EHC: Non-parametric Editing by finding Homogeneous Clusters

Stefanos Ougiaroglou* and Georgios Evangelidis

Department of Applied Informatics, School of Information Sciences,
University of Macedonia, 156 Egnatia str, GR-54006 Thessaloniki, Greece
{stoug, gevan}@uom.gr

Abstract. Editing is a crucial data mining task in the context of k-Nearest Neighbor classification. Its purpose is to improve classification accuracy by improving the quality of training datasets. To obtain such datasets, editing algorithms try to remove noisy and mislabeled data as well as smooth the decision boundaries between the discrete classes. In this paper, a new fast and non-parametric editing algorithm is proposed. It is called Editing through Homogeneous Clusters (EHC) and is based on an iterative execution of a clustering procedure that forms clusters containing items of a specific class only. Contrary to other editing approaches, EHC is independent of input (tuning) parameters. The performance of EHC is experimentally compared to three state-of-the-art editing algorithms on ten datasets. The results show that EHC is faster than its competitors and achieves high classification accuracy.

Keywords: k-NN classification, clustering, editing, noisy items

1 Introduction

Classification is a traditional data mining problem that has attracted the interest of many researchers in the past decades [11]. 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 (TS) as the classification model.

A popular lazy classification method is k-Nearest Neighbor (k-NN) classifier [5]. It is simple, very easy to implement and has many applications. The k-NN classifier works as follows: for each new item x, it searches TS and retrieves the k nearest items to x according to a distance metric. The class of x is determined by a majority vote, i.e., the most common class among the classes of the k nearest neighbors. Possible ties during voting can be resolved either randomly or by assigning x to the class of the nearest neighbor.

* S. Ougiaroglou is supported by a scholarship from State Scholarships Foundation of Greece (I.K.Y.)

k-NN classifier is considered to be an effective classifier. However, it has some weaknesses. The first one is high computational cost since it must compute all distances between each unclassified item and all items in TS. In cases of large datasets, this drawback renders its use a time-consuming procedure and in some cases even prohibitive. Another weakness is large storage requirements for storing TS. A third weakness is that k-NN classifier is a noise-sensitive method. Classification accuracy depends on the level of noise in TS. Usage of high k values extend the examined neighbourhood and thus can partially remedy this drawback. However, this implies a high number of trial-and-error executions to determine the appropriate k value and that noise is uniformly distributed in TS (otherwise, dynamic determination of k should be adopted [19]).

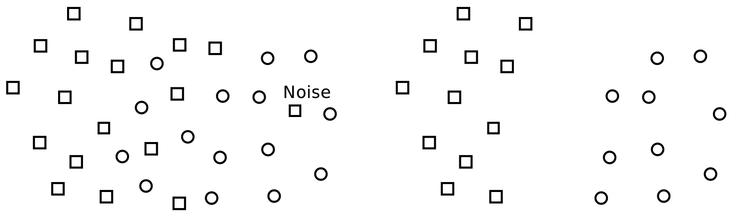
Data Reduction Techniques (DRTs) [26, 8, 15, 25, 28, 13, 10, 4, 17] can effectively deal with the aforementioned weaknesses. They can be divided into Prototype Selection (PS) [8] and Prototype Abstraction (PA) [26] algorithms. PS algorithms select items from TS whereas PA algorithms generate items by summarizing similar items from TS and use them as prototypes.

PS algorithms are divided into condensing and editing algorithms. PA and PS-condensing algorithms aim to built a small representative set (condensing set) of the initial TS. Usage of a Condensing Set (CS) has the benefits of low computational cost and storage requirements while accuracy remains at high levels. PS-editing algorithms aim to improve accuracy rather than achieve high reduction rates. To achieve this, they remove noisy and mislabelled items and smooth the decision boundaries (see Figure 1). Ideally, a PS-editing algorithm builds an Edited training Set (ES) without overlaps between the classes.

