Ann Math Artif Intell (2016) 76:327–349
DOI 10.1007/s10472-015-9472-8

Efficient editing and data abstraction by finding homogeneous clusters

Stefanos Ougiaroglou[1] · **Georgios Evangelidis**[1]

Published online: 15 August 2015

Abstract The efficiency of the k-Nearest Neighbour classifier depends on the size of the training set as well as the level of noise in it. Large datasets with high level of noise lead to less accurate classifiers with high computational cost and storage requirements. The goal of editing is to improve accuracy by improving the quality of the training datasets. To obtain such datasets, editing removes noise and mislabeled data as well as smooths the decision boundaries between the discrete classes. On the other hand, prototype abstraction aims to reduce the computational cost and the storage requirements of classifiers by condensing the training data. This paper proposes an editing algorithm called Editing through Homogeneous Clusters (EHC). Then, it extends the idea by introducing a prototype abstraction algorithm that integrate the EHC mechanism and is capable of creating a small noise-free representative set of the initial training data. This algorithm is called Editing and Reduction through Homogeneous Clusters (ERHC). Both are based on a fast and parameter free iterative execution of k-means clustering that forms homogeneous clusters. Both consider as noise and remove clusters consisting of a single item. In addition, ERHC summarizes the items of the remaining clusters by storing the mean item for each one in the representative set. EHC and ERHC are tested on several datasets. The results show that both run very fast and achieve high accuracy. In addition, ERHC achieves high reduction rates.

Keywords k-NN classification · Clustering · Data reduction · Data abstraction · Editing · Noise · Prototypes

Mathematics Subject Classification (2010) 68T10 · 62H30

✉ Stefanos Ougiaroglou
stoug@uom.gr

Georgios Evangelidis
gevan@uom.gr

[1] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia str., 54006, Thessaloniki, Greece

1 Introduction

Classification is a traditional data mining problem that has attracted the interest of many researchers in the past decades [15]. Classification algorithms (or classifiers) attempt to assign unclassified items to a class from a set of predefined classes. Classifiers can be divided into eager and instance-based (or lazy) classifiers. Contrary to eager classifiers, lazy classifiers do not build any classification model that is then used to classify new items. Instead, they use the training set as the classification model.

The k-Nearest Neighbor (k-NN) classifier [5, 6] is the reference lazy classification algorithm. It is a simple and easy to implement classifier that can be exploited in many application domains and easily integrated in many systems. Moreover, 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 [5].

When a new item needs to be classified, the k-NN classifier scans the training data. In particular, it classifies an item x by searching in the available training set and retrieving the k nearest items (neighbours) to x according to a distance metric. Then, x is classified to the most common class among the classes of the k nearest neighbours. This class is often called major class and is determined by a nearest neighbours voting procedure. Possible ties during voting (more than one classes are voted to be major classes) are resolved by choosing the class of the nearest neighbour or randomly.

The k-NN classifier is considered to be an effective classifier. However, it has some weaknesses that render its use inefficient. The first one is that it is noise-sensitive. More specifically, accuracy highly depends on the quality of the training set. Mislabeled items, noise, outliers and overlaps between regions of discrete classes, negatively affect its accuracy. This drawback is partially remedied by examining a larger neighbourhood, i.e., using a high k value. However, this assumes that noise is uniformly distributed in the training set and requires k to be tuned via a trial and error procedure. Furthermore, high k values do not clearly define the boundaries between distinct classes. Another weak point is the high computational cost involved. The algorithm must compute all distances between each new, unclassified item and all training items. Although nowadays systems are equipped with powerful processors, the distance computations are time-consuming and, in cases of time-constraint environments, can be prohibitive. Finally, the high storage requirements needed to store the training set is also weak point of the k-NN classifier. Contrary to eager classifiers, which can remove the training data after the construction of the classification model, the k-NN classifier needs the training data to be always available.

These weaknesses constitute an active research problem and have attracted the interest of the data mining community in the past decades. Several Data Reduction Techniques (DRTs) can effectively cope with the aforementioned weaknesses. They are distinguished into prototype selection algorithms [11] and prototype abstraction [36] algorithms. The former algorithms select representative items (or prototypes) from the initial training set, whereas the later ones create prototypes by summarizing similar items in the training set. Prototype selection algorithms are also distinguished into two categories. They can be either condensing or editing algorithms. Condensing and prototype abstraction algorithms aim to create a small representative set of the training set, called the condensing set. This has the benefits of low storage requirements and computational cost without sacrificing accuracy. In contrast, editing algorithms have as goal to improve accuracy rather than to achieve high reduction rates. Thus, they try to create an edited set that does not contain region overlaps

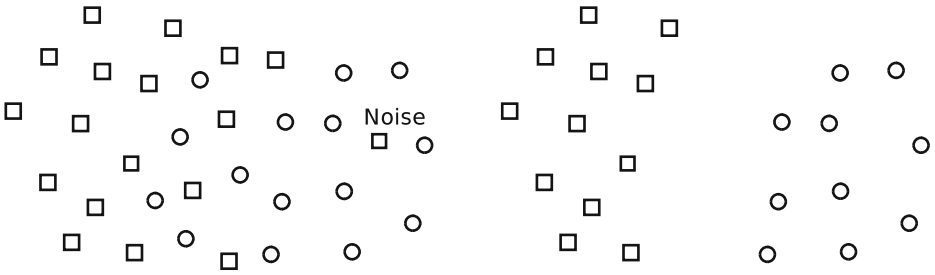


Fig. 1 Smoothing decision boundaries and removing noise

between classes. To achieve this, they attempt to remove outliers, noise and mislabelled data and smooth the boundaries between classes (see Fig. 1).

High levels of noise in the training set prevent many condensing or prototype abstraction algorithms from achieving high reduction rates. In effect, the higher the level of noise, the lower the reduction rates achieved. Therefore, effective application of a condensing or prototype abstraction algorithm usually implies the application of editing beforehand [7, 19]. Hence, an editing algorithm should be used on a training set that contains noise in order to either improve accuracy or make more effective the application of condensing and prototype abstraction algorithms. It is worth mentioning that some condensing approaches combine the idea of editing. They are called hybrid algorithms (see [11, 36]).

Figure 2 summarizes the k-NN classification process through data reduction. The whole process includes two phases, preprocessing and classification. Certainty, the preprocessing phase is optional. Generally, there are four possible combinations of preprocessing: (i) no-preprocessing, (ii) only editing, (iii) only condensing, and (iv) both editing and condensing. If the training set does not contain noise and misleading data and its size is small, no preprocessing is required. When the size of the training set is small, but it contains noise, only an editing algorithm should be executed during preprocessing. On the other hand, in cases of large and noise-free training sets, data reduction without editing should be executed. Finally, in cases of large training data with noise, both kinds of preprocessing algorithms must be ran.

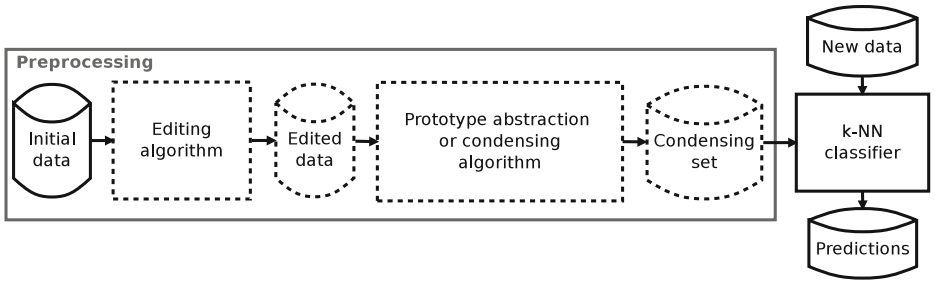


Fig. 2 Data reduction preprocessing

Here, we should clarify that the concept of data reduction is wider [12]. It does not only concern prototype selection and abstraction algorithms. Many other preprocessing algorithms have been characterized as DRTs. They include feature selection and generation algorithms as well as space transformation techniques. However, in this paper, we consider DRTs from the point of view of prototype selection and abstraction. Moreover, we should note that prototype selection and abstraction algorithms can be hybridized with other type of DRTs [8, 37] or other learning algorithms (e.g., [14]).

