Received June 23, 2020, accepted June 26, 2020, date of publication June 29, 2020, date of current version July 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005815

ELS: A Fast Parameter-Free Edition Algorithm With Natural Neighbors-Based Local Sets for *k* Nearest Neighbor

SUWEN ZHAO[1,2] AND JUNNAN LI[3]
[1]Department of Electronic Engineering, Guilin University of Aerospace Technology, Guilin 541004, China
[2]College of Bioengineering, Chongqing University, Chongqing 400030, China
[3]Chongqing Key Laboratory of Software Theory and Technology, College of Computer Science, Chongqing University, Chongqing 400044, China

Corresponding author: Junnan Li (junnanli@cqu.edu.cn)

ABSTRACT The instance reduction is one of the data preprocessing methods and aims to remove noises and (or) redundant instances from the training set. In the instance reduction, one of the most representative techniques is the edition method which can remove harmful instances from the training set to improve the prediction accuracy of k nearest neighbor (KNN). Nevertheless, most of existing edition methods still have some drawbacks, such as the parameter dependency, high computational time and relatively low accuracy. To solve these problems, we present a new fast parameter-free edition method based on local sets with natural neighbors (ELS). In ELS, we define a new concept of local sets by introducing natural neighbors. ELS can use the local sets to keep more reasonable class boundaries and effectively filtering out noisy instances (including global outliers). The main advantages of ELS are that (a) it is parameter-free; (b) it can remove global outliers and noisy instances; (c) it is relatively fast. Experiments clearly verify that (a) ELS outperforms existing representative edition methods in improving the prediction accuracy of KNN; (b) ELS can improve the performance of the condensation method and hybrid method in terms of both accuracy and reduction; (c) ELS consumes relatively low running time.

INDEX TERMS Data preprocessing, instance reduction, edition methods, natural neighbors, local sets, nearest neighbors.

I. INTRODUCTION

The k-nearest neighbor (KNN) [1]–[3] is one of the most representative instance-based classifiers and usually found as a benchmark for experimental and theoretical studies in data mining, image processing, pattern recognition and machine learning, etc. Despite the good performance of KNN, it also has some drawbacks, such as large storage requirements, a great deal of computational time for classification and sensitivity to noises. The instance reduction (IR) [4], [5] can solve these problems by removing the noises and (or) redundant instances.

The IR is one of the data pre-processing techniques and often applied to instance-based classifiers (i.e., KNN). Specifically, the IR searches the optimized subset $\boldsymbol{S}$ in the same feature space as the original set $\boldsymbol{X}$, subject to accuracy($\boldsymbol{S}$) $\geq$ accuracy($\boldsymbol{X}$) [6]. In practical applications, we often treat IR as the bi-objective problem of the trade-off between reduction and accuracy. Therefore, there exist different solutions to different applications.

In general, the IR is divided into condensation methods, edition methods and hybrid methods. Condensation methods usually remove some well-classified instances and then obtain a minimal set that does not affect the performance of the whole training set. The condensing nearest neighbor (CNN) [7] is the most classical condensation method. Subsequently, some improved algorithms of condensation methods [8]–[13], such as the SNN (selective nearest neighbor rule) [8], the GCNN (generalized condensed nearest neighbor rule) [9] and the IB2-3 (instance-based) [10], are proposed. Compared to condensation methods, edition methods aim to improve the prediction accuracy of the trained classifier by filtering out harmful instances (i.e., noises). One of the representative examples of the edition method is the ENN (edited nearest neighbor) [14] which filters out some instances that are not misclassified by KNN. Other variants of ENN include the all-KNN (ALL-KNN) [15], the repeated ENN (RENN) [15] and the modified ENN (MENN) [16].

The associate editor coordinating the review of this manuscript and approving it for publication was Nilanjan Dey.

Hybrid methods, which combine the characteristics of edition methods and condensation methods, also have been widely used for the time being [17]–[22]. The edition method is our focus in this paper.

Recently, many improved approaches [23]–[32] of edition methods have been proposed by using different theories and algorithmic models. However, most of the existing edition methods [23]–[32] still have drawbacks.

- Most of them heavily rely on user-defined parameters, resulting in difficulty in application and instability in performance.
- Although some parameter-free methods [27], [31] are developed, they still achieve relatively low accuracy or (and) consume relatively high running time.

To solve these drawbacks above, we introduce a new fast parameter-free edition method with natural neighbors-based local sets (ELS) in this paper. In ELS, we introduce natural neighbors [33] to define a new concept for local sets without any parameters. Compared to the existing local set [31], our local set has faster computational time and better describes the local characteristics. Through ELS, we can keep more reasonable class boundaries, while effectively filtering out noisy instances and global outliers. The main advantages of ELS are that (a) it is parameter-free; (b) it can remove global outliers and noisy instances; (d) it is relatively fast.

In our experiments, we adopt 4 representative edition methods, 1 condensation method and 1 hybrid method as comparison algorithms. Also, 22 real data sets are used to evaluate the proposed algorithm. Main contributions of our work are listed as follows:

(a) We propose a fast parameter-free edition algorithm (ELS) to remove noisy instances and several global outliers effectively. Compared with representative ones, ELS is parameter-free, has faster computational time and improves the accuracy of KNN better.

(b) We define a new concept for local sets by the fast searching for natural neighbors. Compared to the local set of the work [30], [31], the new concept for local sets can describe the local characteristics of data sets faster and better.

(c) Condensation methods and hybrid methods can use ELS as a noise filter to improve their performance in terms of both accuracy and reduction.

The rest of the paper is organized as follows. Section II describes related work. In Section III, some preparations are introduced. In Section IV, we describe our algorithm. Comparison experiments prove the effectiveness of our algorithm in Section V. Section VI concludes this paper and makes plans for the future.

II. RELATED WORK

The earliest edition method is ENN proposed by Wilson [14]. Many hybrid methods, such as the ATISA [18] (adaptive threshold-based instance selection algorithm), the BNNT (binary nearest neighbor tree) [19] and the IRB (instanceRank based on borders) [21], use the ENN as a step to filter out noisy instances. Despite the excellent performance of ENN, it is oversimple. Recently, some improvements based on ENN are proposed to improve the performance of edition methods. To sum up, they can be divided into three categories, edition methods based on nearest neighbors (NN-based edition methods), graph-based edition methods and edition methods based on local sets.

Most of the edition methods including ENN are based on nearest neighbors. Two well-known variants of ENN are the RENN [15] and the ALL-KNN [15]. The ALL-KNN run ENN k times, where the number of neighbors varies from 1 to k. In addition, the RENN runs ENN repeatedly until no noisy instances are removed. The MENN [16] is another variant of ENN, and it removes the noisy instance if it does not agree with all of its $k+l$ nearest neighbors, where l is the number of instances in the training set which are at the same distance as the last neighbor. By considering sample-size sensitivity, the multi-edit (Multiedit) [26] is proposed. It divides the training set into n blocks ($n > 2$), where an instance in each block misclassified by KNN is discarded. By taking account of the symmetric distribution of prototypes, the nearest centroid neighbor edition (NCNEdit) [24] is proposed, where an instance can be removed if it is misclassified by its k nearest centroid neighbors. In addition, the edited nearest-neighbor estimating class probabilistic and threshold (ENNTh) [25] is also an improved algorithm of ENN. The ENNTh applies a probabilistic NN rule and deletes an instance from the training set if it is misclassified by this probabilistic NN rule. In recent years, an edition method based on natural neighbors (ENaN) [27] have been developed to improve ENN. ENaN filters out noisy instances if they are misclassified by their natural neighbors. In general, the performance of almost all existing NN-based edition methods depends on the selection of parameter k. Moreover, most of them have relatively high time complexity (i.e., $O(n^3)$).

