Information Sciences 626 (2023) 293–314

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fast semi-supervised self-training algorithm based on data editing

Bing Li [a,b], Jikui Wang [b,*], Zhengguo Yang [b], Jihai Yi [b], Feiping Nie [c]

[a] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, Guizhou, China
[b] College of Information Engineering, Lanzhou University of Finance and Economics, Lanzhou 730020, Gansu, China
[c] School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, Shanxi, PR China

ARTICLE INFO

Article history:
Received 7 November 2022
Received in revised form 11 December 2022
Accepted 2 January 2023
Available online 6 January 2023

Keywords:
Semi-supervised learning
Self-training
classification
Ball-*k*-means
Data editing

ABSTRACT

Self-training is a commonly semi-supervised learning Algorithm framework. How to select the high-confidence samples is a crucial step for algorithms based on self-training framework. To alleviate the impact of noise data, researchers have proposed many data editing methods to improve the selection quality of high-confidence samples. However, the state-of-the-art data editing methods have high time complexity, which is not less than $O(n^2)$, where n denotes the number of samples. To improve the training speed while ensuring the quality of the selected high-confidence samples, inspired by Ball-*k*-means algorithm, we propose a fast semi-supervised self-training Algorithm based on data editing (EBSA), which defines ball-cluster partition and editing to improve the quality of high-confidence samples. The time complexity of the proposed EBSA is $O\left(t\left(2kn + n\log n + n + k^2\right)\right)$, where k denotes the number of centers, t denotes the number of iterates. k is far less than n, EBSA has linear time complexity with respect to n. A large number of experiments on 20 benchmark data sets have been carried out and the experimental results show that the proposed Algorithm not only ran faster, but also obtained better classification performance compared with the comparison algorithms.

1. Introduction

Supervised classification methods need to train the classifier on the labeled data set. However, obtaining the labels for all the data is a time-consuming and expensive job, especially in the era of big data. In practice, it is found that only a small part of the data is labeled, which provides a promising application scenario for semi-supervised learning. Semi-supervised learning can use the information of unlabeled data to improve the performance of the classifier because it trains the classifier using labeled data and unlabeled data simultaneously.

Researchers have proposed semi-supervised learning methods such as self-training [1–4], collaborative training [5–7], and generative models [8,9]. Recently, researchers have proposed some semi-supervised learning algorithms in the direction of deep learning[10–12]. [10] enables the model to overcome the interference of pseudo-labels when using unlabeled data and taking the features applicable to the model as the research focus. [11] enhances the ability of classification models to

* Corresponding author.
E-mail addresses: 376581763@qq.com (B. Li), wjkweb@163.com (J. Wang), yangzg@lzufe.edu.cn (Z. Yang), yijihai@163.com (J. Yi), feipingnie@gmail.com (F. Nie).

https://doi.org/10.1016/j.ins.2023.01.029
0020-0255/

distinguish accurate data from fake data by introducing task-conditioned adversarial generators. [12] uses the sparse representation to evaluate whether pairs of pixels belong to the same class and construct a probability matrix to solve the problem of sample label scarcity.

Asemi-supervised learning Algorithm framework termed self-training is commonly used [13–16]. Self-training iteratively trains the classifier until the Algorithm converges. The performance of self-training relies heavily on the selected high-confidence samples. Once the selected high-confidence sample points contain noise points or abnormal points, they will always exist in the iterative training, which will degrade the performance of the learned classifier. Therefore, how to select the high-confidence points is very crucial. .

To deal with the problem, researchers have proposed many methods. One approach is using data editing techniques to improve the quality of high-confidence sample selection. [17] proposed an Algorithm named SNNRCE, which builds a relative tangent neighborhood graph based on the nearest neighbor rule, and calculates tangent weights. Samples without trimming are selected as high-confidence samples, and trimming weights are used to limit the number of samples in each class to avoid extreme outputs. [18] proposed the MLSTE algorithm, which combines ENN (Edited Nearest Neighbor) data editing technology to evaluate whether a sample point has high confidence by searching the label distribution of the k nearest neighbors of the sample point. [19] proposed a semi-supervised classification Algorithm based on density peaks of data and differential evolution termed STDP-DE, which uses differential evolution [20] to edit and optimize the attribute values of "previous" and "next" unlabeled data with labeled data in the self-training process, and takes the edited datas as high-confidence samples.

There are other ways to select high-confidence samples, such as: changing the distance metric, combining with semi-supervised FCM [21]. [2] proposed the SDTC algorithm, which uses the base classifier to classify the labeled data set into positive and negative classes, and then uses the Mahalanobis distance [22] to calculate the distance between each unlabeled sample and the mean value of the positive p and negative samples q. The $w = |p - q|$ is used as the sample score, and the first n samples with the highest scores are selected as high-confidence samples.

The Ball-k-means[23] Algorithm first selects the ball-cluster centers randomly, and then divides the samples into "stable regions" and "active regions". During each iteration, only the distances between the centers and the distances between the centers and the samples in "active regions" need to be calculated, which reduces the amount of calculation and reduces the time complexity of the algorithm.

Existing methods for selecting high-confidence samples have high time complexity. In order to reduce the time complexity of the semi-supervised algorithm, inspired by the Ball-k-means algorithm, this paper proposes a fast semi-supervised self-training Algorithm based on data editing termed EBSA. The main characteristics of the proposed EBSA Algorithm are as follows:

(1) We propose a novel method termed as ball-cluster partitioning to divide the data set into the stable and the disputed regions.
(2) The ball-cluster partition and editing Algorithm is proposed to improve the quality of high-confidence samples through evaluates whether the samples in the stable region are mislabeled or not, and then edits the found mislabeled sample points. As a result, the proposed method has the ability to deal with noise.
(3) Since only the distances between the samples and the centers of the ball-clusters need to be calculated in each iteration, the proposed EBSA has lower time complexity compared with other semi-supervised learning algorithms based on self-training.
(4) Furthermore, EBSA is non-parameter and the character makes this method have a wide application prospect. The experimental results validate that the EBSA Algorithm not only has a low time complexity but also improves the performance of the learned classifier compared with SETRED, STDPCEW, STDPNaN and STDPNF on 20 benchmark data sets.

The other parts of this article are described as follows: in the second part, some related algorithms are introduced. The third part describes the proposed Algorithm EBSA in detail. The fourth part is the experimental setting. In the fifth part, we conducted a large number of comprehensive experiments to verify the performance of the proposed Algorithm EBSA. In the last part, we summarized the paper and discuss some research aspects in the future.

2. Related work

This section lists the important symbols and formulas that appear in this paper, and introduces some basic theoretical knowledge and comparative algorithms.

2.1. Symbols and description

The important symbols and descriptions involved in this article are as follows:

- $X = \{x_1, x_2, x_3, \cdots, x_n\}$ denotes the data set containing n samples, and $x_i \in \Re^{d\times 1}$ denotes one sample.
- $Y = \{y_1, y_2, y_3, \cdots, y_K\}$ denotes the label set with K possible classes. y_i represents the i-th label.

- $L = \{(x_1, f_1), (x_2, f_2), \cdots, (x_l, f_l)\}$ denotes the labeled samples set.
- $U = \{x_1, x_2, \cdots, x_u\}$ represents the unlabeled samples set.

2.2. Self-training

Semi-supervised learning is performed by combining information from unlabeled and labeled data. Self-training[24] is one of the typical semi-supervised learning framework. First, a base classifier is trained on the labeled data set, and then the high-confidence samples are selected from the unlabeled set and added to the labeled set for iterative training. The self-training Algorithm is described in Algorithm 1.