Although editing algorithms contribute in obtaining high quality training data, they constitute an extra costly preprocessing step. Moreover, to the best of our knowledge, all the well-known editing algorithms are parametric [9, 35, 39]. Users have to define at least one parameter value via expensive trial-and-error procedures (tuning). These observations are behind the motivation of our work. In [27], we presented an effective parameterless editing algorithm called EHC. Furthermore, in [25, 28], we proposed RHC, an efficient parameterless prototype abstraction algorithm. Both are very fast and are based on a similar k-means clustering [20, 41] procedure that finds homogeneous clusters, i.e., clusters with items of a specific class. In this paper, we extend our previous work and develop and evaluate an effective fast, parameterless prototype abstraction algorithm which combines the ideas of EHC editing and RHC data abstraction. It is called Editing and Reduction though Homogeneous Clusters (ERHC) and is a descendant of our RHC and EHC algorithms.

The rest of the paper is organized as follows: Section 2 reviews the most well-known editing algorithms. Section 3 presents the family of the three algorithms based on homogeneous clusters, i.e., EHC, RHC and ERHC. Performance evaluation experiments and the results of a non-parametric test are presented in Section 4. Finally, Section 5 concludes the paper.

2 Editing algorithms

2.1 The edited nearest neighbour (ENN) rule

The reference editing algorithm is the Wilson's Edited Nearest Neighbor (ENN) rule [39]. It constitutes the base of all other editing algorithms. ENN-rule is very simple. Algorithm 1 lists the pseudo-code of ENN-rule. Initially, the edited set (ES) is set to be equal to the training set (TS) (line 1). For each item x of TS, the algorithm scans TS and retrieves its k nearest neighbors (line 3). If x is misclassified by the majority vote of the retrieved nearest neighbours, it is removed from ES (lines 4–7). ENN-rule considers wrongly classified items to be noise or close-border items and, thus, they must be removed. Note that, in each algorithm iteration, ENN-rule searches for nearest neighbours in the original training set and not in the "under construction" edited set.

Obviously, the cost of editing depends on the size of the training set. In cases of large datasets, ENN-rule is a time-consuming algorithm since it must compute all distances between the training items. More specifically, $\frac{N\times(N-1)}{2}$ distances must be computed, where N is the number of training items.

A crucial issue that should be addressed is the determination of the value of k that defines the size of the examined neighbourhood. [13, 21, 40] consider $k = 3$ to be a typical value. This is adopted in many papers (e.g. [30]), whereas, other papers use $k = 3$ and additional k values (e.g., [17, 34]). In some cases, researchers determine the value of k that achieves the best performance through trial-and-error procedures (e.g., [38]). In [39], the impact of

Algorithm 1 ENN-rule

Input: TS, k
Output: ES

1: $ES \leftarrow TS$
2: **for** each $x \in TS$ **do**
3: $NNs \leftarrow$ find the k nearest to x neighbors in $TS - \{x\}$
4: $majorClass \leftarrow$ find the most common class of NNs
5: **if** $x_{class} \neq majorClass$ **then**
6: $ES \leftarrow ES - \{x\}$
7: **end if**
8: **end for**
9: **return** ES

k is discussed in detail. Furthermore, in [17], a large number of k values are experimentally evaluated. It turns out that the best value of k depends on the dataset at hand and should be determined by considering the distribution of items in the multidimensional space. Even the best value of k may not be optimal and it may remove items that are not noise (see [13]) or keep items that are noise. This happens because ENN-rule uses a unique k value for the entire training set. Different k values may be optimal for different regions in space.

2.2 All *k*-NN

All-kNN [35] is a popular variation of ENN-rule. It iteratively executes ENN-rule with different k values (see Algorithm 2. All-kNN adopts $kmax$ as an upper limit for the value of k. Initially, the edited set (ES) is set to be the whole training set (TS) (line 1). For each item x in TS (line 2), All-kNN applies the k-NN classifier on the items of TS (lines 6–7), initially with $k = 1$, and tries to remove x from ES in a way similar to ENN-rule. If x is misclassified, it is removed and the procedure continues with the next item (lines 8–10). Otherwise, k is incremented by one (line 12) and the algorithm retries to remove x. If the item is not removed after $kmax$ iterations (line 5), x remains in the final ES and All-kNN continues with the next item.

Since All-kNN uses more than one values for k, it removes more items than ENN-rule. Although All-kNN is an iterative version of ENN-rule, an efficient implementation of it does not re-compute the same distances again and again. Therefore, All-kNN computes as many distances as ENN-rule and is parametric, too. The value of $kmax$ must be defined by the user. This usually implies tuning via trial-and-error. Garcia-Borroto et al. consider $kmax = 7$ or $kmax = 9$ to be appropriate values [13].

2.3 Multiedit

Multiedit [9] is another well-known editing approach. Its pseudo-code is presented in Algorithm 3. Initially, the edited set (ES) is set to be equal to the training set (TS) (line 1). Then, TS is divided into n random subsets, $s_1, s_2, \ldots, s_n$ (line 5). The algorithm continues by applying ENN-rule over each item $x \in s_i$ (line 7) of each subset s_i (line 6), but searching for the single nearest neighbor (1-NN) in the modulo n following subset, i.e., $s_{(i+1) \bmod n}$ (line 8). The misclassified items are removed from ES (line 10). If at least one item is removed,

Algorithm 2 All-kNN

Input: $TS, kmax$
Output: ES

```
1:  ES ← TS
2:  for each x ∈ TS do
3:    k ← 1
4:    flag ← FALSE
5:      while (k ≤ kmax) and (flag == FALSE) do
6:        NNs ← find the k nearest to x neighbors in TS − {x}
7:        majorClass ← find the most common class in NNs
8:          if x_class ≠ majorClass then
9:            ES ← ES − {x}
10:           flag ← TRUE
11:         end if
12:         k ← k + 1
13:     end while
14: end for
15: return ES
```

TS is set to be ES (line 20) and the whole process is repeated. Multiedit continues until the last R iterations produce no editing (lines 11,15–16,21).

Here, parameter k is not used since multiedit utilizes the 1-NN classifier. However, parameters n and R influence the resulting edited set. Parameter $n \geq 3$ defines the number of subsets. In many papers (e.g., [13, 34]), $n = 3$ is either adopted or proposed. Parameter R defines the number of non-editing iterations. In [13], $R = 2$ is suggested as an appropriate value. Nevertheless, the best values for these parameters can not be determined without tuning through a trial-end-error procedure.

Multiedit usually achieves higher reduction rates than ENN-rule. It can successfully remove noise, outliers and close-border items. However, it may also remove items that are not noise. If items of two or more classes are close to each other, multiedit may eliminate entire classes [13]. Another drawback of multiedit is that it is based on a random formation of subsets, i.e., repeated applications may build a completely different edited set from the same training data.

Multedit is usually more time-consuming than ENN-rule. However, it may compute even fewer than $\frac{N \times (N-1)}{2}$ distances. An implementation of multiedit that does not compute a distance more than once should have the distances that have been already computed available until the end of the execution. Therefore, such an implementation requires more memory. In case of a simple implementation where each distance may be computed more than once, the computational cost of the algorithm highly depends on the value of R.