Graph-based edition methods usually need to construct a very complex structure of graphs, and then they use the information of edges and points implicit in the graph structure to delete noisy instances. Therefore, most of the graph-based approaches have a high time complexity of $O(n^3)$ and depend on parameters. The most representative algorithms of graph-based edition methods are the relative neighborhood graph edition (RNGE) [23] and the cut edges weight statistic (CEWS) [28]. They filter out noisy instances by constructing the relative neighborhood graph. The edition based on hit miss networks (HMN-E) [29] is another graph-based edition method. HMN-E constructs a k-nearest neighbor graph ($k = 1$) and deletes the sample if it is isolated or has more ‘miss’ than ‘hit’ points.

The local set is proposed by Brighton and Mellish [30]. Recently, Leyva *et al.* propose a new edition method based on local sets (LSSm) [31]. The LSSm deletes sample $\boldsymbol{x}$ if $harmfulness(\boldsymbol{x}) > usefulness(\boldsymbol{x})$. Although LSSm is parameter-free and sometimes achieves relatively high performance, the process of computing local sets is also relatively time-consuming (the time complexity is $O(n^2)$).

III. PRELIMINARIES

A. NOTATIONS

The Euclidean distance is adopted. For convenience, the main notations used in this paper are listed as follows:

- $\boldsymbol{X}$: the training set of nd-dimensional instances.
- $\boldsymbol{S}$: the optimized subset.
- $l(\boldsymbol{x})$: the class label of sample $\boldsymbol{x}$.
- $\boldsymbol{NaN}(\boldsymbol{x})$: the natural neighbors of sample $\boldsymbol{x}$.
- $\boldsymbol{NN}_r(\boldsymbol{x})$: the r nearest neighbors of sample $\boldsymbol{x}$.
- $\boldsymbol{Nb} = \{time_1, time_2, \ldots time_n\}$: it records the number,

where $time_i$ $(i = 1, 2, \ldots, n)$ or $\boldsymbol{Nb}(\boldsymbol{x}_i)$ is the number of sample $\boldsymbol{x}_\text{i}$ that is considered as the neighbor of other samples.

- λ : the natural neighbor eigenvalue.
- ***Outliers***: the set of outliers.
- $\boldsymbol{LS}(\boldsymbol{x})$: the local set of sample $\boldsymbol{x}$.
- $\boldsymbol{LSC}(\boldsymbol{x})$: the local set cardinality of sample $\boldsymbol{x}$.
- $Harmfulness(\boldsymbol{x})$: the harmfulness of sample $\boldsymbol{x}$.
- $Usefulness(\boldsymbol{x})$: the usefulness of sample $\boldsymbol{x}$.
- $|*|$: $|*|$ indicates the number. For example, $|\boldsymbol{NaN}(\boldsymbol{x})|$ is the instances number of $\boldsymbol{NaN}(\boldsymbol{x})$.

B. NATURAL NEIGHBORS

The natural neighbor [33] is a completely new definition of neighbors without parameters and has been used to solve the problem of the parameter selection for KNN [34], [35]. This definition comes from the understanding of social networks. If two people are true friends, they should treat each other as friends. When all people have a friend, a natural stable structure will be formed in social networks. This situation also happens in data objects.

Definition 1 (Natural Stable Structure): Object $\boldsymbol{x}_i$ is a natural neighbor of object $\boldsymbol{x}_j$ if $\boldsymbol{x}_i$ considers $\boldsymbol{x}_j$ as a neighbor and $\boldsymbol{x}_j$ considers $\boldsymbol{x}_i$ as a neighbor at the same time. If every object has at least one natural neighbor, the data set has formed a relatively stable state, named Natural Stable Structure (NSS)

$$(\forall x_i)(\exists x_j)(r \in n) \wedge (x_i \neq x_j) \rightarrow (x_i \in NN_r(x_j)) \wedge (x_j \in NN_r(x_i)) \tag{1}$$

In formula (1), the search round r increases from 1 to λ, where λ is the natural neighbor eigenvalue (NaNE). The $NN_r(x)$ is the r nearest neighbors of sample $\boldsymbol{x}$.

Definition 2 (Natural Neighbor Eigenvalue): Natural Neighbor Eigenvalue λ is equal to the search round r, when the NSS has been formed.

$$\lambda = r_{r \in n}\{r|(\forall \boldsymbol{x}_i)(\exists \boldsymbol{x}_j)(r \in n) \wedge (\boldsymbol{x}_i \neq \boldsymbol{x}_j) \rightarrow (\boldsymbol{x}_i \in \boldsymbol{NN}_r(\boldsymbol{x}_j)) \wedge (\boldsymbol{x}_j \in \boldsymbol{NN}_r(\boldsymbol{x}_i))\} \tag{2}$$

According to the descriptions above, the natural neighbor for data objects is described as follows:

Definition 3 (Natural Neighbor): The natural neighbor (NaN) of x_i is defined as follows:

$$x_j \in NaN(x_i) \Leftrightarrow x_i \in NN_\lambda(x_j) \tag{3}$$

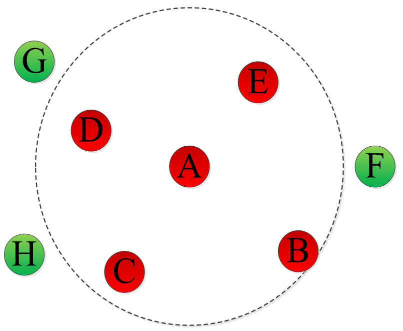

FIGURE 1. **A LS in a two-dimensional space.**

In fact, samples with a higher density have more NaNs, while samples with a lower density have fewer NaNs. Especially, Outliers have no natural neighbors.

Definition 4 (Outliers): Sample $\boldsymbol{x}$ belongs to outliers if sample $\boldsymbol{x}$ has no natural neighbors.

$$\boldsymbol{x} \in \boldsymbol{Outliers} \Leftrightarrow |NaN(\boldsymbol{x})| == 0 \tag{4}$$

In general, the natural neighbor is parameter-free because it has an adaptive value of k (i.e., λ). Besides, we also find some outliers by the natural neighbor in Definition 4. A more detailed description can be seen in existing work [22], [27]. Next, we define the concept of local sets with natural neighbors.

IV. PROPOSED ALGORITHM

A. LOCAL SETS WITH NATURAL NEIGHBORS

The earliest definition for the local set of an instance $\boldsymbol{x}$ is the set of samples whose distance to $\boldsymbol{x}$ is smaller than the distance between $\boldsymbol{x}$ and its nearest neighbor from a different class [30]. However, the process of calculating the local sets [30] is relatively time-consuming. In section, we define a new concept for local sets by fast searching for natural neighbors.

