT	115.76	68.61	8.48	318.82	11.44	10.15	11.28	12.42	21.75	**2.83**	6.18
PD	38.65	9.25	1.51	85.16	6.73	17.57	27.65	32.33	33.74	2.83	**1.25**
LS	13.25	6.49	0.99	30.64	2.86	4.83	6.79	9.97	11.82	1.73	**0.72**
SH	1076.46	26.02	6.35	15652.75	107.47	52.46	176.21	189.71	213.61	22.41	**6.05**
TXR	9.68	3.90	0.72	27.04	3.35	10.33	9.60	11.10	15.78	3.00	**0.57**
PH	9.35	5.57	0.86	15.67	0.68	1.04	1.89	2.18	3.15	**0.47**	0.73
BN	8.99	2.50	0.435	14.50	1.39	1.43	2.10	2.28	2.96	0.53	**0.434**
TN	17.52	12.50	1.41	34.20	1.81	3.13	4.02	6.38	9.56	1.36	**0.34**
MN2	0.06	0.03	0.005	0.12	0.01	0.06	0.07	0.12	0.13	0.007	**0.004**
Avg	141.77	24.71	3.91	1647.94	19.09	19.58	36.75	40.49	49.10	6.62	**3.14**

Table 8 Experimental results on the original KDD dataset

Criterion	IB2	RHC	dRHC
Accuracy (%):	**99.87**	99.84	99.82
Reduction Rate (%):	**99.78**	99.77	99.76
Preprocessing Cost (M):	**209.08**	291.91	223.97

the testing portions of the original KDD dataset contain more items, higher accuracies are achieved.

Another issue that we wanted to explore is how the size of data segment influences the performance of dRHC. Therefore, we ran experiments by adopting different segment sizes. The corresponding results are presented in Table 9. Considering the measurements, we conclude that segment size is rather irrelevant during dRHC execution. None of the comparison criteria is substantially improved by increasing or decreasing the segment size.

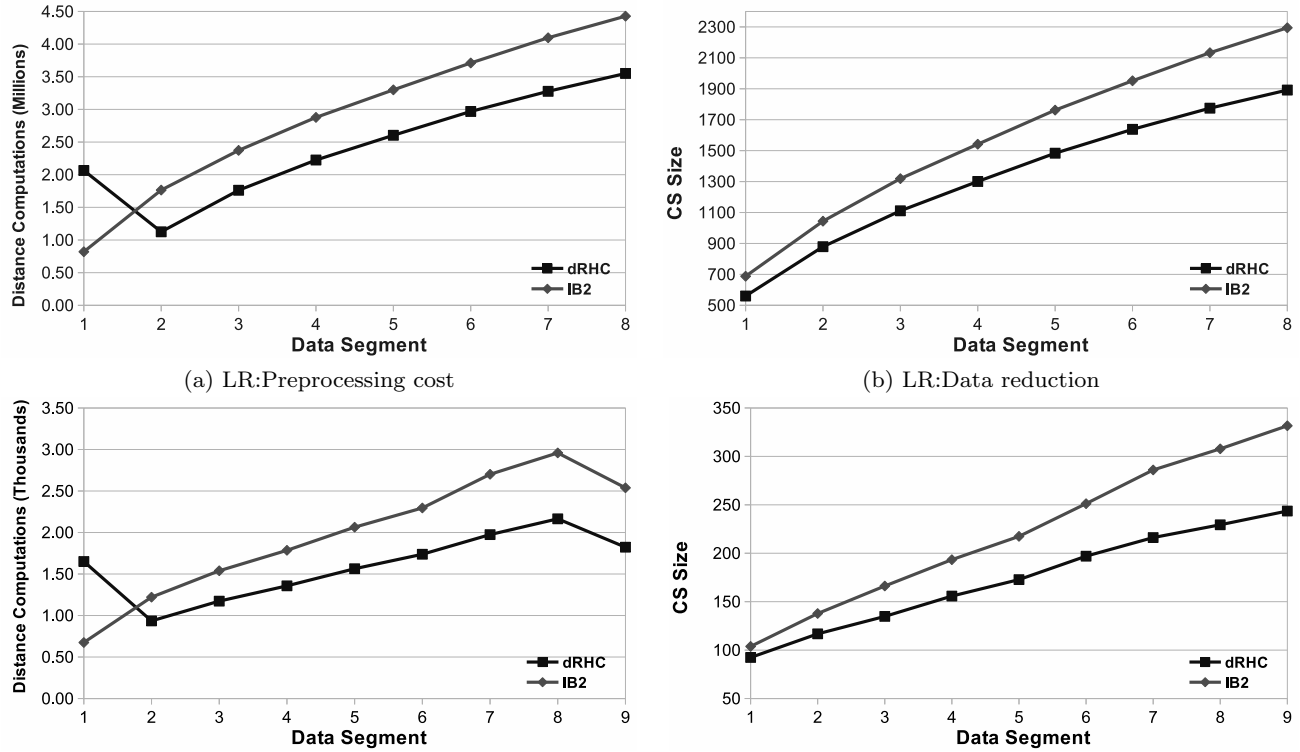
For dRHC and IB2, which are dynamic algorithms, we studied how the sizes of the condensing set increase over time and estimated the cost needed for the preprocessing of each data segment. Indicatively, Figure 5 presents these measurements for the LR and PD datasets in their non-edited form. The measurements are similar for all datasets, so we do not include figures for the other datasets. Let's recall that IB2 examines each individual item and decides whether to put it in the condensing set or not. On the other hand, dRHC considers data in data segments. Regarding the diagram for preprocessing cost measurements, it reports a preprocessing cost value for each *CS update* phase of dRHC and for each pass of t items for IB2, where t is equal to the size of the data segment used by dRHC.