2.4 Other editing algorithms

Subsections 2.1, 2.2 and 2.3 presented in detail three state-of-the-art editing algorithms that we use for comparison purposes in our experimental study in Section 4. Many more editing approaches have been proposed in the literature.

EENProb and ENNth [38] are extensions of ENN-rule. Both retrieve the k nearest neighbors, and then perform editing based on probability estimations. Repeated ENN (RENN)

rule [35] is also a variation of ENN-rule. Actually, it is quite similar to All-kNN. RENN-rule applies ENN-rule in an iterative way until each item's majority of k nearest items have the same class. In [17], another simple variation of ENN-rule is proposed that places an item in the edited set, only if all its k nearest neighbors share the same class label with it (distance ties increase the value of k).

Sanchez et al. proposed two editing algorithms that are based on geometric information provided by proximity graphs [34]. They are also based on the concept of removal of misclassified items. To the best of our knowledge, they are the only parameterless editing algorithms. Nevertheless, the type of proximity graphs used influence the resulting edited set. In [34], two types of proximity graphs were used. Consequently, four editing approaches were obtained and evaluated. From this point of view, even these algorithms can be characterized as parametric methods.

k-NCN editing and its iterative version [30] are also based on ENN-rule. Particularly, they use the k nearest centroid neighbourhood classifier [33] instead of the k-NN classifier. Both are based on the following simple idea: the appropriate neighbourhood that should be examined for each item is defined by taking into consideration not only its nearest neighbours but also the symmetrical distribution of neighbours around it.

In [4, 30] a depuration algorithm is proposed for editing training data. In addition to removing some training items, the algorithm also changes the class labels of some items. To achieve this, it uses two input parameters (see [4] or [30] for details). [18] considers and evaluates editing approaches based on the depuration algorithm and proposes the Neural

Algorithm 3 Multiedit

Input: TS, n, R
Output: ES

$ES \leftarrow TS$
$r \leftarrow 0$
repeat
  $flag \leftarrow$ FALSE
  $S \leftarrow$ set of n random subsets, $s_1, s_2, \ldots, s_n$ of TS
  for each $s_i \in S$ **do**
    for each $x \in s_i$ **do**
      $nn \leftarrow$ find the nearest neighbor in $s_{(i+1)modn}$
      if $x_{class} \neq nn_{class}$ **then**
        $ES \leftarrow ES - \{x\}$
        $flag \leftarrow$ TRUE
      end if
    end for
  end for
  if $flag =$ FALSE
    $r \leftarrow r + 1$
  else
    $r \leftarrow 0$
  end if
  $TS \leftarrow ES$
until $r == R$ {until the last R iterations do not edit data}
return ES

Network Ensemble Editing (NNEE). This method is also parametric. NNEE trains a neural network ensemble that is then used to relabel some items. Last by not least, a recent paper [31] proposes the use of local support vector machines for noise reduction. Like the other methods, its performance depends on parameter tuning.

3 Algorithms based on homogeneous clusters

3.1 An algorithm for finding homogeneous clusters

We propose three algorithms that are based on a procedure that forms clusters containing items of a specific class only, i.e., they are homogeneous clusters. It is a quite simple procedure that utilizes k-means clustering. More specifically, initially, the whole training set is considered to be a non-homogeneous cluster. The algorithm for finding homogeneous clusters begins by computing the class-mean for each class by averaging the corresponding items of the cluster. Therefore, for a dataset with r classes, it estimates r class-means. The algorithm continues by executing k-means clustering adopting the r class-means as initial means. The result is the construction of r clusters. The above clustering procedure is applied recursively on the items of each non-homogeneous cluster and terminates when all clusters become homogeneous. Notice that using the class-means as initial means for k-means clustering, the number of clusters is automatically determined.

The algorithm is easy to implement. Algorithm 4 lists the pseudo-code of a possible implementation. It uses a queue structure Q to store clusters. Initially, Q holds the whole training set TS as an unprocessed cluster (lines 2–3). In each iteration, the head cluster C is dequeued from Q (line 5). If C is non-homogeneous (line 8), a class-mean for each class in C is computed and added to set R (lines 9–12). The later as well as C is the input to a k-means clustering call (line 13). The resulting clusters $Clusters$ are enqueued in Q (lines 14–16). The repeat-until loop (lines 4, 18) terminates when Q is empty, i.e., all clusters become homogeneous. The proposed algorithms, i.e., RHC, EHC and ERHC, differ to each other on how they treat the homogeneous clusters (line 7).

In effect, when the algorithm is executed over a noise-free dataset, it forms a small number of large clusters. On the other hand, if a dataset with high level of noise is used, a high number of small clusters are constructed. Concerning the computational cost, we can easily conclude that the algorithm is fast. It uses the fast k-means clustering algorithm that is also sped-up by considering as initial means the means of the classes that are present in each cluster. One expects that the resulting clusters are quickly consolidated and the cost is lower than when opting for random means initialization. It is worth mentioning that we used the full cluster consolidation for the stopping condition of k-Means clustering. The algorithm could have been even faster had we used another stopping condition. Certainly, the algorithm always builds the same clusters regardless of data ordering. Last but not least, the algorithm is quite simple and can be easily integrated in many existing data mining software tools. Since EHC, RHC and ERHC are based on this algorithm, they inherit all these desirable properties.

3.2 The editing through homogeneous clusters algorithm

As we already mentioned in Section 2, editing algorithms either extend ENN-rule or are based on the same idea. The proposed Editing through Homogeneous Clusters (EHC) algorithm follows a completely different, parameterless strategy in order to remove noise,

Algorithm 4 FindHomogeneousClusters

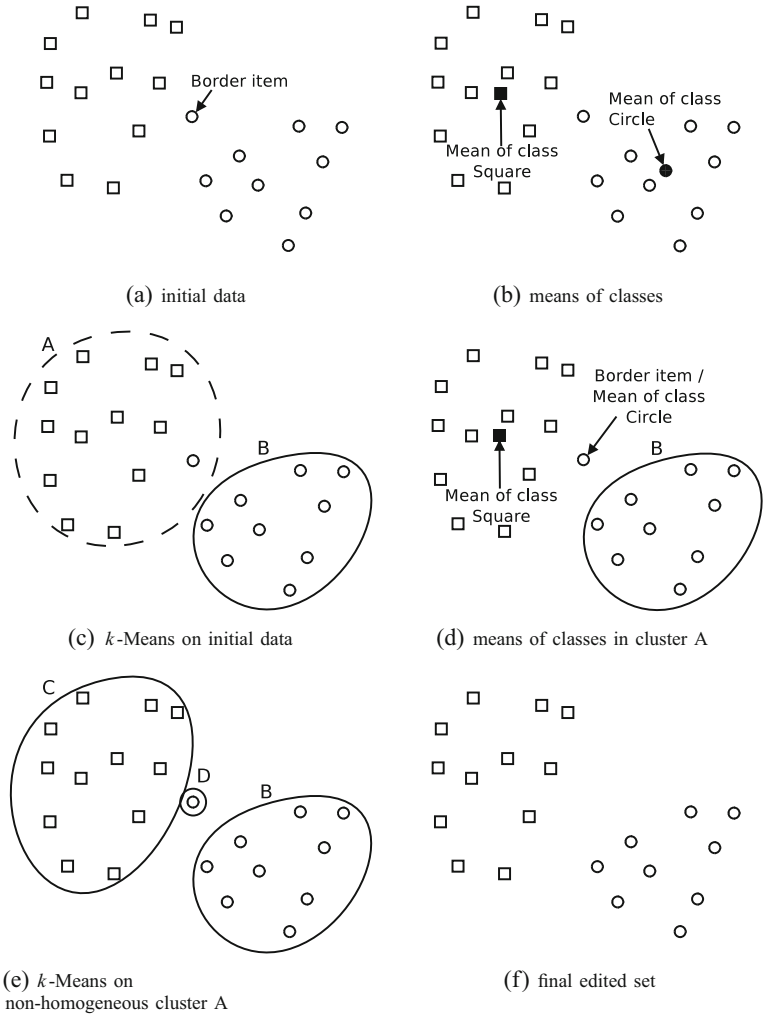
Input: TS
Output: RS

```
1: RS ← ∅
2: Q ← ∅
3: Enqueue(Q, TS)
4: repeat
5:   C ← Dequeue(Q)
6:   if C is homogeneous then
7:     RS ← DATA_REDUCTION(RS, C) {RHC or EHC or ERHC is called}
8:   else
9:     R ← ∅ {R is the set of class means}
10:    for each class M in C do
11:      R ← R ∪ mean_of(M)
12:    end for
13:    Clusters ← K-MEANS(C, R)
14:    for each cluster Cl ∈ Clusters do
15:      Enqueue(Q, Cl)
16:    end for
17:  end if
18: until IsEmpty(Q) {until all constructed clusters are homogeneous}
19: return RS
```