2.3. SETRED

SETRED [25] identifies mislabeled sample points from self-labeled data using a specific data editing method to eliminate the effect of mislabeled sample points (noise points) during the iterative training. SETRED calls the edges connected between points with different class labels tangent edges by constructing the related adjacency graph. CEW (Cut Edge Weight) is added to each iteration of self-training to evaluate whether the newly labeled sample points have high-confidence or not, and then only the samples with high-confidence are added to the labeled data set L. The optimized classifier H is obtained iteratively.

Algorithm 1: Self-training Algorithm

Input Labeled set L, unlabeled set U
Output Classifier H
1. Initialize high-confidence set $S = \varnothing$
2. WHILE $U \neq \varnothing$DO
3. Train the classifier H on the labeled set L
4. Use classifier H to assign labels to unlabeled set U
5. Select some samples from the unlabeled set U to form a high-confidence set S, and give pseudo-labels by H
6. Update $L \leftarrow L \cup S$, $U \leftarrow U - S$
7. End Whlie
8. When classifier H is stable or $U \neq \varnothing$, output classifier H

2.4. STDPNF

Recently, a self-training method based on density peaks termed STDPNF [26] has been proposed. STDPNF proposes a new local noise filter ENaNE that removes noise using both labeled and unlabeled data, which overcomes the technical shortcomings of the local noise filter in existed self-training methods. STDPNF redefines the spatial structure of the data found by the density peak clustering Algorithm (DPC) [27], uses the self-training method to label unlabeled samples and extend labeled data, and adds the filtered samples to the training set L to obtain the classifier H.

2.5. STDPCEW

STDPCEW [28] discovers the underlying spatial structure of the data set using the density peak clustering Algorithm (DPC) to find the "previous" sample set L' and "next" sample set L'' of the labeled data set L. The "previous" and "next" unlabeled samples of all the labeled samples in L are labeled, and the "previous" and "next" samples are evaluated with high-confidence using hypothesis testing with cut-edge weights, and finally the samples with high-confidence are added to the labeled data set L, and the optimized classifier H is iteratively obtained.

2.6. STDPNaN

STDPNaN [29] uses an integrated classifier to improve the label prediction capability of the self-training algorithm. STDPNaN proposes a parameter-free density peak clustering Algorithm DPCNaN by introducing natural nearest neighbors. DPCNaN discovers the spatial structure of the entire data set by making each sample point to its nearest sample with higher local density. STDPNaN labels the "next" and "previous" unlabeled samples of all labeled samples in L on the basis of the data space constructed by DPCNaN, adds the labeled sample points to the set of labeled samples L, and then iterates to derive the optimized classifier H.

2.7. Summary

In this section, we summarize the advantages and disadvantages of the relevant algorithms, as shown in Table 1. Let n be the number of samples and t denotes the number of iterations. As can be seen from Table 1, these state-of-the-art algorithms have a common disadvantage, that is, the time complexity of the algorithms is very high.

Table 1
Advantages and disadvantages of the relevant algorithms.

Algorithms	Advantages	Disadvantages
SETRED	Improves the selection quality of high-confidence samples with trim weights. The effect of noise samples can be reduced.	Sensitive to unbalanced data, high time complexity $O(tn^3)$.
STDPCEW	Uses the statistically identifying cutoff weights to identify incorrectly labeled samples. And solves the problem that the samples are incorrectly labeled.	The cut edge proximity graph has a great influence on the algorithm. The Algorithm has high time complexity and is $O(tn^3)$.
STDPNaN	Parameter-free, suitable for working with spherical and non-spherical data sets.	High time complexity $O(tn^2)$.
STDPNF	Proposes a new local noise filter to overcome the technical defects of the existing local noise filters, and improves the classification accuracy of KNN.	Not available for multi-label classification tasks. The time complexity is $O(tn^2)$, which is high.

3. Fast semi-supervised self-training Algorithm based on data editing (EBSA)

The existing semi-supervised self-training algorithms, such as SNNRCE, MLSTE, SETRED, STDPCEW, STDPNaN and STDPNF have high computational complexity. Therefore, improving the selection speed of high-confidence samples in self-training is an important research topic. Currently, [23] proposed a novel and fast clustering Algorithm termed Ball-*k*-means, which only calculates the distances between the samples in the active region and its center in each iteration. This method greatly reduces the algorithm's time complexity and improves the clustering's speed. Inspired by this idea, to reduce the computational complexity of the semi-supervised learning algorithms based on self-training framework, this paper proposes a fast semi-supervised self-training Algorithm based on data editing (EBSA). In each iteration, EBSA divides the data set into the stable and the disputed regions and then evaluates whether the samples in the stable region are mislabeled. EBSA edits the found mislabeled sample points. Since only the distances between the samples and the centers need to be calculated in each iteration, the proposed method inherits the fast property of Ball-*k*-means.

3.1. Definitions

Definition 1. (Cluster center and cluster radius[23]) Let C' denote a cluster, and x_i is the i-th sample in C'. $|C'|$ represents the number of samples in C'. Let ς represent the cluster center, and r is the cluster radius.

$$\varsigma = \frac{1}{|C'|}\sum_{i=1}^{|C'|} x_i \tag{1}$$

$$r = \max\left(\|x_i - \varsigma\|_2\right) \tag{2}$$

Definition 2. The cluster center $\varsigma$is calculated by Eq. (1), and cluster radius r is the maximum distance between the sample x_i and the cluster center ς.

Definition 3. (Ball-cluster) The spherical region with cluster center ςas the center of the sphere and cluster radius r as the radius is called a ball-cluster.

Definition 4. (Ball-cluster label) Let C_p denote the p-th ball-cluster, $p \in [1, K]$. K indicates the number of ball-cluster. The label of ball-cluster C_p is the label with the largest number of samples within ball-cluster C_p.

$$k_{C_p} = \arg\max_i |F_{pi}| \tag{3}$$

$$l_{C_p} = y_{k_{C_p}} \tag{4}$$

where F_{pi} is the set with label y_i in ball-cluster C_p. l_{C_p} is the label of ball-clusters C_p. k_{C_p} is the label of the most numerous class labels in the ball cluster C_p.

Definition 5. (Neighbor of ball-cluster[23]) Let C_p and C_q denote two different ball-clusters, ς_p and ς_q are the centers of the two ball-clusters respectively, and r_p denotes the radius of the ball-cluster C_p.

The distance d_{pq} between C_p and C_q is:

$$d_{pq} = \left\|\varsigma_p - \varsigma_q\right\|_2 \tag{5}$$

When d_{pq} satisfies the following formula (6), C_q is the neighbor of C_p.

$$\frac{1}{2}d_{pq} < r_p \tag{6}$$

Obviously, the neighbor relationship is asymmetric. Any two ball-clusters C_p and C_q may be related in the following three relationships.

(1) C_p and C_q are neighbor mutually.
(2) C_p is a neighbor cluster of C_q, and C_q is not a neighbor cluster of C_p.
(3) C_p and C_q have no neighbor relationship.

Given four ball-clusters C_1 ,C_2, C_3 and C_4, let $\varsigma_1, \varsigma_2, \varsigma_3$ and ς_4 denote the cluster centers of these four ball clusters, and d_{12}, d_{13}, d_{14} are the distances between the centers of ball-clusters C_1 and C_2, C_1 and C_3, C_1 and C_4, respectively. r_1, r_2, r_3 and r_4 represent the radii of ball-clusters C_1 ,C_2, C_3 and C_4, respectively. The three neighbor relationships among them are shown in Fig. 1:

From Fig. 1, the blue line, the yellow line and the red line are the vertical bisectors. It can be seen that $\frac{1}{2}d_{13} < r_1, \frac{1}{2}d_{13} < r_3$, so C_1 and C_3 are neighbor clusters to each other, $r_2 < \frac{1}{2}d_{12} < r_1, C_2$ is a neighbor cluster of C_1, C_1 is not a neighbor cluster of C_2, and $\frac{1}{2}d_{14} > r_1, \frac{1}{2}d_{14} > r_4$, then C_1 and C_4 have no neighbor relationship.