Both measurements increase over time because the size of the available condensing set increases, and thus, more distances need to be computed over time. The last PD data segment contains 874 items instead of 1000. Therefore, lower preprocessing cost is needed for that segment (see Figure 5(c)). Concerning dRHC, as we expected, the *initial CS construction* phase is more time-consuming than the following *CS update* phases (see discussion near the end of Subsection 3.2). Considering Figure 5 as well as the measurements in Tables 2–7, one can conclude that dRHC achieves better performance than IB2 in terms of all comparison criteria.

4.4 Non parametric statistical test

We complement the section of the performance evaluation providing the results of a non-parametric statistical test of significance [40]. In particular, we used the Wilcoxon signed ranks test [15] in order to validate the experimental results presented in Tables 2–7. The test compares the DRTs in pairs taking into consideration their performance in each dataset. RHC and dRHC were compared to each other and to each one of the comparison DRTs.

We ran the test four times. Once for each comparison criterion (classification accuracy (ACC), reduction rate (RR), preprocessing cost (PC)) and once for the overall classification performance. By following the idea presented in [20,19], we computed the measurements of the overall classification performance by averaging the measurements of the three comparison criteria. Therefore, we considered the three criteria as having the same significance. Certainly, the computation of the overall classification performance implies that the measurements of the three criteria are in the same range. Therefore, we normalized the measurements to the range $[0, 1]$. Suppose that a dataset stores n items and an attribute a must be normalized to the range $[0, 1]$. The normalized attribute of i-th item, $i = 1, \ldots, n$



(a) LR:Preprocessing cost

(b) LR:Data reduction

(c) PD:Preprocessing cost

(d) PD:Data reduction

Fig. 5 Processing per data segment

is estimated as follows:

$$norm(a_i) = \frac{a_i - E_{min}}{E_{max} - E_{min}}$$

where E_{min} and E_{max} are the minimum and maximum measurements for a, respectively. Since low preprocessing cost is desirable, we used $1 - norm(PC)$ instead of $norm(PC)$.