The reduction rates of many PA and PS-condensing algorithms depend on the level of noise in TS. The higher the level of noise, the lower the reduction rates achieved. Therefore, effective application of such algorithms implies removal of noise from the data, i.e., application of an editing algorithm beforehand [6, 15]. Hence, editing has a double goal: accuracy improvement and effective application of PA and PS-condensing algorithms. We should mention that some condensing algorithms, such as IB3 [1], integrate the idea of editing into their reduction procedures. These algorithms are called hybrid (see [8, 26] for details).

Although PS-editing algorithms contribute in obtaining high quality training data, they constitute a costly preprocessing step. Moreover, most PS-editing algorithms are parametric, i.e., the user defines the values of certain input (tuning) parameters. This implies time-consuming trial-and-error procedures to tune the parameters. These observations are behind the motivation of this paper. The contribution is the development and evaluation of a fast, non-parametric PS-editing algorithm that is based on a k-means clustering [16] procedure that forms homogeneous clusters. The proposed algorithm is called Editing through Homogeneous Clusters (EHC), leads to accurate k-NN classifiers and has low preprocessing cost.

The rest of the paper is organized as follows: Section 2 reviews the most well-known editing algorithms. Section 3 presents the proposed EHC algorithm.



(a) Initial training set

(b) Edited set

Fig. 1: Smoothing decision boundaries and removing noisy items

Performance evaluation experiments are presented in Section 4 and, finally, Section 5 concludes the paper.

2 Editing algorithms

2.1 The Edited Nearest Neighbor (ENN) rule

The reference editing algorithm is Wilson's Edited Nearest Neighbor (ENN) rule [29]. It constitutes the base of all other editing algorithms. ENN-rule is very simple. Algorithm 1 contains the pseudocode of the algorithm. Initially, the edited set (ES) is set to be equal to the 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 neighbors, it is removed from ES (lines 4–7). ENN-rule considers wrongly classified items to be noisy or close-border items and, thus, they must be removed. Note that, in each algorithm iteration, ENN-rule searches for nearest neighbors in the original TS and not in the "under construction" ES.

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 $ES$
```

Obviously, the cost of editing depends on the size of TS. In cases of large datasets, ENN-rule is a time-consuming algorithm. ENN-rule must compute all distances between the items of TS. Therefore, $\frac{N*(N-1)}{2}$ distances must be estimated, where N is the number of items in TS.

A crucial issue that should be addressed is the determination of the value of k that defines the size of the examined neighborhood. [28, 9, 17] consider $k = 3$ to be a typical value. This is adopted in many papers (e.g. [20]), whereas, other papers use $k = 3$ and additional k values (e.g., [23, 12]). In some cases, researchers determine the value of k that achieves the best performance through trial-and-error procedures (e.g., [27]). In [29], the impact of k is discussed in detail. Furthermore, in [12], 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 non-noisy items (see [9]) or keep noisy items. This happens because ENN-rule uses a unique k value for the entire TS. Different k values may be optimal for different regions in space.

2.2 All *k*-NN

All-kNN [24] 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, ES is set to be the whole TS (line 1). For each item x in TS (line 2), All-kNN applies 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 8–11). 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 through a trial-and-error procedure. M. Garcia-Borroto et al. consider $kmax = 7$ or $kmax = 9$ to be appropriate values [9].

2.3 Multiedit

Multiedit [7] is another well-known editing approach (see Algorithm 3). Initially, ES is set to be equal to 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 module n following subset, i.e., $s_{(i+1)modn}$ (line 8). The misclassified items are removed from ES (line 10). Then, 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 15–19, line 21).

Algorithm 2 All-kNN