Definition 6. (Stable area and active area) Let N_{C_p} denote the set of centers of the neighbor clusters of C_p. C_{pw} is the stable area of the ball-cluster C_p.

$$\hat{r} = \frac{1}{2}\min\left(d_{pq}\right)_{\varsigma_q \in N_{C_p}} \tag{7}$$

When the sphere center ς_q in C_q satisfies formula (7), the stable region C_{pw} of C_p is the spherical region formed with ς_p as the cluster center and $\hat{r}$ as the radius. The active area C_{ph} of C_p is $C_p - C_{pw}$.

Definition 7. (Area to be demarcated) Given ball-clusters C_p and C_q, C_q is a neighbor cluster of C_p, ς_p and ς_q respectively represent the centers of the two ball-clusters, and r_p and r_q are the radii of the two ball-clusters respectively.

$$\widetilde{r} = \frac{1}{2}d_{pq} \tag{8}$$

The area to be demarcated C_{pdh} of the ball-cluster C_p is a spherical area with ς_q as the cluster center and $\widetilde{r}$ as the radius.

Definition 8. (Disputed area) The disputed area C_{pzy} of ball-cluster C_p is:

$$C_{pzy} = C_{pdh} \cap C_p \tag{9}$$

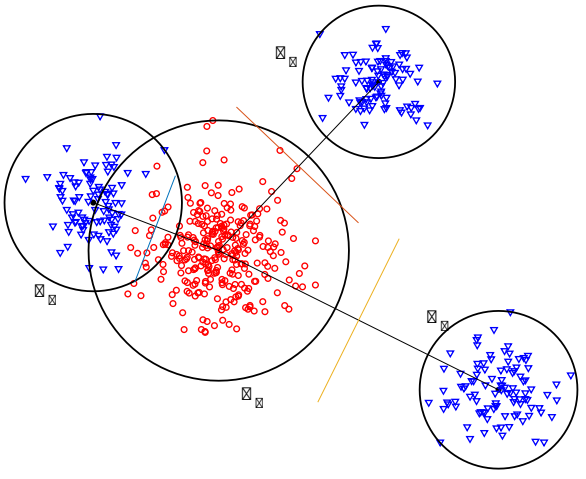


Fig. 1. Neighbors of ball-cluster.

Given ball-clusters C_p and C_q, C_q is a neighbor cluster of C_p, ς_p and ς_q respectively represent the centers of the two ball-clusters, and r_p and r_q are the radii of the two ball-clusters respectively. $\forall x_i \in C_{pzy}, \left\|x_i - \varsigma_p\right\|_2 \leqslant r_p, \left\|x_i - \varsigma_q\right\|_2 \leqslant r_q$. According to the clustering assumptions, point x_i can be classified into the ball-cluster C_p or C_q. Therefore, samples in the disputed area are more likely to be misclassified. This misclassification is amplified during the iterative training of self-training, which leads to a degradation of the classifier performance.

3.2. Ball-cluster partition and editing algorithm

Given the ball-clusters C_p and C_q, C_q is a neighbor cluster of C_p, the disputed area of C_p is C_{pzy}. N_{pq} is the set of neighbor clusters of ball-cluster C_p. N_{pzy} is the set of disputed regions C_{pzy} of ball-cluster C_p. C is the set of all ball-clusters. C_{pw} is the stable area of ball-cluster C_p. The ball-cluster partition and editing Algorithm is summarized in the following Algorithm 2.

Ball-cluster partition and editing Algorithm divides the clusters first and then edits the points in the stable region. Since noisy points are inevitable in the self-training algorithm, we need to edit the samples in the stable region to reduce the influence of mislabeled samples and thus improve the classification performance of the learned classifier.

3.3. High-confidence samples selection algorithm

For self-training algorithm, how to select high-confidence samples is a crucial step. To improve the quality of the selected high-confidence samples, a novel high-confidence sample selection Algorithm is proposed, which first divides the data set into ball-clusters, and then calculates the stable area and disputed area of each ball-cluster, finally selects the high-confidence samples based on data editing of the samples in the ball-clusters. In the Algorithm 3, S represents high-confidence samples set. Algorithm 3 describes the proposed high-confidence samples selection algorithm.

Algorithm 2: Ball-cluster partition and editing algorithm(EC)

Input Data set X, ball-clusters set C
Parameters N_{pq}, C_p
Output C_{pw}, C_{pzy}
1. $C_p = \varnothing, N_{pq} = \varnothing$
2. For each C_p in C
3. Calculate the cluster center ς_p by formula (1)
4. Calculate the ball-cluster radius r_p by formula (2)
5. Calculate the ball-cluster label l_{C_p} by formula (3)
6. End for
7. Calculate the distance d_{pq} between ς_p and ς_q
8. For each C_p in C
9. If $\frac{1}{2} d_{pq} < r_p$
10. $N_{pq} = N_{pq} \cup C_q$
11. End if
12. End for
13. For each C_p in C
14. For each C_q in N_{pq}
15. Calculate C_{pw} using formula (7)
16. Calculate C_{pzy} using formula (9)
17. End for
18. For each x_i in C_{pw}
19. If $y_i \neq l_{C_p}$
20. $y_i == l_{C_p}$
21. End if
22. End for
23. End for
24. Output C_{pw}, C_{pzy}

Algorithm 3: High-confidence samples selection algorithm(*SG*)

Input Ball-clusters C, N_{pzy}// Denotes the samples in C_{pzy}
Parameters S', X_p //All samples with labels contained in ball-cluster C_p
Output High-confidence samples set S
1. $S = \varnothing$
2. For each C_p in C
3. $S' = X_p - N_{pzy}$
4. $S = S \cup S'$
5. End for
6. Output S

The samples in the disputed area will affect the classification performance during the classification process. Therefore, we delete the samples in the disputed area belonging to the cluster C_p, then we select the edited samples of the stable region and the samples of the non-disputed region as high-confidence samples, which can reduce the impact of misclassification and improve the classification performance in the subsequent iterations.

The following Fig. 2 illustrates the selection process of high-confidence samples: in Fig. 2, C_1 , C_2 and C_3 represent ball-clusters, ς_1, ς_2 and ς_3 respectively represent the centers of the three ball-clusters. r_1, r_2 and r_3 are the radii of the three ball-clusters respectively. C_2 and C_3 are the neighbor clusters of the ball-cluster C_1.

In Fig. 2, (a) is the original samples, (b) is the plot after ball-cluster division of (a), and (c) is the plot after ball-cluster editing. The sample points labeled as 1 in (c) are the high-confidence samples of ball-cluster C_1. It can be seen from Fig. 2 that the original spatial sample distribution of sub-graph (a) is divided into sub-graph (b) by ball-cluster partition. In sub-graph (b), rectangle 2 and hexagon 3 are located in the stable area of ball-cluster C_1. According to the ball-cluster partition and editing algorithm, the labels of samples with inconsistent labels in the stable region of ball-cluster C_1 are changed to label 1. In the proposed Algorithm EBSA, we do not need to calculate the distance between two sample points, but only to calculate the cluster centers, and the distances between two cluster centers. We have divided the stable region and the disputed region, and employed SG to select high-confidence samples. Thus the computational complexity is greatly reduced and the performance of the learned classifier is improved at the same time.