Also, we ran all the tests twice, both on non-edited and edited data. Notice that we do not include the measurements obtained by the experimentation on the original form of the KDD dataset.

Tables 10 and 11 present the results of the Wilcoxon test. The columns labeled with "w/l/t" show the number of wins, losses and ties respectively (e.g., in Table 10, in the overall performance of RHC vs CNN case, "12/2/0" means that RHC was better than CNN 12 times and worse 2 times). The Wilcoxon value (columns labeled with "Wilc.") depicts how significant the difference of the corresponding methods is. If it is lower than 0.05, one can claim that the difference between the two methods is statistically significant.

This is true almost in all comparison pair in terms of overall classification performance. Therefore, the results of the test confirm that RHC and dRHC perform better than the other DRTs. It is worth mentioning that the results of the statistical tests presented in Table 10 reveal that the difference between dRHC and RHC is statistically significant. Thus, the test confirms that dRHC performs slightly better than RHC. In terms of reduction rate, the tests confirm that RHC and dRHC perform better than the other DRTs. Although IB2 has more wins than RHC in terms of preprocessing cost, the difference is not statistically significant ($Wilc. = 0.397$). In all other cases, RHC and dRHC perform better than the other DRTs in terms of preprocessing cost. In terms of accuracy, RHC is statistically better than PSC and dRHC is statistically better than IB2 and PSC. Although RHC has more wins than IB2 (8/5/1 in Table 10 and 6/4/0 in Table 11), the difference is not statistically significant (note that we have adopted a very strict threshold, i.e., $Wilc. = 0.05$, for the Wilcoxon significance level). In addition, the results in Table 10 confirm that RSP3 leads to the most accurate classification. Last but not least, the tests show that RHC and dRHC are as accurate as CNN-rule.

5 Conclusions

Data reduction is a crucial data mining task especially when using the k-NN classifier on large datasets. We

Table 10 Results of the Wilcoxon signed ranks test on the measurements obtained from non-edited data

Methods	ACC		RR		PC		Overall	
	w/l/t	Wilc.	w/l/t	Wilc.	w/l/t	Wilc.	w/l/t	Wilc.
RHC vs CNN	2/12/0	**0.009**	14/0/0	**0.001**	14/0/0	**0.001**	12/2/0	**0.005**
RHC vs IB2	8/5/1	0.311	10/4/0	**0.030**	5/9/0	0.397	10/4/0	**0.022**
RHC vs RSP3	1/13/0	**0.009**	14/0/0	**0.001**	14/0/0	**0.001**	14/0/0	**0.001**
RHC vs PSC (j=2)	13/1/0	**0.002**	10/4/0	0.245	12/2/0	**0.011**	13/1/0	**0.002**
RHC vs PSC (j=4)	12/2/0	**0.002**	10/4/0	0.245	14/0/0	**0.001**	13/1/0	**0.001**
RHC vs PSC (j=6)	13/1/0	**0.004**	9/5/0	0.221	14/0/0	**0.001**	11/3/0	**0.005**
RHC vs PSC (j=8)	13/1/0	**0.002**	10/4/0	0.109	14/0/0	**0.001**	13/1/0	**0.002**
RHC vs PSC (j=10)	14/0/0	**0.001**	11/3/0	0.074	14/0/0	**0.001**	13/1/0	**0.002**
dRHC vs CNN	5/9/0	0.363	14/0/0	**0.001**	14/0/0	**0.001**	14/0/0	**0.001**
dRHC vs IB2	9/5/0	**0.026**	12/2/0	**0.002**	11/3/0	**0.041**	11/3/0	**0.005**
dRHC vs RSP3	2/12/0	**0.026**	14/0/0	**0.001**	14/0/0	**0.001**	14/0/0	**0.001**
dRHC vs PSC (j=2)	13/1/0	**0.001**	10/4/0	0.124	12/2/0	**0.019**	13/1/0	**0.001**
dRHC vs PSC (j=4)	14/0/0	**0.001**	11/3/0	0.064	11/3/0	**0.026**	14/0/0	**0.001**
dRHC vs PSC (j=6)	13/1/0	**0.001**	11/3/0	**0.041**	13/1/0	**0.004**	12/2/0	**0.002**
dRHC vs PSC (j=8)	14/0/0	**0.001**	12/2/0	**0.030**	14/0/0	**0.001**	13/1/0	**0.001**
dRHC vs PSC (j=10)	14/0/0	**0.001**	12/2/0	**0.026**	14/0/0	**0.001**	13/1/0	**0.001**
dRHC vs RHC	10/4/0	**0.048**	11/3/0	0.056	11/3/0	**0.109**	13/1/0	**0.006**