```
Input: $TS, kmax$
Output: $ES$
ES ← TS
for each $x \in TS$ do
    $k \leftarrow 1$
    $flag \leftarrow FALSE$
    while $(k \leq kmax)$ and $(flag == FALSE)$ do
        $NNs \leftarrow$ find the $k$ nearest to $x$ neighbors in $TS - \{x\}$
        $majorClass \leftarrow$ find the most common class in $NNs$
        if $x_{class} \neq majorClass$ then
            $ES \leftarrow ES - \{x\}$
            $flag \leftarrow TRUE$
        end if
        $k \leftarrow k + 1$
    end while
end for
return $ES$
```

Here, parameter k is not used since multiedit utilizes 1-NN classifier during editing. However, parameters n and R influence the resulting ES. Parameter $n \geq 3$ defines the number of subsets. In many papers (e.g., [9, 23]), $n = 3$ is either adopted or proposed. Parameter R defines the number of non-editing iterations. In [9], $R = 2$ 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 noisy, outlier and close-border items. However, it may also remove non-noisy items. If items of two or more classes are close to each other, multiedit may eliminate entire classes [9]. Another drawback of multiedit is that it is based on a random formation of subsets, i.e., repeated applications may build a completely different ES from the same TS.

Multedit is usually more time-consuming than ENN-rule. However, it may compute even fewer than $\frac{N*(N-1)}{2}$ distances. An efficient implementation of multiedit does not compute a distance more than once. However, since the distances that have been already computed should be available until the end of the execution, such an implementation requires more memory. In cases where each distance is 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.

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 then
        $r \leftarrow r + 1$
    else
        $r \leftarrow 0$
    end if
    $TS \leftarrow ES$
until $r == R$ {until the last R iterations do not edit data}
return $ES$
```

EENProb and ENNth [27] are extensions of ENN-rule. Both retrieve the k nearest neighbors, and then perform editing based on probability estimations. Repeated ENN (RENN) rule [24] 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 [12], another simple variation of ENN is proposed. It places an item in ES, if all its k nearest neighbors have 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 [23]. They are also based on the concept of removal of misclassified items. To the best of our knowledge, they are the only non-parametric editing algorithms. Nevertheless, the type of proximity graphs used influence the resulting ES. In [23], 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 to be parametric methods.

k-NCN editing and its iterative version [20] are also based on ENN-rule. Particularly, they use k nearest centroid neighborhood classifier [22] instead of k-NN classifier. Both are based on the following simple idea: the appropriate neighborhood that should be examined for each item is defined by taking into consideration not only its nearest neighbors but also the symmetrical distribution of neighbors around it.

In [3, 20] a depuration algorithm is proposed for editing training data. In addition to removing some items from TS, the algorithm also changes the class labels of some items. To achieve this, it uses two input parameters (see [3] or [20] for details). [14] considers and evaluates editing approaches based on the depuration algorithm and proposes the Neural 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 [21] proposes the use of local support vector machines for noise reduction. Like the other methods, its performance depends on parameter tuning.