mislabelled and close-border data items. EHC uses the procedure for finding homogeneous clusters (see Algorithm 4). When a homogeneous cluster is identified, EHC counts the items in the cluster and removes it if it has only one item. We call the clusters that have only one item, one-item clusters. The idea behind EHC is quite simple: one-item-clusters are redundant. These items are probably outliers or lie in a region of a different class (noise) or lie close to a decision-boundaries region, and thus, they must be removed.

Two examples that demonstrate the operation of EHC are depicted in Figs. 3 and 4. More specifically, Fig. 3 demonstrates how EHC identifies and removes a close-border item, while Fig. 4 demonstrates how the algorithm removes an item that is noise. Note that non-homogeneous clusters are depicted with dashed borders. In particular, Fig. 3a presents a dataset with a border item that should be removed. Initially, EHC computes the class-means by averaging the items that belong to each class (Fig. 3b). Then, k-means is executed using the class-means as initial means and identifies two clusters (Fig. 3c): cluster A is non-homogeneous while cluster B is homogeneous. Since cluster B is homogeneous and has more than one item, it is ignored. Then, the class-means in cluster A are computed (Fig. 3d) and k-means is executed on the data of the particular cluster. The result is the construction of two homogeneous clusters (Fig. 3e). Since cluster D is a one-class cluster, it is removed (Fig. 3f). In an analogous way, EHC removes the item that represents noise in Fig. 4a. EHC identifies and removes outliers in a similar way. X

EHC may assign a typical data item (that is not noise or a close-border item) to an one-item cluster and remove it. For instance, suppose that a non-homogeneous cluster with two items is built. EHC will remove both items even when one of them belongs to the major class of the region. It is worth mentioning that contrary to all other editing methods, EHC does not compute distances between "real" items. It computes distances between items and



(a) initial data (b) means of classes

(c) k-Means on initial data (d) means of classes in cluster A

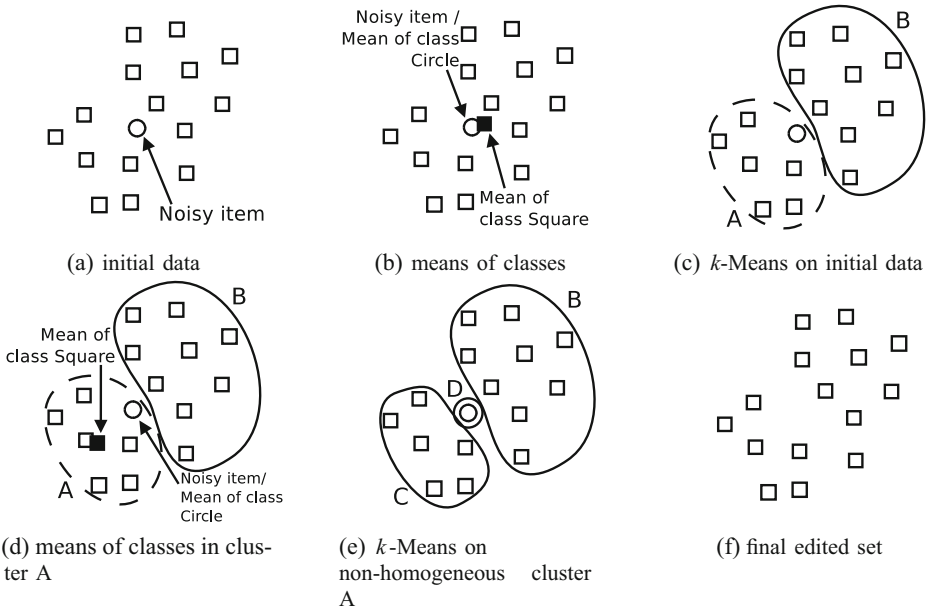
(e) k-Means on non-homogeneous cluster A (f) final edited set

Fig. 3 EHC: Smoothing decision boundaries

mean items. Moreover, contrary to ENN-rule and some of its variations that compute a fixed number of distances regardless the item distribution in the multidimensional space, the number of distances computed by EHC is difficult to predict in advance. It exclusively depends on the item distribution in the data space. The main advantages of EHC are that, contrary to all other editing approaches, it is very fast and is the only parameterless editing algorithm. Hence, costly and time-consuming trial-end-error procedures for parameter tuning are avoided.

3.3 The reduction through homogeneous clusters algorithm

Although the Reduction through Homogeneous Clusters (RHC) algorithm [25, 28] and EHC are based on the idea of forming homogeneous clusters (see Algorithm 4), each one aims to a



(a) initial data (b) means of classes (c) *k*-Means on initial data

(d) means of classes in cluster A (e) *k*-Means on non-homogeneous cluster A (f) final edited set

Fig. 4 EHC: Removing noise

different goal. RHC is a prototype abstraction algorithm while EHC is an editing algorithm. RHC, like other condensing and prototype abstraction algorithms [22, 23, 29], is based on the concept of homogeneity. When a homogeneous cluster is constructed, RHC computes its mean by averaging the items that have been assigned to it. This mean item is stored in the condensing set as prototype. Algorithmically, RHC differs from EHC in the following point: EHC stores in the edited set all training items that have not been assigned to one-item clusters, while RHC computes the mean item for each homogeneous cluster and stores it in the condensing set.

Considering RHC, we realize that it generates many prototypes for the close-border data areas and few prototypes for the non-close-border data areas. Hence, the more the classes in the data and the higher the level of noise, the more borders exist, and therefore, the more prototypes are generated. By adopting the class-means as initial means for *k*-means clustering, RHC quickly finds homogeneous clusters and also achieves high reduction rates. Contrary to many other condensing and prototype abstraction algorithms, RHC builds the same condensing set regardless the order of the training data.

In effect, RHC retains the advantages of PSC [22–24, 26] and RSP3 [29] and avoids their drawbacks. PSC is fast and parametric. RSP3 is parameterless but needs high preprocessing cost because it executes an "expensive" procedure for finding the most distant items in clusters. Contrary to PSC, RHC is parameterless. Contrary to RSP3, RHC is a fast algorithm because it is based on the fast *k*-means clustering that is also sped-up by the class-mean initializations.

In [25, 28], RHC was compared to the state-of-the-art CNN-rule [16], IB2 [1, 2], RSP3, PSC on several datasets. The performance measurements were validated by the non-parametric test. The results show that almost in all cases, RHC appears to achieve the highest

reduction rates with the lowest preprocessing cost and an acceptable classification accuracy, similar to that of CNN-rule. Moreover, the results illustrated that the reduction rates achieved by RHC depend on the level of noise in the training data, and this is true for all data reduction algorithms.