In the following Fig. 3a, four different ball-clusters are denoted by C_1 , C_2, C_3 and C_4. Let $\varsigma_1, \varsigma_2, \varsigma_3$ and ς_4 denote the cluster centers of these four ball-clusters, and the radii of these four ball-clusters are denoted by r_1, r_2, r_3 and r_4. C_2 and C_3 are the neighbor clusters of the ball-cluster C_1.

The stable and disputed area of C_1 are shown in Fig. 3a. The stable and disputed regions of the other spherical clusters are divided in the same way as C_1. The disputed area is the area enclosed by the green dotted line and the black solid line, and the area enclosed by the purple dotted line and the black solid line.

The area to be demarcated of C_1 is shown in Fig. 3b. C_2 is a neighbor cluster of C_1. Let ς_1 and ς_2 denote the cluster centers of C_1 and C_2, and the radii of C_1 and C_2 are denoted by r_1, r_2, respectively. The blue line is the vertical bisector of the distance d_{12} between the centers ς_1 and ς_2 of ball-clusters C_1 and C_2. The area within the green dotted line is the area to be demarcated.

EBSA employs the decision tree [30–33] as base classifier to label all unlabeled samples and then divides stable areas and disputed area of each ball-cluster. The samples within each ball-cluster are edited by the proposed ball-cluster division editing Algorithm and the high-confidence samples are selected at the same time. Finally, the optimized classifier is obtained by the self-training framework. The proposed EBSA is summarized in Algorithm 4.

4. Experimental methodology

All the experiments in the paper were conducted with 32G RAM, 64-bit Windows 10 and Inter Core i9 processor. All the codes are implemented with MATLAB 2019b. We use Accuracy and F-score as classification evaluation metrics, which can be calculated from the confusion matrix [34]. The related SETRED, STDPCEW, STDPNaN and STDPNF are selected as the comparison algorithms.

4.1. Data sets

In the experiments, the data sets used in the experiments are all public data sets. The details of these data sets are shown in Table 2.

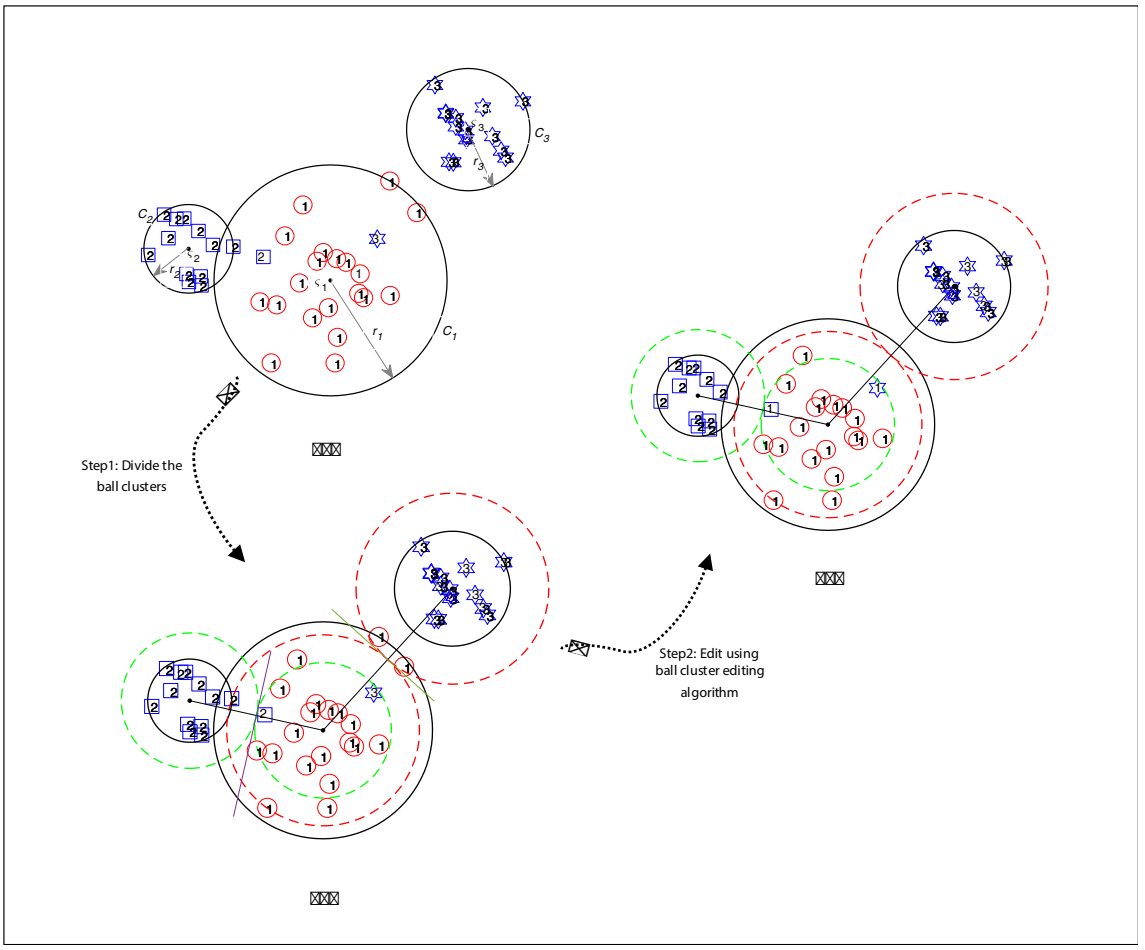
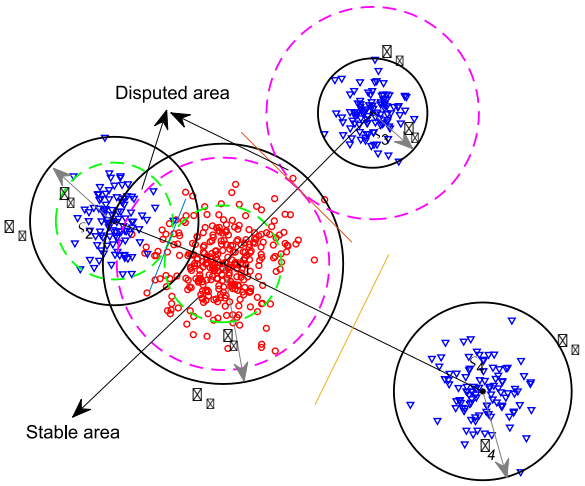


Fig. 2. Selecting the high-confidence samples.

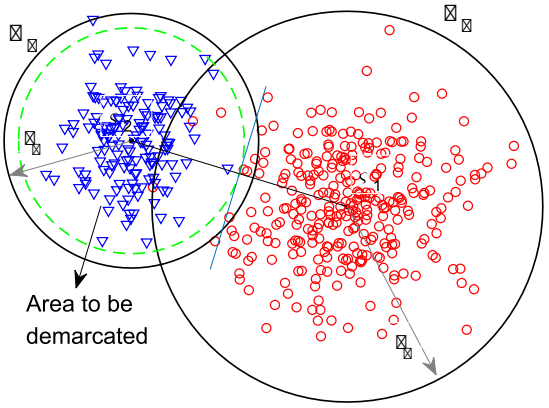
Algorithm 4: EBSA algorithm

InputLabeled sample set L, unlabeled sample set U
Parameters High-confidence sample set S
Output Classifier H

1. $S = \varnothing$
2. Train the classifier H on the labeled set L
3. For each $x_i \in U$
4. $y_i = H(x_i)$
5. $C_{y_i} = C_{y_i} \cup x_i$
6. End for
7. Calculate ball-clusters C
8. While not converges
9. Calculate C_{pw}, N_{pzy} by Algorithm 1
10. Calculate high-confidence samples set S by Algorithm 2
11. Update the labeled sample set $L \leftarrow L \cup S$
12. Train the classifier H on the updated labeled set L
13. End Whlie
14. Output H



(a) Stable area and disputed area



(b) Area to be demarcated

Fig. 3. Ball-cluster partition and editing.