3 Editing through Homogeneous Clusters (EHC) algorithm

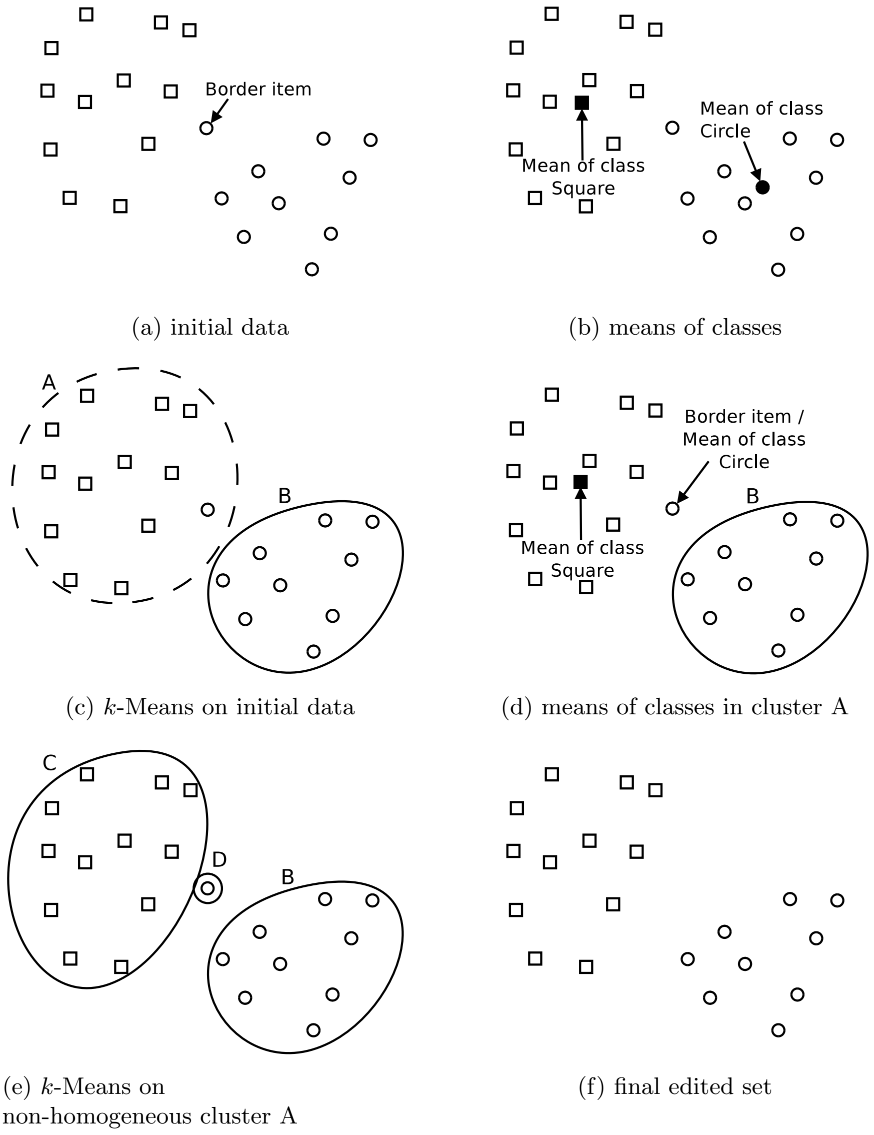
As we already mentioned in Section 2, PS-editing algorithms either extend ENN-rule or are based on the same idea. The proposed EHC algorithm follows a completely different, non-parametric strategy in order to remove noisy, mislabeled and close-border data items. Actually, it is based on RHC [18], a PA algorithm we have recently proposed. EHC iteratively applies k-means clustering on TS until all constructed clusters contain items of a specific class only, i.e., they are homogeneous. In the process, EHC removes all the clusters that contain only one item. We call these clusters one-item clusters.

Initially, EHC considers TS to be a non-homogeneous cluster. The algorithm computes a mean item for each class (class-mean) by averaging the corresponding items in the non-homogeneous cluster. If the cluster contains items from c classes, EHC computes c means. Then, it applies k-means clustering on the cluster, using the class-means as initial means, and builds c clusters. k-means clustering is recursively applied on the items of each non-homogeneous cluster built. One-item clusters are removed.

Two examples that demonstrate the operation of EHC are depicted in Figures 2 and 3. More specifically, Figure 2 demonstrates how EHC identifies and removes a close-border item, while Figure 3 demonstrates how the algorithm removes a noisy item. Note that non-homogeneous clusters are depicted with dashed borders. EHC identifies and removes outliers in a similar way.

Of course, EHC may assign a typical data item (non-noisy, non-close-border) 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.

Algorithm 4 describes a possible implementation of EHC. It utilizes a queue data structure Q in order to hold the unprocessed clusters. Initially, ES is set to be TS (line 1) and Q includes the whole TS as one unprocessed cluster (lines 2–3). In each algorithm iteration, cluster C is taken from the head of Q and is examined (line 5). If C is homogeneous (line 6), the algorithm counts the items in C and if C is a one-item cluster, its item is removed from ES (lines 7–9). If C