3.4 The editing and reduction through homogeneous clusters algorithm

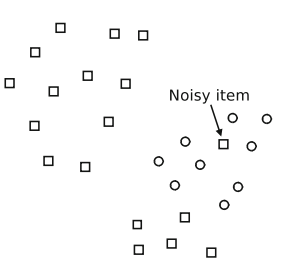
The Editing and Reduction through Homogeneous Clusters (ERHC) algorithm constitutes a combination RHC and EHC. Practically, it is a variation of RHC that can effectively manage datasets with noise. ERHC works in a way similar to RHC and EHC. However, whenever a homogeneous cluster is identified, ERHC counts the items in it. If it contains a single item (it is one-item cluster), ERHC removes it. If the homogeneous cluster has more than one item, its cluster-mean is computed and placed in the condensing set as a prototype. Therefore, the final condensing set built by ERHC contains the cluster-means of the homogeneous clusters that contain more than one item.

The aforementioned procedure is depicted in Fig. 5. Suppose that the initial training set contains two classes, squares and circles (Fig. 5a). ERHC computes two class-means (Fig. 5b). k-Means is applied on the training set and builds two clusters, A and B (Fig. 5c). A is homogeneous and contains more than one item. Thus, its cluster-mean is placed in the condensing set. B is non-homogeneous since it contains items from both classes. Therefore, ERHC computes two class-means (Fig. 5d), and then k-means is applied on B and builds clusters C and D (Fig. 5e). Since D is homogeneous and contains more than one item, its cluster-mean is placed in the condensing set. Since C is non-homogeneous, its class-means are computed (Fig. 5f) and k-means is applied on C building clusters E and F (Fig. 5g). Then, the cluster-mean of the homogeneous cluster F is placed in the condensing set, while the class-means are computed for the non-homogeneous cluster E (Fig. 5e). k-means is applied on E and the result is clusters G and H. Both are homogeneous (Fig. 5i). However, H is one-item-cluster. Hence, it is removed (it is not represented in the condensing set). The cluster-mean of G is placed in the condensing set (Fig. 5j). The final condensing set contains only four items (reduction rate over 85 %). Note that the only difference between RHC and ERHC is that RHC will place in the condensing set a prototype for cluster H.

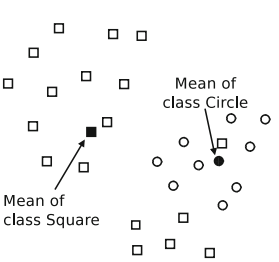
Obviously, ERHC is quite similar to RHC. However, we expect that the simple editing mechanism integrated in ERHC can effectively improve classification performance especially when data contains noise. ERHC inherits all the benefits of the algorithm for finding homogeneous cluster (Algorithm 4). Therefore, it is very fast and does not depend on the data order in the training set. We note that ERHC is not equivalent to the successive execution of EHC and RHC algorithms (let's call this EHC-RHC). Different clusters are built by the two approaches and consequently different reduction rates are achieved. Also, EHC-RHC has higher preprocessing cost than ERHC since it applies the procedure for finding homogeneous clusters twice: once on the original and once on the edited data. ERHC can simultaneously remove noise from the data and reduce the size of the training set.

4 Performance evaluation

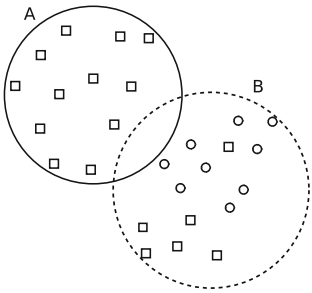
This section presents the results of the experiments we conducted. Subsection 4.1 presents the experimental study regarding EHC, while Subsection 4.2 concerns ERHC. In [25,



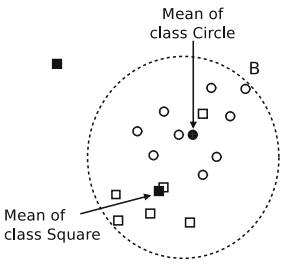
(a) initial data



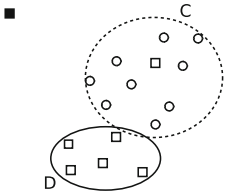
(b) means of classes



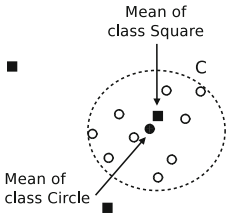
(c) k-Means on initial data



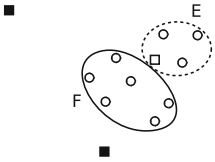
(d) means of classes in B



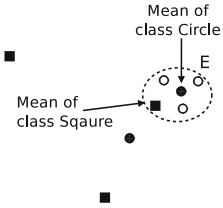
(e) k-Means on cluster B



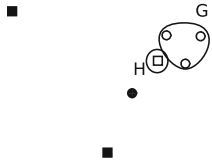
(f) means of classes in C



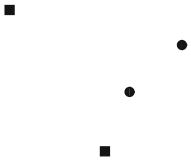
(g) k-Means on cluster C



(h) means of classes in E



(i) k-Means on cluster E



(j) final condensing set

Fig. 5 ERHC: data abstraction and editing process

28], we compared RHC to CNN-rule [16], IB2 [1, 2], RSP3 [29], PSC [22–24, 26] on original and edited forms of several datasets. Since the performance of RHC has been extensively evaluated and experimentally compared to state-of-the-art condensing and prototype abstraction algorithms, this paper does not include comparison experiments for the particular algorithms.

In all experimentations, the euclidean distance was adopted as the distance metric. Moreover, all algorithm implementations were coded in C. The experiments were run over datasets distributed by the KEEL dataset repository[1] [3] and the UCR Time Series Classification/Clustering Homepage.[2] All measurements are averages obtained via a five-fold cross-validation. Accuracy measurements were estimated by executing the 1-NN classifier on the resulting (reduced) training sets.

Please note that we have integrated the algorithms presented in this paper, i.e., EHC, RHC, ERHC, ENN-rule, Multiedit and All-k-NN as well as many other data reduction techniques in WebDR.[3] This is a web application that allows the user to plan and run experiments over several known datasets through an interactive interface. All conducted experiments can be easily re-executed by the interested reader.

4.1 Experimental study for EHC

4.1.1 Experimental setup

EHC was evaluated on fifteen datasets. We downloaded thirteen datasets from the repositories (the first ten from KEEL and the rest from UCR) and give their profile in Table 1. The LIR dataset is an almost noise-free dataset and some datasets have low levels of noise. Since, we wanted to test how editing behaves on noise-free datasets, we decided to include them in our experimentation. Moreover, we built two additional datasets by adding 10 % random noise in the LS and PH datasets. We refer to these datasets as LS-n and PH-n, respectively. The noise was added by setting the class label of the 10 % of the training items to a randomly chosen different class label. No other data transformation was performed.

For comparison purposes, we coded the three state-of-the-art algorithms presented in detail in Section 2 (ENN-rule [39], All-kNN [35], Multiedit [9]). An important issue that we had to address was the tuning of the parameters of the aforementioned methods. For all of them, we adopted the settings proposed in [13]. In particular, we used $k = 3$ for ENN-rule, $k = 7$ and $k = 9$ for All-kNN and $n = 3$ and $R = 2$ for Multiedit. These settings are very common in many experimental studies in the literature. In addition, we used $k = 5$ for ENN-rule and $n = 5$ for Multiedit after running several experiments with different parameter values. These experiments depicted that these values constitute a good choice. Finally, we also measured and present the performance of the conventional 1-NN classifier (classification without editing).