Among the 20 data sets, AR [35], COIL20 [36], MINIST2k [37], ORL,[1] Palm,[2] UPS[38] and YaleB [39] are all image data sets. Isolet [40] is the spoken audio data set, the data set is relatively large, and Solar [3] is the solar flare data set. Sonar is the sonar data set. Cleve, Solar and Sonar data sets are relatively small. The data set Yeast is a data frame of 112 observations of 50 variables: genotype data (genotype states at 12 SNP markers) and phenotype data (normalized and discretized expression values of 38 genes). MSRA25 [41] data set is a face data set consisting of 1799 images from 12 individuals of different genders. FERET32 has 1400 samples that is a subset of FERET [42], which contains 1564 sets of images for a total of 14,126 images. We used three small data sets from UCI database.[4] The BUPA data set is 345 values obtained from observations of 7 variables. Heart is the single proton emission of the heart computed tomography (SPECT). Ecoli data set is a summary of protein location information.

[1] http://www.uk.research.att.com/facedatabase.html.
[2] https://www.gwern.net/Crops.
[3] https://www.kaggle.com.
[4] https://archive.ics.uci.edu/ml/datasets.

Table 2
Data sets details.

index	data set	samples	features	clusters
1	AR	1680	1024	120
2	Cleve	303	13	4
3	COIL20	1440	1024	20
4	Isolet	1560	617	2
5	UPS	2007	256	10
6	Heart	270	13	2
7	MINIST2k	4000	784	10
8	ORL	400	1024	40
9	Palm	2000	256	100
10	Sonar	208	60	2
11	Solar	323	12	6
12	YaleB	2414	1024	38
13	BUPA	345	6	2
14	uspst	2007	256	10
15	MSRA25	1799	256	12
16	Yeast	1484	1470	10
17	Ecoli	336	7	8
18	autouni	205	25	6
19	pendigis	3498	16	10
20	FERET32	1400	1024	200

4.2. Experimental settings

The running parameters of the comparison algorithms are set according to the original articles. Specifically, $\theta = 0.1$ for SETRED, $\alpha = 2$ and $\theta = 0.1$ for STDPCEW. We set $\alpha = 2$ for STDPNF. It is worth mentioning that the proposed EBSA Algorithm does not require hyper-parameters, which makes EBSA easy to be applied.

In order to keep consistent with the comparison algorithms, the decision tree of version C4.5 is chosen as the base classifier. The operating parameters of the decision tree are as follows: the Gini diversity index is used as the division criterion for attribute division, the cluster name used in training and testing is consistent with the label of label set Y, and the priority of each class is set according to the class frequency in label set Y. When the nodes are subdivided, the number of samples of each node is required to be no less than 2. The number of all features is taken as the maximum number of features considered when dividing, and the maximum number of splits is the number of all samples in the data set minus one, that is, $n - 1$.

We conducted the Wilcoxon signed ranks test at the level of confidence of 95%. The symbol "+", "−", and "∼" respectively indicate that the Algorithm EBSA proposed is significantly better, worse or equivalent with the comparison algorithm.

5. Results and discussion

5.1. Classification performance and analysis

In the real world, labeled data often accounts for a relatively small proportion. Some of the state-of-art semi-supervised learning methods use different ratio of labeled samples [43], but most of the papers use 10% labeled samples for experiments. Therefore, we randomly selected 10% of the samples as training set for experiments. In order to avoid the randomness of the experimental results, all experiments in this paper were carried out 50 times. The experimental results are shown in Table 3 and Table 4.

We can draw the following conclusions from the experimental results in Table 3 and Table 4:

(1) The classification accuracies(F-score) of the Algorithm EBSA on most of the data sets are higher than that of the other four comparison algorithms. Obviously, the classification performances of the proposed EBSA are higher than that of SETRED, STDPCEW, STDPNaN and STDPNF.
(2) On the 7 image data sets including AR, COIL20, ORL, Palm, YaleB, FERET32 and MSRA25, the classification performances exceed 90%. Experimental results show that the proposed Algorithm is suitable for high-dimensional image data sets.
(3) It can be seen from the results of statistical tests that the classification performance of EBSA is significantly better than that of SETRED, STDPCEW and STDPNaN, and is almost the same as the STDPNF.
(4) The average classification accuracy standard deviation of the EBSA Algorithm on the 20 data sets is 1.39, which lower than that of SETRED, STDPCEW, STDPNaN and STDPNF and the average F-score got the same result. The experimental results prove the robustness of the proposed EBSA algorithm.

Table 3
Accuracy of each Algorithm on 20 data sets (mean ± std).

Accuracy	EBSA	SETRED	STDPCEW	STDPNF	STDPNaN
AR	**96.23 ± 0.61**	81.58 ± 1.03	78.87 ± 0.95	96.18 ± 0.47	79.50 ± 1.52
Cleve	**86.17 ± 2.13**	79.94 ± 4.28	79.04 ± 5.62	84.38 ± 3.78	79.88 ± 5.64
COIL20	**92.61 ± 1.07**	83.70 ± 1.45	80.97 ± 2.23	90.89 ± 1.42	82.59 ± 1.93
Heart	**74.55 ± 3.81**	72.98 ± 3.82	74.88 ± 3.61	70.36 ± 7.52	69.56 ± 4.49
Isolet	**72.36 ± 1.81**	69.46 ± 3.53	70.01 ± 2.38	71.78 ± 1.88	67.87 ± 2.93
MINIST2k	**85.94 ± 0.62**	78.20 ± 0.95	77.08 ± 1.31	85.91 ± 1.28	77.64 ± 1.01
ORL	**95.58 ± 0.66**	83.19 ± 1.48	80.46 ± 0.76	94.79 ± 0.88	81.30 ± 1.27
Palm	**95.46 ± 0.47**	79.90 ± 1.30	75.98 ± 0.81	95.22 ± 0.55	77.80 ± 1.51
Solar	**86.25 ± 1.54**	81.35 ± 4.01	84.61 ± 2.06	81.30 ± 2.69	81.43 ± 3.20
Sonar	**75.75 ± 2.81**	65.63 ± 9.00	66.36 ± 4.95	73.91 ± 4.93	65.87 ± 4.30
UPS	**88.23 ± 1.39**	82.93 ± 1.46	82.33 ± 1.52	87.82 ± 1.70	82.60 ± 1.74
YaleB	**91.93 ± 1.68**	72.04 ± 1.45	67.51 ± 1.18	91.42 ± 1.93	68.73 ± 1.76
BUPA	**67.41 ± 4.72**	63.70 ± 3.60	61.22 ± 3.34	66.95 ± 3.12	62.71 ± 3.86
FERET32	**97.75 ± 0.50**	93.03 ± 0.18	86.86 ± 0.96	97.67 ± 0.38	88.13 ± 1.06
MSRA25	**90.63 ± 0.87**	87.53 ± 0.91	86.88 ± 1.62	90.40 ± 0.64	86.45 ± 1.18
Yeast	**93.51 ± 0.02**	80.22 ± 1.69	81.51 ± 2.21	93.42 ± 0.15	80.97 ± 2.60
uspst	**88.34 ± 1.03**	83.48 ± 1.65	82.58 ± 1.41	87.76 ± 1.12	82.90 ± 0.95
autouni	**79.91 ± 1.36**	76.67 ± 3.56	74.46 ± 4.63	79.68 ± 2.08	73.19 ± 6.14
Ecoli	**93.03 ± 0.18**	87.87 ± 2.09	88.26 ± 3.36	92.73 ± 0.73	89.77 ± 2.49
pendigits	90.37 ± 0.57	91.53 ± 1.23	91.55 ± 0.74	89.88 ± 0.97	**91.68 ± 0.60**
WSR-test	N/A	+	+	∼	+
Ave.ACC	**87.1**	79.59	78.57	86.12	78.53
Ave.std	1.39	2.48	2.28	1.91	2.51

Table 4
F-score of each Algorithm on 20 data sets (mean ± std).