(a) initial data

(b) means of classes

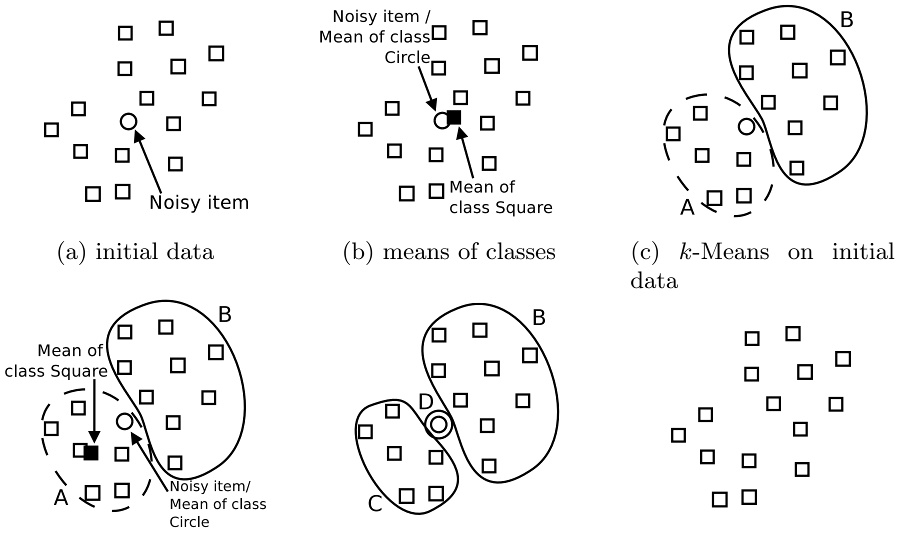
(c) *k*-Means on initial data

(d) means of classes in cluster A

(e) *k*-Means on non-homogeneous cluster A

(f) final edited set

Fig. 2: EHC: Smoothing decision boundaries



(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: Removing noisy items

is a non-homogeneous cluster, the class means for all the classes present in it are computed and added to set R (lines 11–14). Set R and cluster C are the input parameters to k-means clustering (line 15). The returned clusters are enqueued in Q (lines 16–18). The loop continues as long as there are non-homogeneous clusters (line 20).

Concerning the computational cost, we can easily conclude that EHC is a fast algorithm. 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 contrary to all other editing methods, EHC does not compute distances between "real" items. It computes distances between items and 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. Finally, the main advantage of the proposed method is that it is non-parametric. Therefore, there is no need for time-consuming trial-end-error procedures. Finally, note that EHC builds the same ES regardless of data ordering.

Algorithm 4 EHC

```
Input: $TS$
Output: $ES$
$ES \leftarrow TS$
$Q \leftarrow \varnothing$
Enqueue($Q$, $TS$)
repeat
  $C \leftarrow$ Dequeue($Q$)
  if $C$ is homogeneous then
    if $|C| = 1$ then
      $ES \leftarrow ES - C$
    end if
  else
    $R \leftarrow \varnothing$ {R is the set of class means}
    for each class $M$ in $C$ do
      $R \leftarrow R \cup mean\_of(M)$
    end for
    $Clusters \leftarrow K$-MEANS($C$, $R$)
    for each cluster $Cl \in Clusters$ do
      Enqueue($Q$, $Cl$)
    end for
  end if
until IsEmpty($Q$) {until all constructed clusters are homogeneous}
return $ES$
```

4 Performance evaluation

4.1 Experimental setup

The proposed EHC algorithm was coded in C and evaluated on ten datasets. We downloaded eight datasets from KEEL dataset repository[1] [2]. Their main characteristics are shown in Table 1. Initially, we did not know the level of noise in each dataset. After our experimentation, we realized that LIR is an almost noise-free dataset and LS and PH have low levels of noise. Since, we wanted to test how editing behaves on noise-free datasets, we decided to include these datasets in our experimentation. Moreover, we built two additional datasets by adding 10% random noise in LS and PH. We refer to these datasets as LS-n and PH-n respectively. Practically, we changed the class label of each item with a probability of 0.1. No other data transformation was performed. No dataset included missing values. Finally, euclidean distance was adopted as the distance metric.

For comparison purposes, we coded the three state-of-the-art algorithms presented in detail in Section 2 (ENN-rule [29], All-kNN [24], Multiedit [7]). We coded and used a non-optimized implementation of multiedit that may recompute same distances more than once.