The four editing algorithms were compared to each other in terms of two main criteria: classification accuracy and preprocessing (editing) cost. The latter was estimated by counting the distances computed by each algorithm. Although the reduction rates achieved by each method do not indicate the best performing algorithm, they reveal the percentage

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

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

[3] https://ilust.uom.gr/webdr

Table 1 Dataset details

Dataset	Size (items)	Attributes	Classes
Magic Gamma Telescope (MGT)	19020	10	2
Landsat Satellite (LS)	6435	36	6
Phoneme (PH)	5404	5	2
Letter Image Recognition (LIR)	20000	16	26
Banana (BN)	5300	2	2
Ecoli (ECL)	336	7	8
Pima (PM)	768	8	2
Yeast (YS)	1484	8	10
MONK2 (MN2)	432	6	2
Twonorm (TN)	7400	20	2
CBF	930	128	3
Face All (FA)	2250	131	14
Synthetic Control (SC)	600	60	6

of data that is considered as noise by each algorithm. Therefore reduction rates are also reported.

4.1.2 Experimental measurements

The performance measurements of our experimental study are presented in Table 2. Each table cell contains three measurements that correspond to the execution of an editing approach on a particular dataset. The three measurements are: accuracy, reduction rate and preprocessing cost in millions of distance computations. The best measurements are in bold.

As we expected, EHC is the fastest approach. It achieves very low average preprocessing cost compared to its competitors (see the last row of the table). EHC computes the fewest distances in fourteen out of fifteen datasets. Furthermore, we observe that the cost gains are very high for large datasets. Finally, as we predicted in Section 3, EHC computes a completely different number of distances for LS, LS-n and PH, PH-n.

Concerning accuracy measurements, we observe that the proposed algorithm is comparable to ENN-rule and All-kNN. Multiedit has the worst accuracy, especially for the LIR and ECL datasets where its accuracy is unacceptable. Although the differences in accuracy between EHC, ENN and All-kNN are not statistically significant, we observe that EHC has the highest accuracy in seven out of fifteen datasets. However, ENN-rule has the highest average accuracy.

In some datasets, all editing approaches seem to negatively affect accuracy, since the conventional 1-NN classifier is the most accurate approach. This could be an indication that the datasets do not contain a lot of noise. However, in all these cases with the exception of the MN2 dataset, EHC is the most accurate editing algorithm. In contrast, for some datasets (MGT, BN, ECL, PM, YS, TN, LS-n and PH-n), most of the editing approaches achieve higher accuracy than the conventional 1-NN classifier. This could be an indication that the datasets contain a lot of noise and, therefore, it appears that editing constitutes a necessary preprocessing step.

The proposed algorithm has the lowest reduction rate. EHC removes items by using the strict criterion of one-item clusters. For datasets with extremely high levels of noise (e.g.,

Table 2 Comparisson of editing algorithms in terms of Accuracy (Acc(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC (millions of distance computations))

Dataset		1-NN	ENN1 ($k=3$)	ENN2 ($k=5$)	ME1 ($n=3$) ($R=2$)	ME2 ($n=5$) ($R=2$)	All$k$NN1 ($k=7$)	All$k$NN2 ($k=9$)	EHC
	Acc	78.14	80.44	80.57	76.75	75.26	**80.76**	80.86	79.52
MGT	RR	–	20.08	19.20	39.98	42.36	29.67	30.38	10.70
	PC	–	115.76	115.76	2,839.55	1,447.93	115.76	115.76	**4.08**
	Acc	**90.60**	90.30	90.43	86.79	86.03	90.12	90.16	**90.55**
LS	RR	–	9.07	9.27	24.13	26.17	13.92	14.51	3.11
	PC	–	13.25	13.25	266.22	139.53	13.25	13.25	**1.69**
	Acc	**90.10**	88.14	87.53	80.77	79.72	86.55	86.23	**89.06**
PH	RR	–	11.25	11.93	34.14	36.91	17.92	19.30	7.36
	PC	–	9.35	9.35	166.22	53.71	9.35	9.35	**0.66**
	Acc	**95.83**	94.98	94.87	70.94	8.35	94.28	94.00	**95.23**
LIR	RR	–	4.33	4.44	43.43	56.59	7.31	7.97	3.95
	PC	-	127.99	127.99	7,214.38	2,900.53	127.99	127.99	**41.85**
	Acc	86.91	89.36	89.55	89.83	**90.38**	89.509	89.79	88.60
BN	RR	–	11.53	10.98	20.12	21.64	17.10	17.51	10.65
	PC	–	8.99	8.99	106.69	60.26	8.99	8.99	**0.56**
	Acc	79.78	81.57	81.86	63.10	46.11	81.26	80.66	**82.16**
ECL	RR	–	20.45	20.45	47.29	60.15	28.63	30.48	17.01
	PC	–	0.036	0.036	0.100	0.055	0.036	0.036	**0.035**
	Acc	68.36	71.87	71.75	71.36	68.89	72.65	**73.30**	70.32
PM	RR	–	30.16	29.43	53.07	58.96	45.56	46.24	16.59
	PC	–	0.19	0.19	0.51	0.26	0.19	0.19	**0.06**
	Acc	52.16	56.47	57.07	52.90	50.54	58.29	**58.42**	54.45
YS	RR	–	45.73	43.89	74.34	80.93	59.90	61.25	29.58
	PC	–	0.70	0.70	1.19	**0.58**	0.70	0.70	0.84
	Acc	**90.505**	**89.580**	89.113	69.663	62.989	83.317	83.317	81.481
MN2	RR	–	2.081	0.925	47.861	58.728	12.370	12.659	0.405
	PC	–	0.060	0.060	0.218	0.112	0.060	0.060	**0.007**
	Acc	94.878	95.689	95.541	**96.608**	96.595	95.770	95.797	95.108
TN	RR	–	3.611	3.135	17.017	20.838	6.791	6.905	0.956
	PC	–	17.520	17.520	578.025	411.564	17.520	17.520	**1.642**
	Acc	**98.387**	**98.280**	**98.280**	86.989	80.538	98.172	98.172	**98.280**
CBF	RR	–	1.398	0.860	24.005	32.930	2.016	2.043	0.269
	PC	–	0.276	0.276	3.924	1.865	0.276	0.276	**0.057**
	Acc	**95.067**	92.578	92.222	67.778	53.689	91.333	90.844	**93.778**
FA	RR	–	5.867	7.556	48.133	63.900	10.211	11.444	3.844
	PC	–	1.619	1.619	18.957	6.216	1.619	1.619	**1.072**

Table 2 (continued)

Dataset		1-NN	ENN1 ($k = 3$)	ENN2 ($k = 5$)	ME1 ($n = 3$) ($R = 2$)	ME2 ($n = 5$) ($R = 2$)	All$k$NN1 ($k = 7$)	All$k$NN2 ($k = 9$)	EHC
	Acc	**91.667**	87.333	87.167	72.000	50.833	85.333	84.833	**91.667**
SC	RR	–	8.833	10.167	39.208	53.250	13.292	14.125	0.417
	PC	-	0.115	0.115	0.753	0.313	0.115	0.115	**0.040**
	Acc	82.58	89.64	89.74	86.47	85.55	89.73	**89.84**	87.55
LS-n	RR	–	19.82	18.45	38.33	40.19	29.64	30.17	10.93
	PC	–	13.25	13.25	139.02	78.43	13.25	13.25	**2.00**
	Acc	82.14	**86.94**	86.70	81.31	79.29	86.31	85.90	86.16
PH-n	RR	–	21.20	20.61	44.93	49.85	33.29	34.68	17.66
	PC	–	9.35	9.35	52.65	31.74	9.35	9.35	**0.71**
	Acc	85.140	**86.211**	86.160	76.884	70.984	85.559	85.475	85.594
AVG	RR	–	14.361	14.086	39.732	46.893	21.841	22.644	8.895
	PC	–	21.230	21.230	759.227	342.206	21.230	21.230	**3.687**

30 % or more), it is not certain that EHC will improve classification accuracy like ENN-rule with an appropriate k value does. On the other hand, EHC is not expected to negatively affect classification accuracy as much as the other methods do.