F-score	EBSA	SETRED	STDPCEW	STDPNF	STDPNaN
AR	92.20 ± 1.86	51.51 ± 3.80	39.85 ± 4.48	**92.27 ± 1.11**	42.90 ± 4.81
Cleve	**75.68 ± 5.88**	68.39 ± 8.23	68.50 ± 9.08	71.23 ± 9.30	68.45 ± 7.36
COIL20	**92.20 ± 1.30**	80.90 ± 2.00	76.98 ± 3.35	90.10 ± 1.76	79.32 ± 2.72
Heart	74.55 ± 3.81	72.98 ± 3.82	**74.88 ± 3.61**	70.36 ± 7.52	69.56 ± 4.49
Isolet	**72.36 ± 1.81**	69.46 ± 3.53	70.01 ± 2.38	71.78 ± 1.88	67.87 ± 2.93
MINIST2k	**85.13 ± 0.74**	73.60 ± 1.40	71.92 ± 1.96	85.08 ± 1.55	72.76 ± 1.49
ORL	**91.44 ± 2.01**	55.19 ± 8.16	42.59 ± 5.31	90.00 ± 2.11	46.39 ± 5.99
Palm	**92.76 ± 0.65**	62.21 ± 1.99	51.82 ± 2.66	91.86 ± 1.18	57.07 ± 4.30
Solar	75.92 ± 7.09	72.82 ± 3.86	**77.63 ± 2.91**	57.87 ± 11.47	72.14 ± 5.85
Sonar	**75.75 ± 2.81**	65.63 ± 9.00	66.36 ± 4.95	73.91 ± 4.93	65.87 ± 4.30
UPS	**87.64 ± 1.64**	80.19 ± 1.97	79.36 ± 2.10	87.25 ± 2.00	79.72 ± 2.35
YaleB	**90.73 ± 2.07**	61.70 ± 2.63	52.22 ± 2.67	90.32 ± 2.40	54.62 ± 3.42
BUPA	**67.41 ± 4.72**	63.70 ± 3.60	61.22 ± 3.34	66.95 ± 3.12	62.71 ± 3.86
FERET32	**91.73 ± 1.99**	89.66 ± 1.37	34.48 ± 3.92	91.63 ± 1.84	45.53 ± 5.45
MSRA25	**90.39 ± 0.98**	86.10 ± 1.13	85.25 ± 2.08	90.14 ± 0.73	84.72 ± 1.50
Yeast	93.00 ± 0.31	73.85 ± 5.18	76.74 ± 2.89	**93.05 ± 0.46**	75.42 ± 4.36
uspst	**87.77 ± 1.21**	80.91 ± 2.21	79.68 ± 1.91	87.19 ± 1.32	80.15 ± 1.29
autouni	**73.79 ± 4.17**	62.52 ± 8.84	59.58 ± 6.77	70.26 ± 8.22	61.69 ± 9.58
Ecoli	89.66 ± 1.37	79.61 ± 4.80	80.22 ± 6.74	**89.69 ± 1.93**	84.75 ± 4.29
pendigits	90.14 ± 0.63	90.91 ± 1.40	90.93 ± 0.85	89.70 ± 1.10	**91.08 ± 0.69**
WSR-test	N/A	+	+	∼	+
Ave.ACC	**84.51**	70.03	67.01	82.53	68.14
Ave.std	2.35	4.24	3.70	3.30	4.05

5.2. Impact of labeled sample ratio on classification performance

In order to estimate the impact of the proportion of labeled samples on the classification performance of the Algorithm on a small number of labeled data sets. We randomly selected the proportion of samples with labels from 2% to 20% with the step of 2%, and conducted experiments on 20 data sets. These experiments are conducted 50 times and the mean and standard deviation of the results were calculated. The experimental results are shown in Fig. 4 and Table 5.

(1) The proposed EBSA obtained higher classification performances on 12 data sets than SETRED, STDPCEW, STDPNaN and STDPNF algorithms, which shows that the proposed Algorithm has better classification performance than the comparison algorithms.
(2) The classification performances decrease on the image data sets from Fig. 4. This is because the base classifier is decision tree. For high-dimensional data, classification performance of decision tree will be affected by the dimensionality.

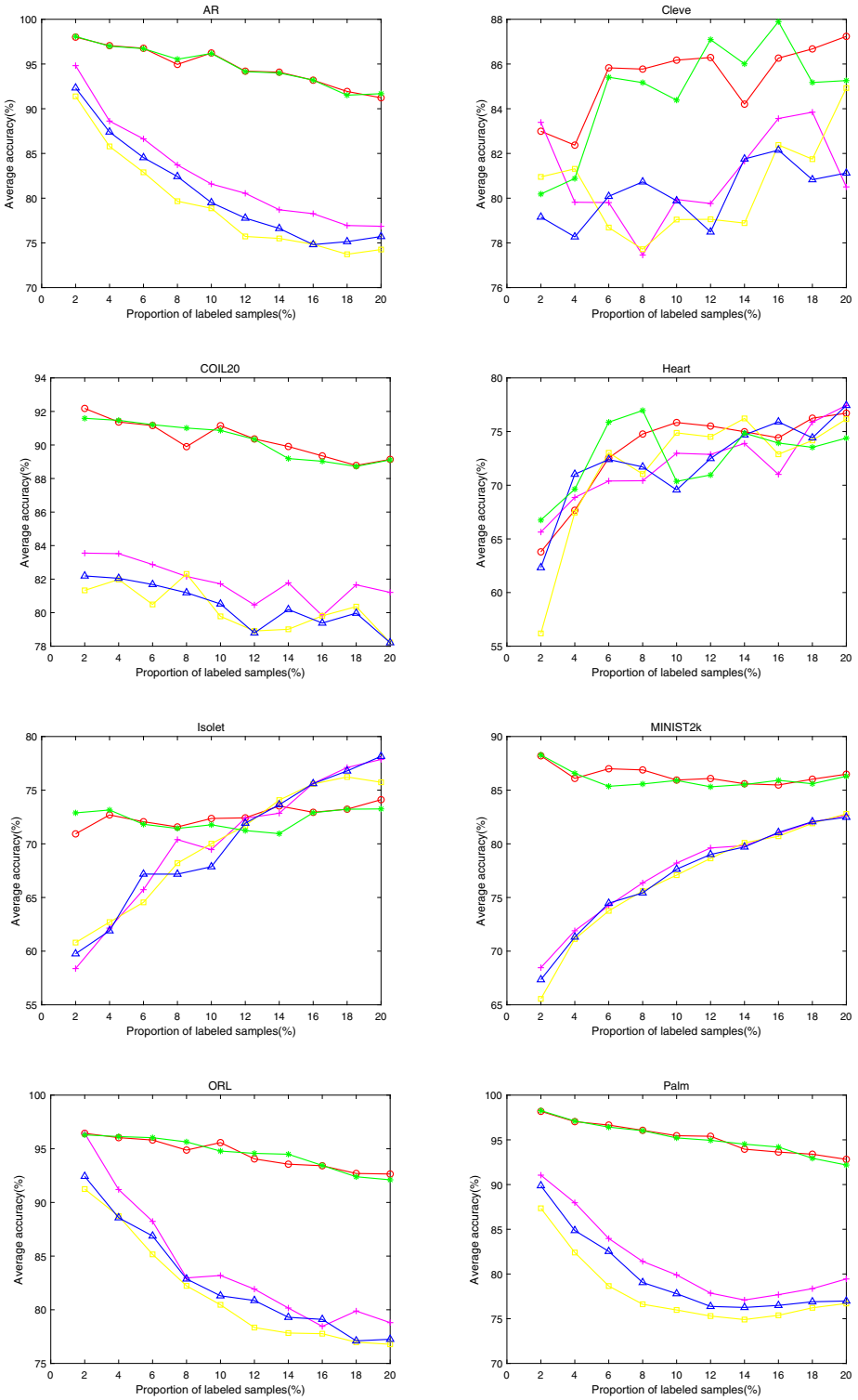


Fig. 4. Classification performance of each Algorithm under different labeled sample ratios.