Table 11 Results of the Wilcoxon signed ranks test on the measurements obtained from edited data

Methods	ACC		RR		PC		Overall	
	w/l/t	Wilc.	w/l/t	Wilc.	w/l/t	Wilc.	w/l/t	Wilc.
RHC vs CNN	5/5/0	0.959	10/0/0	**0.005**	10/0/0	**0.005**	7/3/0	0.093
RHC vs IB2	6/4/0	0.114	9/1/0	**0.013**	3/7/0	0.169	7/3/0	0.074
RHC vs RSP3	1/9/0	0.074	10/0/0	**0.005**	10/0/0	**0.005**	10/0/0	**0.005**
RHC vs PSC (j=2)	10/0/0	**0.005**	8/1/1	**0.011**	10/2/0	**0.005**	10/0/0	**0.005**
RHC vs PSC (j=4)	10/0/0	**0.005**	9/1/0	**0.007**	10/0/0	**0.005**	10/0/0	**0.005**
RHC vs PSC (j=6)	10/0/0	**0.005**	9/1/0	**0.007**	10/0/0	**0.005**	10/0/0	**0.005**
RHC vs PSC (j=8)	10/0/0	**0.005**	9/1/0	**0.009**	10/0/0	**0.005**	10/0/0	**0.005**
RHC vs PSC (j=10)	10/0/0	**0.005**	9/1/0	**0.009**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs CNN	6/4/0	0.386	10/0/0	**0.005**	10/0/0	**0.005**	8/2/0	**0.017**
dRHC vs IB2	8/2/0	**0.037**	9/0/1	**0.008**	9/0/1	**0.008**	8/2/0	**0.017**
dRHC vs RSP3	3/7/0	0.333	10/0/0	**0.005**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs PSC (j=2)	10/0/0	**0.005**	9/1/0	**0.009**	9/1/0	**0.009**	10/0/0	**0.005**
dRHC vs PSC (j=4)	10/0/0	**0.005**	9/1/0	**0.009**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs PSC (j=6)	10/1/0	**0.005**	8/2/0	**0.013**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs PSC (j=8)	10/0/0	**0.005**	8/2/0	**0.013**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs PSC (j=10)	10/0/0	**0.005**	8/2/0	**0.013**	10/0/0	**0.005**	10/0/0	**0.005**
dRHC vs RHC	8/2/0	0.114	6/4/0	0.241	8/2/0	0.093	8/2/0	0.059

presented RHC, a fast non-parametric algorithm for data reduction. It uses k-Means to recursively cluster the training dataset into homogeneous clusters. The condensing set consists of the centroids of the final clusters. RHC combines the advantages of RSP3 and PSC algorithms while avoiding their drawbacks. Furthermore, we presented dRHC, a dynamic version of RHC, which retains all good properties of RHC and, in addition, supports frequent updates of the condensing set. Therefore, it can deal with datasets that cannot fit into the main memory and/or streaming training data.