4.1.3 Non parametric statistical test

We validated the performance measurements by applying a non-parametric statistical test of significance [32]. In particular, we ran Friedman's 2-way ANOVA by ranks test with Dunn-Bonferroni post-hoc tests using IBM SPSS that compares all pairs of algorithms taking into consideration their performance measurements on each dataset.

We executed the particular test once on the fifteen measurements of each criterion (classification accuracy, preprocessing cost).

The test revealed that (a) in terms of accuracy, all methods with the exception of Multiedit1 are better than Multiedit2 and that both versions of ENN are also better than Multiedit1, and, (b) in terms of preprocessing cost, all methods with the exception of Multiedit2 are better than Multiedit1 and that EHC is also better than Multiedit2.

To summarize, EHC is better than Multiedit and as good as ENN and All-kNN.

4.2 Experimental study for ERHC

4.2.1 Experimental setup

ERHC was tested on sixteen datasets. These are the datasets we used in the experimental study of the previous subsection with the removal of the ECL and YS datasets and the addition of three extra large datasets from KEEL (PD, SH, TXR). We decided to exclude the ECL and YS datasets, because the editing mechanism eliminates all items that belong to rare classes (i.e., rare classes are not represented in the condensing set). Table 3 summarizes the datasets used.

Table 3 Dataset details

Dataset	Size (items)	Attributes	Classes
Letter Image Recognition (LIR)	20000	16	26
Pen-Digits (PD)	10992	16	10
Shuttle (SH)	58000	9	7
Texture (TXR)	5500	40	11
Banana (BN)	5300	2	2
Landsat Satellite (LS)	6435	36	6
Magic Gamma Telescope (MGT)	19020	10	2
Phoneme (PH)	5404	5	2
Pima (PM)	768	8	2
MONK2 (MN2)	432	6	2
Twonorm (TN)	7400	20	2
CBF	930	128	3
Face All (FA)	2250	131	14
Synthetic Control (SC)	600	60	6

ERHC is compared to the RHC, ENN-RHC and EHC-RHC algorithms. ENN-RHC is the successive execution of ENN-rule for editing and RHC for data abstraction, whereas EHC-RHC is the successive execution of EHC for editing and RHC for data abstraction. We used ENN-rule because it is the reference editing algorithm. We ran experiments with $k = 3$ because that value is either adopted or suggested by many researchers [13, 21, 40]. The four algorithms were evaluated by estimating three measurements: accuracy, reduction rate, and, preprocessing cost in terms of distance computations.

In this experimental study, ERHC is not compared to state-of-the-art condensing and prototype abstraction algorithms. In effect, this study complements the experimental study presented in [28] where RHC is extensively evaluated against several state-of-the-art condensing and prototype abstraction algorithms.

4.2.2 Experimental measurements

Table 4 presents the measurements obtained with the best ones shown in bold. Preprocessing cost measurements are in million distances. In addition, Table 4 reports the accuracies obtained by applying the conventional 1-NN classifier (1-NN) on the original training data (without data reduction).

For the BN, PM, MN2, CBF, SC, LS-n and PH-n datasets, one or more algorithms achieved higher accuracy than conventional 1-NN. Almost in all datasets, ERHC appears to achieve higher accuracy and reduction rate measurements than RHC. It is worth mentioning that no editing approach (ERHC included) negatively affects accuracy on noise-free datasets (e.g., LIR, PD, SH, TXR).

We expected EHC-RHC to have higher reduction rates than ERHC. However, the results show that this is not always true. In nine datasets, ERHC has higher reduction rates than EHC-RHC. Nevertheless, the differences are not significant.

Concerning the preprocessing cost, we observe that in all cases EHC-RHC has almost the double cost compared to ERHC and RHC. Also, ENN-RHC is the slowest approach. This is because it includes the execution of the costly ENN-rule procedure. In practice, the

Table 4 DRT comparison in terms of Accuracy (Acc(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC(M))

Dataset		1-NN	RHC	ENN-RHC	EHC-RHC	ERHC
	Acc	95.825	**93.585**	92.720	93.045	92.690
LIR	RR	–	88.081	90.343	90.383	**92.029**
	PC	–	**41.844**	159.039	73.710	**41.844**
	Acc	99.354	98.299	98.453	98.472	**98.626**
PD	RR	–	96.516	97.189	**97.589**	97.448
	PC	–	**2.882**	41.489	5.521	**2.882**
	Acc	99.822	98.095	**99.597**	98.481	98.038
SH	RR	–	99.550	99.658	99.669	**99.690**
	PC		**16.827**	1098.864	32.695	**16.827**
	Acc	99.018	97.036	97.109	96.873	**97.364**
TXR	RR	–	94.705	95.582	95.732	**95.936**
	PC	–	**3.629**	12.675	6.133	**3.629**
	Acc	86.906	83.283	**88.094**	87.019	88.000
BN	RR	–	79.684	**95.660**	93.000	90.330
	PC	–	**0.562**	9.519	1.014	0.562
	Acc	90.598	88.951	**89.138**	88.392	89.013
LS	RR	–	89.841	**95.062**	92.273	92.949
	PC	–	**1.693**	14.984	3.192	**1.693**
	Acc	78.144	71.966	**77.781**	74.716	77.014
MGT	RR	–	73.757	**93.057**	83.843	84.456
	PC	–	**4.082**	118.591	7.480	**4.082**
	Acc	90.100	85.585	85.400	86.158	**86.565**
PH	RR	–	80.708	**92.098**	89.008	88.053
	PC	–	**0.658**	9.812	1.161	**0.658**
	Acc	68.358	63.281	**72.653**	69.927	69.793
PM	RR	–	63.577	**91.792**	80.977	80.065
	PC	–	**0.062**	0.219	0.103	**0.062**
	Acc	90.505	94.678	**96.750**	94.221	95.140
MN2	RR	–	96.474	**97.043**	96.696	96.763
	PC	–	**0.007**	0.067	0.015	**0.007**
	Acc	94.878	88.689	**93.108**	91.473	91.527
TN	RR	–	96.628	**98.517**	97.588	97.584
	PC	–	**1.642**	18.884	3.357	**1.642**
	Acc	98.387	**98.602**	**98.602**	98.495	98.495
CBF	RR	–	97.742	**98.011**	97.957	**98.011**
	PC	–	**0.057**	0.319	0.112	**0.057**
	Acc	95.067	**93.022**	90.800	91.111	91.200
FA	RR	–	87.811	90.189	91.122	**91.656**
	PC	–	**1.072**	2.377	1.908	**1.072**

Table 4 (continued)

Dataset		1-NN	RHC	ENN-RHC	EHC-RHC	ERHC
	Acc	91.667	98.667	98.333	**98.833**	98.667
SC	RR	–	97.292	**97.792**	97.667	97.708
	PC	–	**0.040**	0.151	0.079	**0.040**
	Acc	82.580	78.819	**88.578**	84.817	85.377
LS-n	RR	-	76.632	**95.361**	88.465	87.560
	PC	-	**1.999**	14.744	3.637	**1.999**
	Acc	82.143	75.407	83.993	81.255	**84.030**
PH-n	RR	-	64.246	**92.019**	86.394	81.910
	PC	-	**0.706**	116.164	1.180	**0.706**
	Acc	90.210	87.998	**90.694**	89.581	90.096
Avg	RR	-	86.453	**94.961**	92.398	92.009
	PC	-	**4.860**	101.119	8.831	**4.860**

actual preprocessing cost of ENN-rule may be even higher: tuning its parameter may require multiple executions of a trial-and-error-procedure.