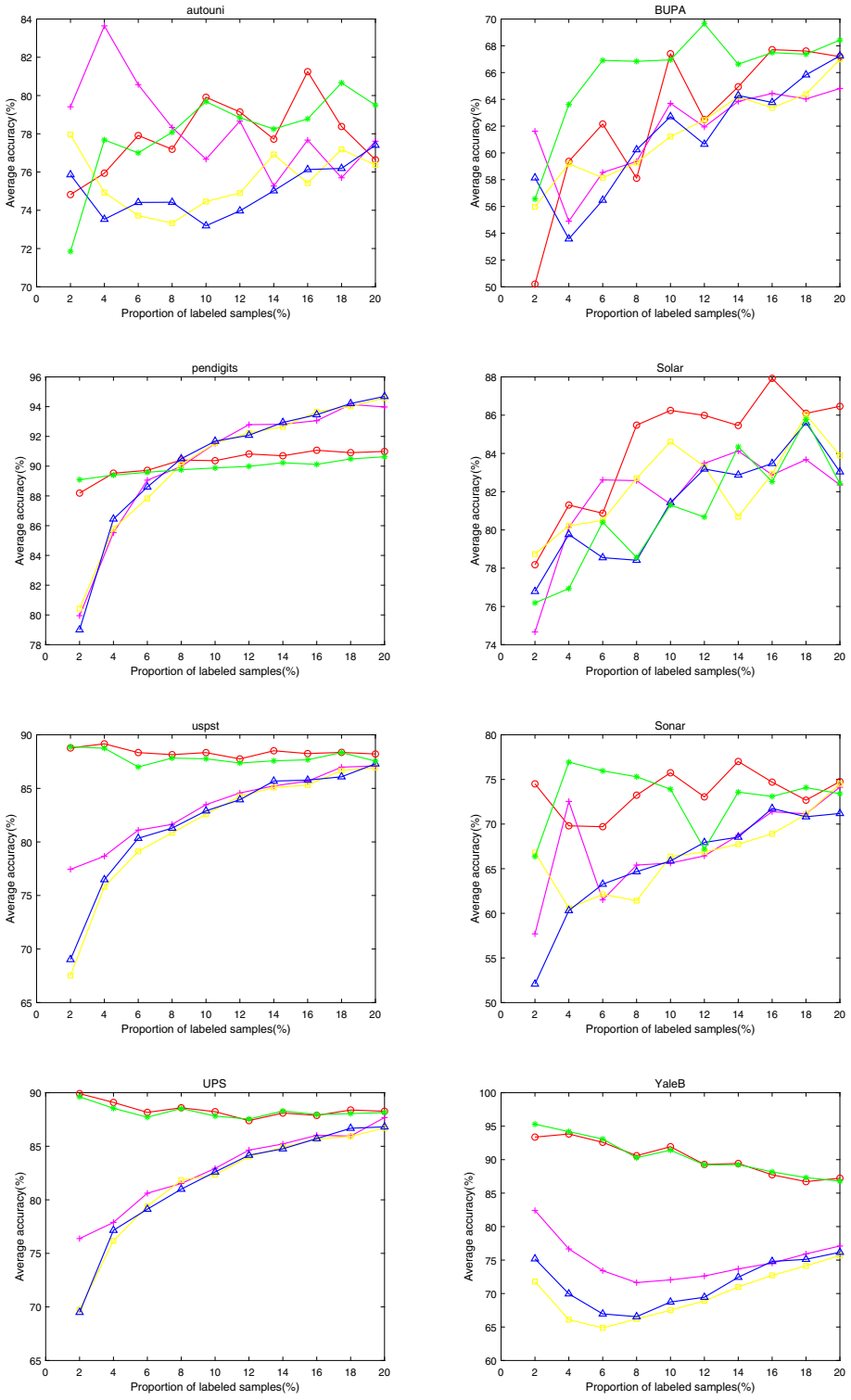


Fig. 4 (*continued*)

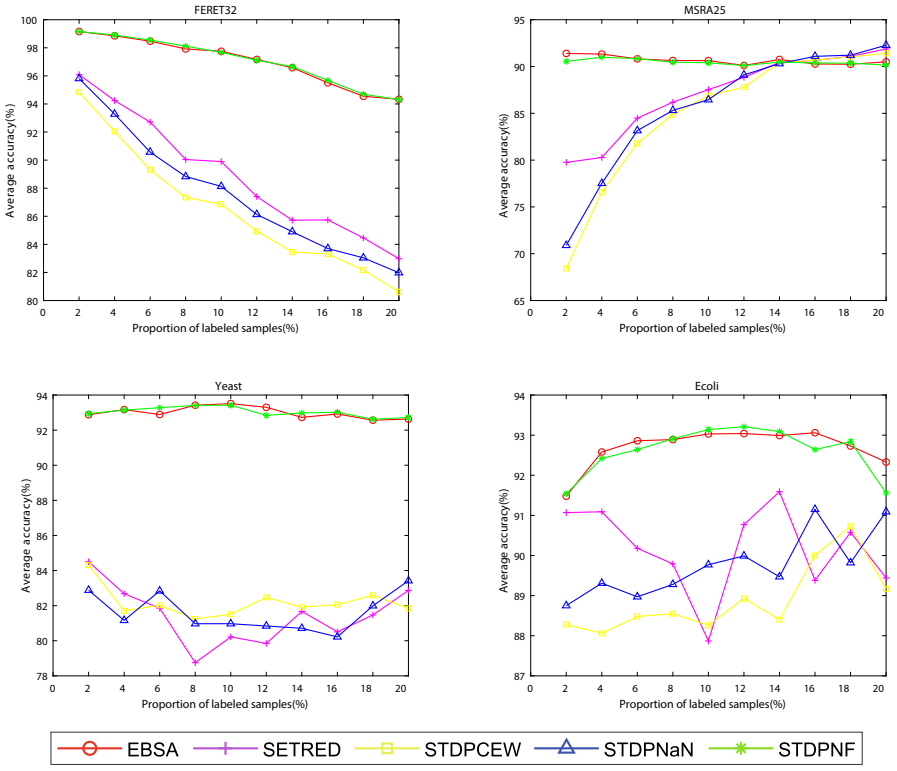


Fig. 4 (*continued*)

Table 5
Accuracy of each Algorithm on 20 data sets (mean ± std).