Experimental results, obtained by using edited and non-edited versions of fourteen datasets, showed that both algorithms had low preprocessing cost and achieved the highest reduction rates without significant loss of accuracy. We claim that these properties render RHC

Table 9 Using different segment (Memory/Buffer) sizes during dRHC execution on non-edited datasets (Acc (%)), Reduction Rate (RR (%)) and Preprocessing Cost (PC (M))

Dataset		Segment size			
		500	**1000**	**2000**	**4000**
LR	Acc:	94.20	93.40	93.93	94.12
	RR:	88.06	88.10	88.18	88.37
	PC:	19.75	19.48	19.57	19.99
MGT	Acc:	72.52	72.68	72.96	72.51
	RR:	72.92	73.63	74.59	75.29
	PC:	30.28	28.45	26.12	22.77
PD	Acc:	98.42	98.49	98.41	98.40
	RR:	97.29	97.23	97.22	96.95
	PC:	1.38	1.44	1.63	2.07
LS	Acc:	88.92	87.69	88.73	89.18
	RR:	88.24	88.94	89.48	89.41
	PC:	1.57	1.43	1.52	1.64
SH	Acc:	99.74	99.68	99.73	99.70
	RR:	99.46	99.48	99.45	99.49
	PC:	8.12	7.70	8.38	7.48
TXR	Acc:	97.31	97.56	97.82	97.29
	RR:	95.25	95.06	94.90	94.56
	PC:	0.67	0.75	1.03	2.59
PH	Acc:	85.38	84.97	85.47	85.64
	RR:	82.34	82.28	81.88	80.99
	PC:	1.64	1.55	1.34	0.82
KDD	Acc:	99.39	99.42	99.42	99.38
	RR:	99.22	99.22	99.24	99.25
	PC:	57.59	57.32	54.58	51.62
BL	Acc:	68.8	-	-	-
	RR:	78.04	-	-	-
	PC:	0.05	-	-	-
BN	Acc:	83.15	83.64	82.42	83.19
	RR:	82.63	81.98	81.63	79.86
	PC:	1.51	1.47	1.22	0.73
YS	Acc:	47.10	48.99	-	-
	RR:	49.85	49.23	-	-
	PC:	0.32	0.49	-	-
TN	Acc:	93.34	92.96	92.69	92.03
	RR:	95.19	95.75	96.37	96.31
	PC:	0.72	0.64	0.65	1.06

and dRHC appropriate for environments where fast classification and/or low preprocessing cost are critical. Moreover, we demonstrated that dRHC is independent of the size of data segment (memory/buffer).

We plan to keep on examining the issue of data reduction for efficient k-NN classification. Our future work includes investigation and development of non-parametric data reduction techniques that can deal with large and fast data streams with concept drift. Moreover, we are going to devise a RHC-based algorithm that integrates an editing mechanism for noise removal into the data reduction process.

Acknowledgements We are grateful to the anonymous reviewers for their valuable comments on the original form of the paper.

References

1. 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
2. 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
3. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing **17**(2-3), 255–287 (2011)
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. Chang, C.L.: Finding prototypes for nearest neighbor classifiers. IEEE Trans. Comput. **23**(11), 1179–1184 (1974). DOI 10.1109/T-C.1974.223827. URL http://dx.doi.org/10.1109/T-C.1974.223827
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. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. **13**(1), 21–27 (2006). DOI 10.1109/TIT.1967.1053964. URL http://dx.doi.org/10.1109/TIT.1967.1053964
12. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)