Definition 5 (Local Sets): Local set (LS) of $\boldsymbol{x}_i$ is defined as follows:

$$\boldsymbol{LS}(\boldsymbol{x}_i) = \boldsymbol{NaN}(\boldsymbol{x}_i) \cup \boldsymbol{x}_i \tag{5}$$

In definition 5, $\boldsymbol{LS}(\boldsymbol{x}_i)$ represents the local neighborhood consisting of sample $\boldsymbol{x}_i$ and $\boldsymbol{x}_i$'s natural neighbors. Each local neighborhood (i.e., LS) describes the local characteristics of data sets. Next, we describe another definition named the local sets cardinality.

Definition 6 (Local Sets Cardinality): Local set cardinality (LSC) of $\boldsymbol{x}_i$ is the number of instances contained in $\boldsymbol{LS}(\boldsymbol{x}_i)$. The LSC can be defined as follows:

$$\boldsymbol{LSC}(\boldsymbol{x}_i) = |\boldsymbol{LS}(\boldsymbol{x}_i)| \tag{6}$$

Fig. 1 gives an example to show the LS with 8 instances in a two-dimensional space, where $\boldsymbol{NaN}(\mathbf{A}) = \{\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}\}$, $\boldsymbol{LS}(\mathbf{A}) = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}\}$ and $\boldsymbol{LSC}(\mathbf{A}) = 5$. In fact, there are distinctive characteristics and advantages of our LS.

We first summarize the characteristics of our LS:

(a) For $\forall \boldsymbol{x}$, $\boldsymbol{LSC}(\boldsymbol{x}) \geq 1$.

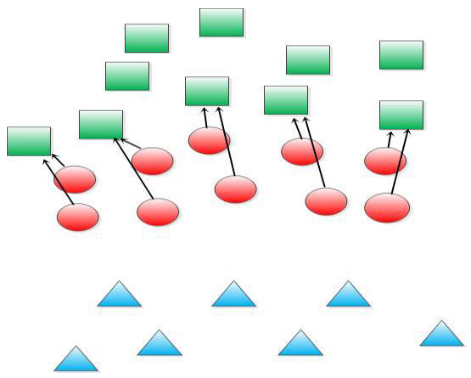

FIGURE 2. A toy example on a complex data set, where enemy neighbors of red circles are green squares.

(b) In a data set, the LSC of different samples may be different. If the LSC of a sample is relatively large, the sample is likely to be in a high-density region, and vice versa. The reason is that samples with higher density have more natural neighbors.

(c) The $Max(\ LSC\ (\boldsymbol{x} \in \boldsymbol{X})) = \lambda + 1$. In other words, the largest LSC of all local sets on a data set is $\lambda + 1$. Especially note that the λ is not an external parameter. It is an adaptive and dynamic value [33].

(d) Our LS is parameter-free because the natural neighbor contained in our local sets does not require user-defined parameters.

Next, we illustrate the advantages of our local set by giving an in-depth comparison between our LS and the one of the work [30].

(a) Our LS can use LSC to describe local densities of data sets and find several global outliers because the LS of a global outlier usually contains only itself. However, the local set of the work [30] does not have the ability.

(b) The computational time of our LS is faster than the one of the work [30]. In detail, the time complexity of calculating our LS is $O(n\log n)$, while the time complexity of the work [30] is $O(n^2)$.

(c) Our LS is more suitable for complex data sets than the one of the work [30]. Fig. 2 uses a toy example to illustrate, where there are 3 classes of the green square, the red circle and the blue triangle. In Fig. 2, all circles' neighbors from the different classes are the square (the arrow in Fig. 2 points to enemy neighbors), which leads to troubles in using the local set of the work [30] to filter out noises and select border samples [31]. In contrast, our LS is based on more reasonable strategies with natural neighbors. Also, the natural neighbor has been effectively used in complex data sets (e.g. data sets with some variations in densities or manifold distribution [34]).

The more detailed algorithm for calculating LS is described in algorithm 1.

Algorithm 1 returns ***LS*** and ***LSC***. The time complexity of algorithm 2 is $O(n\log n)$ because *kd* tree [36] is applied. The steps 1-8 are used to search for the NaN and calculate LS and LSC. The stopping conditions of algorithm 1 is implied in

Algorithm 1 Calculate Local Sets (*CLS*)

```
Input: $X$
Output: $\boldsymbol{LS}$, $\boldsymbol{LSC}$
1:  $r = 1$;
2:  Create a kd tree $T$ from data set $\boldsymbol{X}$;
3:  $\forall \boldsymbol{x}_i \in \boldsymbol{X}, \boldsymbol{LS}(\boldsymbol{x}_i) = \{\boldsymbol{x}_i\}, \boldsymbol{LSC}(\boldsymbol{x}_i) = 1; \boldsymbol{Nb}(\boldsymbol{x}_i) = 0$;
4:  for each sample $\boldsymbol{x}_i$ in $\boldsymbol{X}$, find its $r$-th neighbor $\boldsymbol{x}_j$
     by using $T$
5:     $\boldsymbol{Nb}(\boldsymbol{x}j) = \boldsymbol{Nb}(\boldsymbol{x}j) + 1$;
6:     $\boldsymbol{LS}(\boldsymbol{x}_j) = \boldsymbol{LS}(x_j) \cup \{\boldsymbol{x}_i\}$;
7:     $\boldsymbol{LSC}(\boldsymbol{x}_j) = \boldsymbol{LSC}(x_j) + 1$;
8:  end for
9:  Compute $Num$% $Num$ is the number of sample $\boldsymbol{x}_i$ that
     $\boldsymbol{Nb}(\boldsymbol{x}_i) == 0$;
10:  If $Num$ does not change for 2 times
11:     return$\boldsymbol{LS}$, $\boldsymbol{LSC}$;
12:  else
13:     $r = r+1$;
14:     goto step 4;
15:  end if
```

formula (1) and calculated in steps 9-15. In addition, noises can also affect the number of iterations of Algorithm 1. So we add another condition that *Num* does not change for 2 times.

In general, the LS with natural neighbors offers a novel fast way to describe the neighborhood of each sample, which makes it an effective tool for some tasks related to data mining, such as the edition methods of IR.

B. PROPOSED EDITION METHOD

In this section, we will introduce a fast parameter-free edition method based on our LS. Our inspiration comes from the understanding of the local neighborhood. Normal instances should have the same class label as their local neighborhoods;. By contrast, noisy instances should have a different class label from their local neighborhoods. Based on this motivation, we designed the ELS by defining the harmfulness and usefulness of each sample in Definitions 7-8, where $\boldsymbol{LS}(\boldsymbol{y})$ represents a local neighborhood and $\boldsymbol{l}(\boldsymbol{y})$ represents the class label of the local neighborhood.