Although ENN-RHC is the slowest approach, it seems to be slightly more accurate than EHC-RHC and ERHC. Moreover, it achieves higher reduction rate measurements on datasets with high levels of noise. This is because ENN-rule considers as noise more items than EHC and ERHC do.

4.2.3 Non parametric statistical test

We validated the performance measurements by applying Friedman's 2-way ANOVA by ranks test with Dunn-Bonferroni post-hoc tests using IBM SPSS. We ran the test once for each comparison criterion (classification accuracy, reduction rate, preprocessing cost (PC)).

The test revealed that (a) in terms of accuracy all methods perform the same, (b) in terms of reduction rate, all methods are better than RHC – this is expected since data reduction always works better on edited data, and, (c) in terms of preprocessing cost, RHC and ERHC are better than ENN-RHC and EHC-RHC.

5 Conclusions

In this paper, we presented a family of three algorithms that are based on a k-means clustering procedure that forms homogeneous clusters. Main motive behind the algorithms constitutes the fast and parameterless (independent of tuning parameters via trial-and-error procedures) preprocessing of the training set. Certainly, effective classification is also an important goal. RHC is a simple "general purpose" prototype abstraction algorithm that achieves high performance. EHC is an editing algorithm that aims to improve the quality of the training data, and as a consequence, the accuracy rather than the speed of classification. This is accomplished by removing outliers, noise, mislabelled and close-border training items. Last but not least, ERHC is a simple variation of RHC that integrates the idea of EHC

editing. Therefore, it can achieve high reduction rates regardless the level of noise in the training data and without requiring as high preprocessing cost as a sequential execution of EHC and RHC. EHC and ERHC were empirically evaluated on several datasets. The experimental results illustrated that they meet the goals for which they were developed, they are at least as good as the state-of-the-art methods and in some cases they improve classification performance.

6 Compilance with ethical standards

A previous version of this paper appears in the Eighth International Symposium on Foundations of Information and Knowledge Systems (FoIKS) (Beierle and Meghini, 2014). The authors visited Bordeaux, France to attend the aforementioned symposium and present the paper and financial support for the travel expenses was provided by the University of Macedonia, Greece. Stefanos Ougiaroglou was supported by the State Scholarships Foundation of Greece (I.K.Y.) with a scholarship for the duration of his Ph.D studies (November 2010 to June 2014). We thank the anonymous reviewers for their useful comments.

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
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
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. MVLSC **17**(2-3), 255–287 (2011)
4. Barandela, R., Gasca, E.: Decontamination of training samples for supervised pattern recognition methods. In: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition, pp. 621–630. Springer-Verlag, London (2000). http://dl.acm.org/citation.cfm?id=645889.673580
5. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. **13**(1), 21–27 (2006). doi:10.1109/TIT.1967.1053964
6. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
7. Dasarathy, B.V., Snchez, J.S., Townsend, S.: Nearest neighbour editing and condensing toolssynergy exploitation. Pattern. Anal. Applic. **3**(1), 19–30 (2000). doi:10.1007/s100440050003
8. Derrac, J., Cornelis, C., García, S., Herrera, F.: Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection. Inf. Sci. **186**(1), 73–92 (2012). doi:10.1016/j.ins.2011.09.027
9. 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)
10. 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
11. 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
12. Garcia, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer Publishing Company, Incorporated (2014)
13. 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

14. García-Pedrajas, N., De Haro-García, A.: Boosting instance selection algorithms. Know.-Based Syst. **67**, 342–360 (2014). doi:10.1016/j.knosys.2014.04.021
15. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science (2011)
16. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. Inf. Theory **14**(3), 515–516 (1968)
17. Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern classification problem. Pattern Recogn. **33**(3), 521–528 (2000). doi:10.1016/S0031-3203(99)00068-0. http://www.sciencedirect.com/science/article/pii/S0031320399000680
18. Jiang, Y., hua Zhou, Z.: Editing training data for knn classifiers with neural network ensemble. In: Lecture Notes in Computer Science, Vol.3173, pp. 356–361. Springer (2004)
19. Lozano, M.: Data Reduction Techniques in Classification processes (Phd Thesis). Universitat Jaume I (2007)
20. 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)
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
22. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: A new fast prototype selection method based on clustering. Pattern. Anal. Applic. **13**(2), 131–141 (2010)
23. 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, 674–683. Springer-Verlag, Berlin, Heidelberg (2007). http://dl.acm.org/citation.cfm?id=1782914.1782996
24. Olvera-Lpez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: Object selection based on clustering and border objects. In: Kurzynski, M., Puchala, E., Wozniak, M., Zolnierek, A. (eds.) Computer Recognition Systems 2, Advances in Soft Computing, vol. 45, pp. 27–34. Springer (2008)
25. 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
26. 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)
27. Ougiaroglou, S., Evangelidis, G.: EHC: Non-parametric editing by finding homogeneous clusters. In: Beierle, C., Meghini, C. (eds.) Foundations of Information and Knowledge Systems, Lecture Notes in Computer Science, vol. 8367, pp. 290–304. Springer International Publishing (2014). doi:10.1007/978-3-319-04939-7_14
28. Ougiaroglou, S., Evangelidis, G.: RHC: a non-parametric cluster-based data reduction for efficient k-nn classification. Pattern Analysis and Applications pp. 1–17 (2014). doi:10.1007/s10044-014-0393-7
29. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recogn. **37**(7), 1561–1564 (2004)
30. Sánchez, J.S., Barandela, R., Marqués, A.I., Alejo, R., Badenas, J.: Analysis of new techniques to obtain quality training sets. Pattern Recogn. Lett. **24**(7), 1015–1022 (2003). doi:10.1016/S0167-8655(02)00225-8
31. Segata, N., Blanzieri, E., Delany, S.J., Cunningham, P.: Noise reduction for instance-based learning with a local maximal margin approach. J. Intell. Inf. Syst. **35**(2), 301–331 (2010). doi:10.1007/s10844-009-0101-z
32. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. A Chapman & Hall book. Chapman & Hall/CRC (2011)
33. Snchez, J., Pla, F., Ferri, F.: On the use of neighbourhood-based non-parametric classifiers. Pattern Recogn. Lett. **18**(1113), 1179–1186 (1997). doi:10.1016/S0167-8655(97)00112-8. http://www.sciencedirect.com/science/article/pii/S0167865597001128
34. Snchez, J., Pla, F., Ferri, F.: Prototype selection for the nearest neighbour rule through proximity graphs. Pattern Recogn. Lett. **18**(6), 507–513 (1997). doi:10.1016/S0167-8655(97)00035-4. http://www.sciencedirect.com/science/article/pii/S0167865597000354
35. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **6**, 448–452 (1976)
36. 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
37. Tsai, C.F., Eberle, W., Chu, C.Y.: Genetic algorithms in feature and instance selection. Know.-Based Syst. **39**, 240–247 (2013). doi:10.1016/j.knosys.2012.11.005

38. Vázquez, F., Sánchez, J.S., Pla, F.: A stochastic approach to wilson's editing algorithm. In: Proceedings of the Second Iberian conference on Pattern Recognition and Image Analysis - Volume Part II, IbPRIA'05, pp. 35–42. Springer-Verlag, Berlin, Heidelberg (2005). doi:10.1007/11492542_5
39. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **2**(3), 408–421 (1972)
40. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-basedlearning algorithms. Mach. Learn. **38**(3), 257–286 (2000). doi:10.1023/A:1007626913721
41. Wu, J.: Advances in K-means Clustering: A Data Mining Thinking. Springer Publishing Company, Incorporated (2012)