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

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

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 [9]. 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. Of course, 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. Accuracy measurements were estimated by executing 1-NN classifier on the edited sets. For each algorithm and dataset, we report the average accuracy and cost measurements obtained via a five-fold cross validation. We used the pairs of training/testing sets distributed by KEEL repository. Although the reduction rates achieved by each method do not indicate the best performing algorithm, they reveal the percentage of data that is considered as noise by each algorithm. Therefore reduction rates were estimated and are reported.

4.2 Comparisons

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 (Acc), reduction rate (RR) and preprocessing cost (PC). The best measurements are in bold.

As we expected, EHC is the fastest approach. It achieves very low average PC measurements compared to its competitors (see the last row of the table). EHC computes the fewest distances in nine out of ten datasets. Furthermore, we observe that the cost gains are very high for large datasets. Finally, as we mentioned in Section 3, EHC computes a completely different number of distances for LS, LS-n and PH, PH-n. Here, we should mention that multiedit would have computed as many distances as ENN-rule and All-kNN had we used a more efficient implementation.

Table 2: Experimental measurements Accuracy (Acc(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC (millions of distance computations))

Dataset		**1-NN**	**ENN**	**ENN**	**Multiedit**	**Multiedit**	**All*k*NN**	**All*k*NN**	**EHC**
			(k=3)	k=5)	(n=3, R=2)	(n=5, R=2)	(k=7)	(k=9)	
MGT	Acc	78.144	80.44	80.57	76.75	75.26	80.76	**80.86**	79.52
	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**
LS	Acc	**90.60**	90.30	90.43	86.79	86.03	90.12	90.16	**90.55**
	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**
PH	Acc	**90.10**	88.14	87.53	80.77	79.72	86.55	86.23	**89.06**
	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**
LIR	Acc	**95.83**	94.98	94.87	70.94	58.35	94.28	94.00	**95.23**
	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**
BN	Acc	86.906	89.36	89.55	89.83	**90.38**	89.509	89.79	88.60
	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**
ECL	Acc	79.781	81.57	81.86	63.10	46.11	81.26	80.66	**82.16**
	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**
PM	Acc	68.358	71.87	71.75	71.36	68.89	72.65	**73.30**	70.32
	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**
YS	Acc	52.156	56.47	57.07	52.90	50.54	58.29	**58.42**	54.45
	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
LS-n	Acc	82.067	89.96	**90.13**	86.70	85.86	89.74	89.79	89.67
	RR	-	20.21	18.08	37.90	40.25	30.01	30.57	13.80
	PC	-	13.25	13.25	131.95	94.98	13.25	13.25	**8.06**
PH-n	Acc	81.884	87.71	87.25	80.63	79.66	86.20	85.83	**88.40**
	RR	-	21.78	20.21	34.14	36.91	32.36	33.46	22.29
	PC	-	9.35	9.35	166.22	53.71	9.35	9.35	**4.35**
AVG	ACC	80.23	83.08	**83.10**	75.98	72.08	82.93	82.90	82.80
	RR	-	19.46	18.79	40.85	46.09	28.24	29.17	13.50
	PC	-	29.89	29.89	1,089.30	475.15	29.89	29.89	**6.22**

Concerning accuracy measurements, we observe that the proposed algorithm is comparable to ENN-rule and All-kNN. Multiedit has the worst accuracy, especially for LIR and ECL, where its accuracy is unacceptable. This happens because multiedit removes data that should not be removed. Although the differences in accuracy between EHC, ENN and All-kNN are not statistically significant, we observe that EHC has the highest Acc measurements in half the datasets. However, ENN-rule has the highest average Acc measurement.

For LIR, LS and PH that contain low levels of noise, all editing approaches seem to negatively affect accuracy since conventional 1-NN classifier achieves the highest Acc measurements. However, in all these cases, EHC is the most accurate editing algorithm. In contrast, in the rest seven datasets, most of the editing approaches achieve higher Acc measurements than conventional-1NN classifier. 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. 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.