Definition 7 (Harmfulness of $\boldsymbol{x}_i$) : The harmfulness of sample $\boldsymbol{x}_i$ can be defined as follows:

$$Harmfu\ln ess(\boldsymbol{x}_i) = |\{\boldsymbol{y}|\boldsymbol{x}_i \in \boldsymbol{LS}(\boldsymbol{y})\ \&\ \& l(\boldsymbol{y}) \neq l(\boldsymbol{x}_i)\}| \quad (7)$$

Definition 8 (Usefulness of $\boldsymbol{x}_i$) : The usefulness of sample $\boldsymbol{x}_i$ can be defined as follows:

$$Usefu\ln ess(\boldsymbol{x}_i) = |\{\boldsymbol{y}|\boldsymbol{x}_i \in \boldsymbol{LS}(\boldsymbol{y})\ \&\ \& l(\boldsymbol{y}) == l(\boldsymbol{x}_i)\}| \quad (8)$$

According to definitions 7-8, the usefulness of sample $\boldsymbol{x}$ is the number of instances y, where $\boldsymbol{LS}(\boldsymbol{y})$ (i.e., a local neighborhood) contains $\boldsymbol{x}$ and $\boldsymbol{y}$'s label (i.e., the label of the local neighborhood) is the same as that of sample $\boldsymbol{x}$. By contrast, the harmfulness of sample $\boldsymbol{x}$ is the number of instances $\boldsymbol{y}$, where $\boldsymbol{LS}(\boldsymbol{y})$ contains $\boldsymbol{x}$ and $\boldsymbol{y}$'s label is different from that

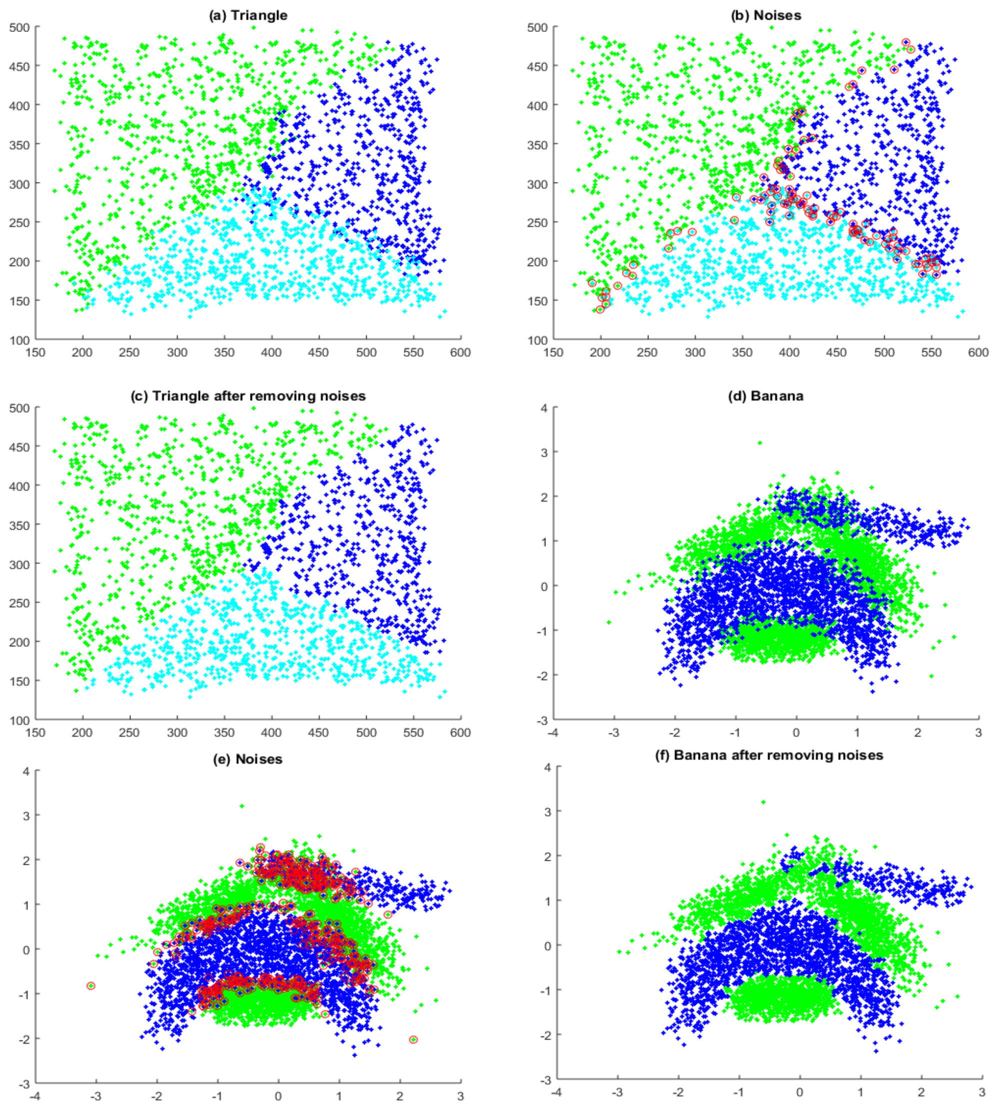

FIGURE 3. ELS on artificial data sets.

of sample $\boldsymbol{x}$. If a sample is a noisy instance, its harmfulness is naturally greater than its usefulness.

Definition 9 (Noisy Instances): If sample $\boldsymbol{x}_i$ is a noisy instance, it should satisfy the following condition:

$$Harmfulness(\boldsymbol{x}_i) > Usefulness(\boldsymbol{x}_i) \tag{9}$$

Especially, if the LSC of sample $\boldsymbol{x}$ is 1, we regard this sample $\boldsymbol{x}$ as an outlier because the LS of an outlier usually contains only itself. The definition of outliers can also be defined by our LS and is described as follows:

Definition 10 (Outliers): If sample x is an outlier, it should satisfy the following condition:

$$\boldsymbol{Outliers} = \{x|\boldsymbol{LSC}(x) == 1\} \tag{10}$$

According to formulas (7)-(10), we can fast filter out noisy instances and global outliers without any parameters. Our proposed edition method is described in Algorithm 2.

Algorithm 2 returns $\boldsymbol{ES}$ without harmful instances and global outliers. In detail, step 2 is used to calculate LS and LSC for $\boldsymbol{X}$. In steps 3-8, some samples whose LSC are 1 or 'Harmfulness' > 'Usefulness' are removed. The time complexity of step 2 is $O(n\log n)$. The time complexity of steps 3-8 is $O(n)$. As a result, the time complexity of ELS is $O(n\log n)$.