13. Dasarathy, B.V., Snchez, J.S., Townsend, S.: Nearest neighbour editing and condensing toolssynergy exploitation. Pattern Analysis & Applications **3**(1), 19–30 (2000). DOI 10.1007/s100440050003. URL http://dx.doi.org/10.1007/s100440050003
14. Datta, P., Kibler, D.F.: Learning symbolic prototypes. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97, pp. 75–82. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)
15. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006). URL http://dl.acm.org/citation.cfm?id=1248547.1248548
16. Devi, V.S., Murty, M.N.: An incremental prototype set building technique. Pattern Recognition **35**(2), 505–513 (2002)
17. Devijver, P.A., Kittler, J.: On the edited nearest neighbor rule. In: Proceedings of the Fifth International Conference on Pattern Recognition. The Institute of Electrical and Electronics Engineers (1980)
18. Fayed, H.A., Hashem, S.R., Atiya, A.F.: Self-generating prototypes for pattern classification. Pattern Recogn. **40**(5), 1498–1509 (2007). DOI 10.1016/j.patcog.2006.10.018. URL http://dx.doi.org/10.1016/j.patcog.2006.10.018
19. García, S., Cano, J.R., Herrera, F.: A memetic algorithm for evolutionary prototype selection: A scaling up approach. Pattern Recogn. **41**(8), 2693–2709 (2008). DOI 10.1016/j.patcog.2008.02.006. URL http://dx.doi.org/10.1016/j.patcog.2008.02.006
20. 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
21. García-Borroto, M., Villuendas-Rey, Y., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F.: Using maximum similarity graphs to edit nearest neighbor classifiers. In: Proceedings of the 14th Iberoamerican Conference on Pattern Recognition: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, CIARP '09, pp. 489–496. Springer-Verlag, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-10268-4_57. URL http://dx.doi.org/10.1007/978-3-642-10268-4_57
22. Gates, G.W.: The reduced nearest neighbor rule. ieee transactions on information theory. IEEE Transactions on Information Theory **18**(3), 431–433 (1972)
23. 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)
24. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science (2011)
25. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory **14**(3), 515–516 (1968)
26. James, M.: Classification algorithms. Wiley-Interscience, New York, NY, USA (1985)
27. 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)
28. Lozano, M.: Data Reduction Techniques in Classification processes (Phd Thesis). Universitat Jaume I (2007)
29. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. of 5th Berkeley Symp. on Math. Statistics and Probability, pp. 281– 298. Berkeley, CA : University of California Press (1967)
30. Mollineda, R., Ferri, F., Vidal, E.: An efficient prototype merging strategy for the condensed 1-nn rule through class-conditional hierarchical clustering. Pattern Recognition **35**(12), 2771 – 2782 (2002)
31. 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
32. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: A new fast prototype selection method based on clustering. Pattern Anal. Appl. **13**(2), 131–141 (2010)
33. Olvera-López, J.A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A.: Mixed data object selection based on clustering and border objects. In: Proceedings of the Congress on pattern recognition 12th Iberoamerican conference on Progress in pattern recognition, image analysis and applications, CIARP'07, pp. 674–683. Springer-Verlag, Berlin, Heidelberg (2007). URL http://dl.acm.org/citation.cfm?id=1782914.1782996
34. Olvera-Lpez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: Object selection based on clustering and border objects. In: M. Kurzynski, E. Puchala, M. Wozniak, A. Zolnierek (eds.) Computer Recognition Systems 2, *Advances in Soft Computing*, vol. 45, pp. 27–34. Springer (2008)
35. 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
36. Ougiaroglou, S., Evangelidis, G.: Fast and accurate k-nearest neighbor classification using prototype selection by clustering. In: 16th Panhellenic Conference on Informatics (PCI), 2012, pp. 168–173 (2012)
37. 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)
38. Samet, H.: Foundations of multidimensional and metric data structures. The Morgan Kaufmann series in computer graphics. Elsevier/Morgan Kaufmann (2006)
39. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition **37**(7), 1561–1564 (2004)
40. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. A Chapman & Hall book. Chapman & Hall/CRC (2011)
41. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics **6**, 448–452 (1976)
42. 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
43. Toussaint, G.: Proximity graphs for nearest neighbor decision rules: Recent progress. In: 34th Symposium on the INTERFACE, pp. 17–20 (2002)
44. 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

45. 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)
46. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE trans. on systems, man, and cybernetics **2**(3), 408–421 (1972)
47. 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
48. Wu, J.: Advances in K-means Clustering: A Data Mining Thinking. Springer Publishing Company, Incorporated (2012)
49. 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