5 Conclusions

Classification accuracy achieved by k-NN classifier strongly depends on the quality of the available training data. Noisy and mislabeled data as well as outliers and overlaps between regions of different classes are the reasons of bad classification performance for the particular classifier. Editing algorithms can improve classification accuracy by removing such data. In this paper, we presented a short review of editing algorithms. Then, we proposed a non-parametric algorithm, called Editing through Homogeneous Clusters (EHC), which follows a completely different strategy than the other editing approaches. EHC is based on a clustering procedure that forms homogeneous clusters in the training data. The clusters that contain only one item are considered redundant (they contain noisy, outlier or close-border items) and are removed. An experimental study with ten datasets showed that the proposed algorithm is very fast and achieves comparable classification accuracy to the state-of-the-art methods.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (Jan 1991), http://dx.doi.org/10.1023/A:1022689900470
2. 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)

3. 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, UK, UK (2000), http://dl.acm.org/citation.cfm?id=645889.673580
4. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Min. Knowl. Discov. 6(2), 153–172 (Apr 2002), http://dx.doi.org/10.1023/A:1014043630878
5. Dasarathy, B.V.: Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society Press (1991)
6. Dasarathy, B.V., Snchez, J.S., Townsend, S.: Nearest neighbour editing and condensing toolssynergy exploitation. Pattern Analysis & Applications 3(1), 19–30 (2000), http://dx.doi.org/10.1007/s100440050003
7. 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)
8. 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 (Mar 2012), http://dx.doi.org/10.1109/TPAMI.2011.142
9. 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. pp. 489–496. CIARP '09, Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10268-4_57
10. 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)
11. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science (2011)
12. Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern classification problem. Pattern Recognition 33(3), 521 – 528 (2000), http://www.sciencedirect.com/science/article/pii/S0031320399000680
13. 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)
14. 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)
15. Lozano, M.: Data Reduction Techniques in Classification processes (Phd Thesis). Universitat Jaume I (2007)
16. 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)
17. 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 (Aug 2010), http://dx.doi.org/10.1007/s10462-010-9165-y
18. Ougiaroglou, S., Evangelidis, G.: Efficient dataset size reduction by finding homogeneous clusters. In: Proceedings of the Fifth Balkan Conference in Informatics. pp. 168–173. BCI '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2371316.2371349

19. Ougiaroglou, S., Nanopoulos, A., Papadopoulos, A.N., Manolopoulos, Y., Welzer-Druzovec, T.: Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors. In: Proceedings of the 11th East European conference on Advances in databases and information systems. pp. 66–82. ADBIS'07, Springer-Verlag, Berlin, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1780119.1780129
20. 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 (Apr 2003), http://dx.doi.org/10.1016/S0167-8655(02)00225-8
21. 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 (Oct 2010), http://dx.doi.org/10.1007/s10844-009-0101-z
22. Snchez, J., Pla, F., Ferri, F.: On the use of neighbourhood-based non-parametric classifiers. Pattern Recognition Letters 18(1113), 1179 – 1186 (1997), http://www.sciencedirect.com/science/article/pii/S0167865597001128
23. Snchez, J., Pla, F., Ferri, F.: Prototype selection for the nearest neighbour rule through proximity graphs. Pattern Recognition Letters 18(6), 507 – 513 (1997), http://www.sciencedirect.com/science/article/pii/S0167865597000354
24. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics 6, 448–452 (1976)
25. Toussaint, G.: Proximity graphs for nearest neighbor decision rules: Recent progress. In: 34th Symposium on the INTERFACE. pp. 17–20 (2002)
26. 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 (Jan 2012), http://dx.doi.org/10.1109/TSMCC.2010.2103939
27. 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. pp. 35–42. IbPRIA'05, Springer-Verlag, Berlin, Heidelberg (2005), http://dx.doi.org/10.1007/11492542_5
28. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-basedlearning algorithms. Mach. Learn. 38(3), 257–286 (Mar 2000), http://dx.doi.org/10.1023/A:1007626913721
29. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE trans. on systems, man, and cybernetics 2(3), 408–421 (July 1972)