C. ILLUSTRATIVE EXAMPLES

In this section, we use 2 two-dimensional artificial data sets of Triangle and Banana to visually demonstrate the efficiency of our algorithm in filtering out noisy instances. The Triangle has 2340 instances and 3 classes. By contrast, the Banana is

Algorithm 2 Edition Method Based on Local Sets (*ELS*)

```
Input: $X$
Output: $ES$ (Edited Set)
1:  $ES = \emptyset$;
2:  $[LS, LSC] = CLS(X)$;
3:  For each sample $x$ in $X$, calculating $Usefulness(x)$
    and $Harmfulness(x)$, according the $LS$;
4:  for all $x_i$ in $X$, $LSC(x_i) > 1$
5:      if $Usefulness\ (x_i) >= Harmfulness(x_i)$
6:          $ES = ES \cup x_i$;
7:      end if
8:  end for
```

an artificial data set with 8800 instances and 2 classes. More details are described in Fig. 3 (a)-(f). Fig. 3 (a)-(c) shows the filtering process on the data set of Triangle while Fig. 3 (e)-(f) show the filtering process on the data set of Banana. Fig. 3 (a) and Fig. 3 (d) show the original data of Triangle and Banana. In Fig. 3 (b) and Fig. 3 (e), noisy instances are detected and surrounded by red circles. As can be seen from Fig. 3 (c) and Fig. 3 (f), noisy instances and several outliers are removed. After that, the decision boundaries are reasonably expanded and made clearer.

V. COMPARISON EXPERIMENTS

We carry out experiments on a server with an Inter (R) Xeon (R) Silver 4100 CPU of 2.10 GHz, a 64GB memory and a 64-bit Windows 10 operating system. The code for all algorithms is written with MATLAB2015.

A. EXPERIMENTAL SETTINGS

We select 22 real data sets from the University of California Irvine (UCI) repositories (http://archive.ics.uci.edu/ml/datasets.html). The description of data sets is listed in Table 1. Specifically, challenging computer vision data sets (e.g. LAS, OPT, PEN, GPS, etc.) are used. Note that for data sets with categorical attributions, we need to convert categorical attributions into integer attributions.

The strategy of 10-fold cross-validation is used to determine the final experimental results in terms of accuracy (ACC) and reduction (R) in all experiments. The formulas of ACC and R are described as follows:

$$ACC = \frac{|T_{correct}|}{|T|} * 100\% \tag{11}$$

$$R = \frac{|S|}{|X|} * 100\% \tag{12}$$

In formula (11), $\boldsymbol{T}$ is the test set and $\boldsymbol{T}_{correct}$ is the set of the correctly classified instances from $\boldsymbol{T}$. In formula (12), $\boldsymbol{S}$ is the reduced set while $\boldsymbol{X}$ is the original set. The KNN ($k = 3$) is used to measure the final result in all experiments and is usually used in existing work [21], [27]. In total, we carry out 3 experiments to verify our algorithm and brief introductions are listed as follows:

(a) Experiment One: we compare our algorithm with 5 representative edition methods.

(b) Experiment Two: our algorithm is used as an effective noise filter in a condensation method and a hybrid method of IR.

(c) Experiment Three: we carry out a comparison experiment to compare the average running time.

B. EXPERIMENT ONE: THE COMPARISON BETWEEN OUR ALGORITHM AND REPRESENTATIVE WORK

In Experiment One, we compare our algorithm with some representative work explained as follows:

- The ENN [7] is a classical and representative NN-based edition method. However, the ENN heavily depends on the parameter k. In this experiment, we choose ENN with different k (1, 3 and 5) as the comparison algorithm.
- The HMN [29], a classical and representative graph-based edition method, is adopted, where $k = 1$ and $\varepsilon = 0.1$.
- The LSSm [31] is the earliest edition method based on local sets. We also use the LSSm as a comparison algorithm because of the relatively high performance, the parameter-free advantage and relatively low computational time.
- The ENaN [27] is a parameter-free edition method based on natural neighbors. At the same time, it is relatively fast and relatively new work. So, we choose it as a comparison algorithm.
- Similar to previous work [21], [27], the KNN with $k = 3$ (denoted as 3NN) is also selected as a comparison algorithm.

In terms of ACC, Table 2 shows experimental results on 22 data sets to demonstrate that our algorithm can outperform some representative work in improving 3NN by effectively removing noisy instances. At the same time, Fig. 4 is the Boxplot of ACC and reflects the degree of dispersion of ACC. Especially note that we do not use the R criterion here because the objective of the edition method is to improve ACC.

To assess whether the difference in ACC is significant, a non-parametric two-sided Wilcoxon signed ranks test with 0.05 significance level is used. The symbols $+$, - and $\sim$, in the row labeled ‘‘Wilcoxon signed ranks test’’, show that ELS is significantly better, worse or equivalent compared with the comparison method, respectively.

As shown in Table 2, ELS achieves the highest ACC for 14 of 22 data sets. Note that ELS does not have a considerable ACC on WAF, YEA, PID, WDBC, IRIS, etc. The possible reason is that comparison methods may better describe the characteristics of noisy instances on some specific distribution.

These average results of ACC for all edition algorithms are higher than that of 3NN. Table 2 also shows that if we set different k in ENN, ENN will have different performances. Compared to ENN, ELS has a more stable performance because it is parameter-free. Moreover, the average results of

TABLE 1. **The description of data sets.**

Index	Data sets	Examples	Attributes	Classes	Data Types	Abbreviations
1	Abalone	4177	8	3	Categorical, Integer, Real	ABA
2	Mammographic Masses	961	5	2	Integer	MAM
3	Indian Liver Patient Dataset	583	10	2	Integer, Real	ILPD
4	Contraceptive Method Choice	1473	9	3	Categorical, Integer	CMC
5	WaveForm	5000	21	3	Real	WAF
6	Haberman's Survival	306	3	2	Integer	HAS
7	Z-Alizadeh Sani	303	55	2	Integer, Real	ZAS
8	Yeast	1484	8	10	Real	YEA
9	Pima Indians Diabetes	768	8	2	Integer, Real	PID
10	IRIS	150	4	3	Real	IRIS
11	Seeds	210	7	3	Real	SEE
12	Wisconsin Diagnostic Breast Cancer	569	30	2	Real	WDBC
13	Wilt	4339	5	2	Real	WILT
14	Australian Credit Approval	690	14	2	Categorical, Integer, Real	ACA
15	Biodegradation	1055	41	2	Integer, Real	BIO
16	Spambase	4601	57	2	Integer, Real	SPA
17	Website Phishing	1353	9	3	Integer	WEP
18	Wireless Indoor Localization	2000	7	4	Real	WIL
19	Landsat Satellite	6435	36	6	Integer	LAS
20	Optdigits	5620	64	10	Integer	OPT
21	Pendigits	10992	16	10	Integer	PEN
22	Gesture Phase Segmentation	9901	19	5	Real	GPS

TABLE 2. **Comparing ELS with representative editions methods in terms of ACC (%).**

Data sets	3NN	ENN (*k*=1)	ENN (*k*=3)	ENN (*k*=5)	HMN-E	LSSm	ENaN	ELS
ABA	52.81	53.56	52.50	52.57	53.34	53.46	53.39	**54.16**
MAM	78.56	78.04	78.88	79.19	79.02	79.40	79.71	**79.92**
ILPD	65.71	64.83	66.21	65.70	64.72	66.05	65.36	**67.41**
CMC	49.22	52.21	51.74	53.02	54.10	49.16	53.84	**54.59**
WAF	81.60	82.08	82.88	83.16	82.80	82.54	**83.58**	83.26
HAS	68.96	71.23	72.23	74.48	72.56	70.57	73.84	**75.16**
YEA	56.07	57.41	58.96	**59.10**	57.42	56.94	57.82	58.62
ZAS	67.00	65.34	68.68	68.67	65.21	67.68	68.95	**69.63**
PID	69.28	70.84	72.28	72.40	71.49	70.98	**73.84**	72.14
IRIS	96.00	**97.33**	**97.33**	**97.33**	96.67	96.00	96.00	**97.33**
SEE	88.57	88.10	89.52	88.57	89.95	88.10	88.57	**90.00**
WDBC	92.09	93.14	93.15	93.15	92.80	92.62	**93.85**	93.32
WILT	99.33	99.26	99.24	99.12	99.24	99.31	99.31	**99.55**
ACA	68.40	69.27	71.01	70.57	69.97	69.27	69.27	**71.72**
BIO	82.07	81.69	82.17	81.70	82.01	**82.92**	81.88	81.47
SPA	81.30	80.40	80.31	79.66	79.60	**81.70**	78.94	81.20
WEP	88.10	81.69	86.91	86.62	85.11	**88.25**	86.10	87.82
WIL	98.15	98.25	98.15	98.15	98.15	98.25	98.20	**98.55**
LAS	90.07	89.81	89.65	89.50	88.28	90.29	89.59	**90.44**
OPT	98.70	98.72	98.61	98.61	98.54	98.90	98.61	**98.94**
PEN	99.41	99.36	99.32	99.36	98.82	99.45	**99.55**	99.51
GPS	95.46	95.91	95.66	95.15	95.25	95.45	95.66	**96.16**
Wilcoxon signed ranks test	+	+	+	+	+	+	+	N/A
Average	80.31	80.38	81.15	81.17	80.68	80.79	81.18	**81.86**

ELS are higher than those of ENN with different k. Compared with other 3 parameter-free edition methods (i.e., HMN-E, LSSm and ENaN), ELS also has a higher average result of ACC. We believe the reason is that the local sets in our ELS can better describe the local characteristics of data sets and the strategy in ELS for removing noises can better describe characteristics of noisy instances.

Observing the row labeled "Wilcoxon signed ranks test", ELS is shown to be significantly better than others. In Fig. 4, the dispersion degree (i.e., the size of the box implied in 25%-75%) of ELS is relatively lower. This result shows that ELS is more robust to different data sets.

C. EXPERIMENT TWO: THE COMPARISON OF A CONDENSATION METHOD AND A HYBRID METHOD

In this section, we use our algorithm as a step of the noise filter in a condensation method (TRKNN, a novel template reduction approach for the K nearest neighbor method) [17] and a hybrid method (ATISA1, an adaptive threshold-based instance selection algorithm) [18]. We use ELS to replace ENN ($k = 3$) in ATISA1_ELS. By contrast, we combine TRKNN with ELS in TRKNN_ELS. Experimental results are shown in Tables 3-4 and Fig 5. Likewise, the non-parametric two-sided Wilcoxon signed ranks test with 0.05 significant level is used to analyze results.

TABLE 3. Comparison between TRKNN with TRKNN_ELS (%).

Data sets	TRKNN		TRKNN_ELS	
	ACC	R	ACC	R
ABA	50.08	45.76	**50.52**	**93.33**
MAM	71.69	58.63	**76.49**	**85.72**
ILPD	57.96	55.95	**60.21**	**84.71**
CMC	45.55	43.33	**51.26**	**83.13**
WAF	79.90	**49.16**	**83.00**	48.12
HAS	55.22	58.21	**67.94**	**91.07**
YEA	48.44	46.54	**56.40**	**83.93**
ZAS	57.81	49.21	**59.15**	**92.56**
PID	63.15	57.25	**71.49**	**84.82**
IRIS	90.00	**77.41**	**95.33**	74.15
SEE	62.86	82.49	**74.29**	**91.32**
WDBC	83.32	86.51	**85.61**	**95.76**
WILT	**98.25**	93.33	96.80	**99.07**
ACA	60.00	55.94	**63.04**	**89.82**
BIO	73.17	65.60	**75.06**	**80.63**
SPA	72.25	63.42	**74.31**	**93.77**
WEP	82.78	62.87	**82.93**	**74.52**
WIL	97.30	**91.24**	**98.10**	90.50
LAS	89.66	67.31	**89.82**	**83.86**
OPT	98.75	**48.85**	**98.93**	47.93
PEN	98.95	**77.12**	**99.42**	**77.12**
GPS	75.11	91.01	**76.44**	**91.31**
Wilcoxon signed ranks test	+	+	N/A	N/A
Average	73.28	64.87	**76.66**	**83.51**

TABLE 4. Comparison between ATISA1 with ATISA1_ELS (%).

Data sets	ATISA1		ATISA1_ELS	
	ACC	R	ACC	R
ABA	96.51	**85.01**	**98.06**	80.81
MAM	**72.52**	82.45	69.27	**84.29**
ILPD	**67.22**	81.09	66.12	**84.85**
CMC	**99.10**	95.60	95.14	**99.04**
WAF	62.00	**89.26**	**95.33**	74.59
HAS	51.27	79.41	**52.27**	**81.96**
YEA	**52.19**	80.19	46.47	**92.86**
ZAS	65.04	**66.12**	**80.70**	**91.41**
PID	**85.24**	91.38	81.90	**92.95**
IRIS	81.56	**63.85**	**83.24**	59.63
SEE	**93.49**	93.81	90.86	**95.94**
WDBC	**77.62**	84.93	76.26	**85.22**
WILT	72.55	86.85	**74.56**	**94.19**
ACA	55.73	78.45	**56.88**	**84.61**
BIO	67.86	82.27	**68.01**	**90.68**
SPA	73.34	79.95	**74.07**	**93.87**
WEP	83.45	74.64	**84.33**	**86.98**
WIL	97.40	**90.95**	**97.95**	90.06
LAS	90.29	84.01	**90.45**	**84.22**
OPT	96.09	**89.30**	**97.69**	85.44
PEN	98.14	87.94	**98.64**	**88.25**
GPS	65.30	86.21	**67.42**	**90.85**
Wilcoxon signed ranks test	+	+	N/A	N/A
Average	75.23	82.57	**77.3**	**86.89**

Table 3 shows an experimental result of the comparison between TRKNN and TRKNN_ELS. In general, TRKNN_ELS has higher average results of ACC and R than TRKNN. In detail, TRKNN only has higher ACC than TRKNN_ELS for 1 of 22 data sets. In terms of R, TRKNN only has a higher R than TRKNN_ELS for 4 of 22 data sets.

Table 4 shows an experimental result of the comparison between ATISA1 and ATISA1_ELS. Likewise, ATISA1_ELS have higher average results of ACC and R than ATISA1. By taking a closer look, ATISA1_ELS has higher ACC than ATISA1 for 15 of 22 data sets. By contrast, ATISA1_ELS has a higher R than ATISA1 for 17 of 22 data sets.

TABLE 5. The comparison about running time for algorithms (seconds).

Data sets	ENN	HMN-E	LSSm	ENaN	ELS
ABA	396.23 (5)	18.76 (3)	40.71 (4)	0.97 (2)	0.70 (1)
MAM	43.35 (5)	4.74 (3)	6.60 (4)	0.22 (2)	0.20 (1)
ILPD	20.45 (5)	2.50 (3)	3.20 (4)	0.12 (2)	0.09 (1)
CMC	150.54 (5)	7.35 (3)	14.33 (4)	0.47 (2)	0.45 (1)
WAF	672.21 (5)	36.82 (3)	83.80 (4)	1.17 (2)	1.14 (1)
HAS	6.20 (5)	0.56 (3)	1.45 (4)	0.06 (2)	0.03 (1)
YEA	6.27 (5)	0.71 (3)	1.78 (4)	0.11 (2)	0.06 (1)
ZAS	95.76 (5)	8.42 (3)	15.85 (4)	0.51 (2)	0.44 (1)
PID	37.15 (5)	1.40 (3)	4.81 (4)	0.20 (2)	0.16 (1)
IRIS	2.26 (5)	0.46 (3)	0.79 (4)	0.06 (2)	0.03 (1)
SEE	2.92 (5)	0.49 (3)	0.93 (4)	0.05 (2)	0.02 (1)
WDBC	22.97 (5)	2.28 (3)	4.13 (4)	0.14 (2)	0.12 (1)
WILT	164.80 (5)	19.58 (3)	25.90 (4)	0.79 (2)	0.77 (1)
ACA	49.20 (5)	1.48 (3)	4.75 (4)	0.23 (2)	0.22 (1)
BIO	84.52 (5)	4.69 (3)	11.52 (4)	0.38 (2)	0.40 (1)
SPA	788.90 (5)	78.26 (3)	134.92 (4)	1.69 (2)	1.67 (1)
WEP	238.43 (5)	3.98 (3)	5.00 (4)	0.14 (1.5)	0.14 (1.5)
WIL	1091.12 (5)	210.73 (3)	329.38 (4)	5.70 (2)	5.22 (1)
LAS	973.12 (5)	132.13 (3)	242.86 (4)	2.28 (2)	2.19 (1)
OPT	1432.88 (5)	234.98 (3)	358.503 (4)	1.644 (2)	1.626 (1)
PEN	2104.56 (5)	378.12 (3)	561.73 (4)	3.55 (2)	3.22 (1)
GPS	1933.44 (5)	224.19 (3)	324.40 (4)	2.78 (2)	2.76 (1)

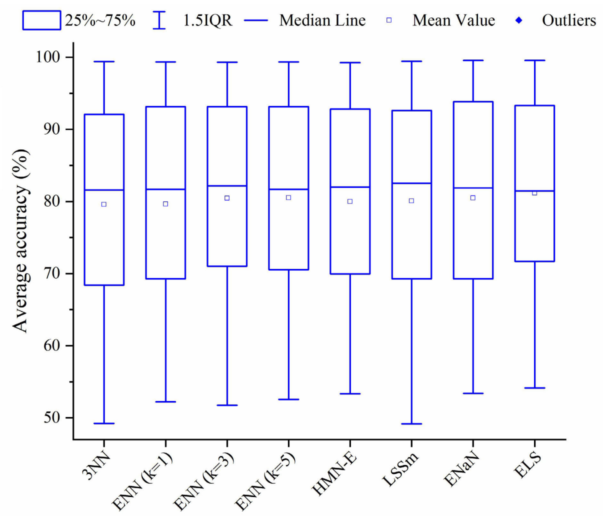

FIGURE 4. Boxplot of average accuracy.

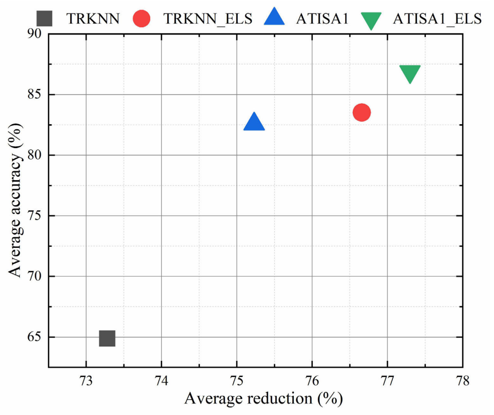

FIGURE 5. The average result of ACC and R for 4 algorithms.

In addition, Observing the row labeled "Wilcoxon signed ranks test", TRKNN_ELS and ATISA1_ELS is shown to be significantly better than TRKNN and ATISA1 in terms of ACC and R.

Also, Fig. 5 shows average results in terms of ACC and R for TRKNN, TRKNN_ELS, ATISA1 and ATISA1_ELS. The top right corner is the best possible solution. In Fig. 5. the ATISA1_ELS is better than others. TRKNN_ELS and ATISA1 have a similar performance. Most important of all, compared to TRKNN and ATISA1, performances (i.e., ACC and R) of TRKNN_ELS and ATISA1_ELS are significantly improved by using ELS.

D. EXPERIMENT THREE: REQUIRED RUNNING TIME

We discuss the time complexity and carry out an experiment on real data sets to compare the average running time between ELS and 4 representative work listed in Experiment One, where all algorithms run 10 times. According to our knowledge, the time complexity of ENN, HMN-E, LSSm, ENaN and ELS is $O(n^3)$, $O(n^2)$, $O(n^2)$, $O(n\log n)$ and $O(n\log n)$ respectively. Theoretically, ELS and ENaN are very fast. Table 5 shows a result to verify and analyze the conclusion in more detail.

We can find from Table 5 that ELS is significantly fast than ENN, HMN-E and LSSm. In addition, ELS is slightly faster than ENaN. Although the difference in terms of running time between ENaN and ELS is very small, ELS is actually better than ENaN by weighing computational time and prediction accuracy.

VI. CONCLUSIONS

Most of existing edition methods still have some drawbacks, such as the parameter dependency, high computational time and relatively low accuracy. To solve these problems, we propose a new fast parameter-free edition method based on local sets (ELS). In ELS, we introduce natural neighbors to define the new local set. Compared to the existing local set [30], our local set has faster computational time and better describes the local characteristics. Through ELS, we can keep more reasonable class boundaries, while effectively filtering out noisy instances and global outliers. The main advantages of ELS are that (a) it is parameter-free; (b) it can remove global outliers and noisy instances; (d) it is relatively fast.

In our experiments, 22 data sets are used and 4 representative algorithms are compared with the proposed one. Through experiments, we verify that our algorithm, compared to representative edition methods, can better remove harmful instances and improve the KNN. Moreover, our algorithm is also relatively fast. In addition, we have found that our algorithm can be effectively used as a step of the noise filter in condensation methods and hybrid methods.

We will apply our local set and ELS in more fields, such as more condensation methods [37], more hybrid methods [38], SMOTE algorithms [39], [40] and semi-supervised self-labeled methods [41]–[43]. Also, we have planned to improve ELS to deal with big data sets.

REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[2] S. Mehta, X. Shen, J. Gou, and D. Niu, "A new nearest centroid neighbor classifier based on k local means using harmonic mean distance," *Information*, vol. 9, no. 9, pp. 234–250, Sep. 2018.

[3] S. Mehta, B.-S. Zhan, and X.-J. Shen, "Weighted neighborhood preserving ensemble embedding," *Electronics*, vol. 8, no. 2, pp. 219–237, Feb. 2019.

[4] X. Sun, L. Liu, C. Geng, and S. Yang, "Fast data reduction with granulation-based instances importance labeling," *IEEE Access*, vol. 7, pp. 33587–33597, 2019.

[5] S. Guo, R. Chen, M. Wei, H. Li, and Y. Liu, "Ensemble data reduction techniques and multi-RSMOTE via fuzzy integral for bug report classification," *IEEE Access*, vol. 6, pp. 45934–45950, 2018.

[6] F. Dornaika and I. Kamal Aldine, "Decremental sparse modeling representative selection for prototype selection," *Pattern Recognit.*, vol. 48, no. 11, pp. 3714–3727, Nov. 2015.

[7] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.

[8] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour, "An algorithm for a selective nearest neighbor decision rule (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 6, pp. 665–669, Nov. 1975.

[9] C.-H. Chou, B.-H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as a data reduction method," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2006, pp. 556–559.

[10] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, Jan. 1991.

[11] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, Jun. 2000.

[12] J. Hamidzadeh, R. Monsefi, and H. Sadoghi Yazdi, "IRAHC: Instance reduction algorithm using hyperrectangle clustering," *Pattern Recognit.*, vol. 48, no. 5, pp. 1878–1889, May 2015.

[13] N. García-Pedrajas and A. de Haro-García, "Boosting instance selection algorithms," *Knowl.-Based Syst.*, vol. 67, pp. 342–360, Sep. 2014.

[14] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.

[15] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 6, pp. 448–452, Jun. 1976.

[16] K. Hattori and M. Takahashi, "A new edited k-nearest neighbor rule in the pattern classification problem," *Pattern Recognit.*, vol. 33, no. 3, pp. 521–528, Mar. 2000.

[17] H. A. Fayed and A. F. Atiya, "A novel template reduction approach for the K-nearest neighbor method," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 890–896, May 2009.

[18] G. D. C. Cavalcanti, T. I. Ren, and C. L. Pereira, "ATISA: Adaptive threshold-based instance selection algorithm," *Expert Syst. Appl.*, vol. 40, no. 17, pp. 6894–6900, Dec. 2013.

[19] J. Li and Y. Wang, "A new fast reduction technique based on binary nearest neighbor tree," *Neurocomputing*, vol. 149, pp. 1647–1657, Feb. 2015.

[20] J. R. Rico-Juan and J. M. Iñesta, "Adaptive training set reduction for nearest neighbor classification," *Neurocomputing*, vol. 138, pp. 316–324, Aug. 2014.

[21] P. Hernandez-Leal, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. A. Olvera-Lopez, "InstanceRank based on borders for instance selection," *Pattern Recognit.*, vol. 46, no. 1, pp. 365–375, Jan. 2013.

[22] L. Yang, Q. Zhu, J. Huang, D. Cheng, Q. Wu, and X. Hong, "Natural neighborhood graph-based instance reduction algorithm without parameters," *Appl. Soft Comput.*, vol. 70, pp. 279–287, Sep. 2018.

[23] J. S. Sánchez, F. Pla, and F. J. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs," *Pattern Recognit. Lett.*, vol. 18, no. 6, pp. 507–513, Jun. 1997.

[24] J. Sanchez, "Analysis of new techniques to obtain quality training sets," *Pattern Recognit. Lett.*, vol. 24, no. 7, pp. 1015–1022, Apr. 2003.

[25] F. Vázquez, J. Sánchez, and F. Pla, "A stochastic approach to Wilson's editing algorithm," in *Proc. 2nd Iberian Conf. Pattern Recognit. Image Anal.*, Jan. 2005, pp. 35–42.

[26] F. J. Ferri, J. V. Albert, and E. Vidal, "Considerations about sample-size sensitivity of a family of edited nearest-neighbor rules," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 5, pp. 667–672, Nov. 1999.

[27] L. Yang, Q. Zhu, J. Huang, and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing*, vol. 230, pp. 427–433, Mar. 2017.

[28] F. Muhlenbach, S. Lallich, and D. Zighed, "Identifying and handling mislabelled instances," *J. Intell. Inf. Syst.*, vol. 39, pp. 89–109, Jan. 2004.

[29] E. Marchiori, "Hit miss networks with applications to instance selection," *J. Mach. Learn. Res.*, vol. 9, pp. 997–1017, Jun. 2008.

[30] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining Knowl. Discovery*, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[31] E. Leyva, A. González, and R. Pérez, "Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective," *Pattern Recognit.*, vol. 48, no. 4, pp. 1523–1537, Apr. 2015.

[32] R. Xia, C. Zong, X. Hu, and E. Cambria, "Feature ensemble plus sample selection: Domain adaptation for sentiment classification," *IEEE Intell. Syst.*, vol. 28, no. 3, pp. 10–18, May 2013.

[33] Q. Zhu, J. Feng, and J. Huang, "Natural neighbor: A self-adaptive neighborhood method without parameter k," *Pattern Recognit. Lett.*, vol. 80, no. 1, pp. 30–36, Sep. 2016.

[34] D. Cheng, Q. Zhu, J. Huang, L. Yang, and Q. Wu, "Natural neighbor-based clustering algorithm with local representatives," *Knowl.-Based Syst.*, vol. 123, pp. 238–253, May 2017.

[35] J. Huang, Q. Zhu, L. Yang, and J. Feng, "A non-parameter outlier detection algorithm based on natural neighbor," *Knowl.-Based Syst.*, vol. 92, pp. 71–77, Jan. 2016.

[36] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[37] Y. Caises, A. González, E. Leyva, and R. Pérez, "Combining instance selection methods based on data characterization: An approach to increase their effectiveness," *Inf. Sci.*, vol. 181, no. 20, pp. 4780–4798, Oct. 2011.

[38] K. Nikolaidis, J. Y. Goulermas, and Q. H. Wu, "A class boundary preserving algorithm for data condensation," *Pattern Recognit.*, vol. 44, no. 3, pp. 704–715, Mar. 2011.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[40] M. R. Prusty, T. Jayanthi, and K. Velusamy, "Weighted-SMOTE: A modification to SMOTE for event classification in sodium cooled fast reactors," *Prog. Nucl. Energy*, vol. 100, pp. 355–364, Sep. 2017.

[41] J. Li and Q. Zhu, "Semi-supervised self-training method based on an optimum-path forest," *IEEE Access*, vol. 7, pp. 36388–36399, 2019.

[42] J. Li, Q. Zhu, and Q. Wu, "A self-training method based on density peaks and an extended parameter-free local noise filter for k nearest neighbor," *Knowl.-Based Syst.*, vol. 184, no. 15, Nov. 2019, Art. no. 104895.

[43] J. Li, Q. Zhu, Q. Wu, and D. Cheng, "An effective framework based on local cores for self-labeled semi-supervised classification," *Knowl.-Based Syst.*, vol. 197, Jun. 2020, Art. no. 105804.