Accuracy	EBSA	SETRED	STDPCEW	STDPNF	STDPNaN
AR	94.77 ± 2.25	82.67 ± 5.84	79.26 ± 5.82	**94.81 ± 2.26**	80.62 ± 5.92
Cleve	**85.38 ± 1.63**	80.97 ± 2.08	80.47 ± 2.20	84.74 ± 2.45	80.25 ± 1.32
COIL20	**92.45 ± 0.51**	84.98 ± 3.54	82.36 ± 6.19	91.89 ± 1.10	82.87 ± 6.02
Heart	**73.25 ± 4.23**	71.94 ± 3.44	71.65 ± 6.02	72.72 ± 3.18	72.19 ± 4.20
Isolet	**72.59 ± 0.94**	70.19 ± 6.44	69.96 ± 5.71	72.26 ± 0.91	70.00 ± 6.26
MINIST2k	**86.38 ± 0.81**	77.43 ± 4.65	76.73 ± 5.38	86.03 ± 0.89	77.05 ± 4.93
ORL	94.51 ± 1.42	84.13 ± 5.96	81.55 ± 5.19	**94.59 ± 1.52**	82.57 ± 5.14
Palm	**95.26 ± 1.77**	81.47 ± 4.75	77.95 ± 3.97	95.18 ± 1.85	79.71 ± 4.60
Solar	**84.40 ± 3.14**	81.79 ± 2.76	82.36 ± 2.26	80.91 ± 3.06	81.31 ± 2.81
Sonar	**73.52 ± 2.37**	67.46 ± 5.17	66.64 ± 4.38	72.98 ± 3.49	65.64 ± 6.03
UPS	**88.40 ± 0.69**	82.89 ± 3.73	81.67 ± 5.31	88.22 ± 0.59	81.75 ± 5.38
YaleB	90.27 ± 2.59	75.00 ± 3.22	69.89 ± 3.71	**90.51 ± 2.92**	71.53 ± 3.65
BUPA	62.72 ± 5.63	61.72 ± 3.22	61.52 ± 3.37	**66.04 ± 3.66**	61.29 ± 4.31
FERET32	97.03 ± 1.74	88.93 ± 4.40	86.49 ± 4.53	**97.09 ± 1.73**	87.64 ± 4.56
MSRA25	**90.67 ± 0.43**	87.11 ± 4.39	84.99 ± 7.50	90.47 ± 0.27	85.73 ± 6.90
Yeast	93.00 ± 0.33	81.44 ± 1.68	82.17 ± 0.85	**93.04 ± 0.27**	81.60 ± 1.10
autouni	77.89 ± 1.90	**78.35 ± 2.46**	75.52 ± 1.54	78.03 ± 2.41	75.01 ± 1.35
Ecoli	**92.70 ± 0.49**	90.18 ± 1.10	88.89 ± 0.86	92.56 ± 0.58	89.76 ± 0.81
pendigits	90.27 ± 0.90	90.29 ± 4.49	90.28 ± 4.45	89.92 ± 0.48	**90.36 ± 4.74**
uspst	**88.38 ± 0.38**	83.19 ± 3.37	81.41 ± 6.02	87.87 ± 0.60	81.88 ± 5.57
WSR-test	N/A	+	+	∼	+
Ave.ACC	86.19	80.11	78.59	86.00	78.94
Ave.std	1.71	3.83	4.26	1.71	4.28

We conducted experiments on decision tree, and the results show that as the proportion of labeled samples increases, the classification accuracy of decision trees decreases.

(3) The average accuracy of the EBSA Algorithm is 86.19%, which is higher than that of SETRED, STDPCEW, STDPNF and STDPNaN. The average standard deviation of EBSA is 1.71, which is lower than that of SETRED, STDPCEW and STDPNaN, which reflects the robustness of EBSA.

(4) From Fig. 4, the classification performance curves of the EBSA are smooth under different labeled proportions, which shows the robustness of EBSA.

5.3. Noise experiment analysis

EBSA uses ball-cluster division to edit the data in the iterative training process, therefore, it has denoising ability and can effectively correct the mislabeled data. In order to verify the denoising ability of EBSA, noise experiments were conducted. In the real world, labeled data usually accounts for a small proportion of all data. Thus, 10% labeled data were chosen as training data [21,44]. Then, we randomly selected [1%, 10%] data from the training data and assigned error labels to them. Accuracy was chosen as the classification performance evaluation metric. These experiments were conducted 50 times and the mean and standard deviation of the results were calculated. The results of the experiments are shown in Fig. 5 and Table 6.

From Fig. 5 and Table 6, we can draw the following conclusions:

(1) As the proportion of noise samples increases, the classification performance of the proposed Algorithm EBSA is higher than 90% on AR, COIL20, ORL, Palm and FERET32 data sets, which is 17.06%, 10.01%, 12.89%, 19.04% and 10.07% higher than the lowest performance of comparison Algorithm STDPCEW. The classification performance on Cleve, MINIST2k, Solar, UPS, YaleB, MSRA25, Yeast, Ecoli and uspst data sets are all higher than 80%, compared with the lowest STDPCEW increased by 7.4%, 8.9%, 4.57%, 5.88%, 22.48%, 3.76%, 11.07%, 2.29% and 5.27%. And the mean values in Table 6 are consistent with the findings in Fig. 5, which proved the good denoising ability of EBSA.
(2) The average classification accuracy of the EBSA Algorithm is 84.64%. Compared with the comparison algorithms SETRED, STDPCEW, STDPNaN and STDPNF, the performance is improved by 6.51%, 7.92%, 7.58% and 0.37%, respectively. It can be seen that the EBSA Algorithm has data editing capabilities, which can reduce the impact of noisy samples on classification performance.
(3) As can be seen from Fig. 5, the classification performance of the EBSA Algorithm on AR, COIL20, ORL, Palm, UPS, YaleB and FERET32 data sets is not affected by the increase of noise ratio. As the noise ratio increases, the classification performance curve fluctuates in the horizontal direction, which verifies the good data editing ability of the EBSA algorithm.
(4) The classification performances of the proposed Algorithm EBSA is much better than SETRED, STDPCEW and STDPNaN, and slightly better than the STDPNF with local filters on 11 data sets including AR, COIL20, Cleve, Isolet, MINIST2k, ORL, Palm, Solar, Sonar, UPS and YaleB. Theses results demonstrate that the excellent ability of EBSA for noise removal.

5.4. Experiment on different base classifiers

In order to study the influence of different base classifiers on the proposed high-confidence sample selection and editing algorithm(SG), we conducted experiments on random forest, KNN(K = 1), SVM, and the decision tree. Let DT be the decision tree. RF represents the random forest. All experiments were run 50 times, and we recorded the mean and standard deviation of accuracies of the experiments, and listed the running time of each classifier. The experimental results are shown in Tables 7 and 8.

From Table 7 and 8, we can draw the following conclusions:

(1) SG + RF obtained the highest averge classification accuracy, SG + KNN is second only to SG + RF. The classification accuracy of SG + DT is similar to that of KNN, and slightly lower than that of random forest. SG + SVM has the lowest classification accuracy.
(2) The average running time of SG + DT is less than that of SG + KNN, and the running time of SG + SVM is the longest. Because the random forest integrates multiple decision trees, the running time of SG + RF is much higher than that of SG + DT and SG + KNN.
(3) To sum up, SG + DT runs faster than other tree algorithms. Although the classification accuracy of SG + DT is lower than that of SG + RF, but the running time is much less than that of SG + RF.

5.5. Details of iteration process on UPS

The UPS was selected to describe the details of the EBSA Algorithm iteration process.

First of all, we randomly selected 10% of the labeled samples, there are 200 samples in total. The proposed EBSA converged after three iterations. We described the details of each iteration as follows:

(1) After the first iteration, EBSA executed the proposed data editing on 62 samples to obtain 353 high-confidence samples.
(2) After the second iteration, EBSA obtained 374 high-confidence samples with 14 edited samples.
(3) After the third iteration, EBSA obtained 387 high-confidence samples with 4 edited samples.

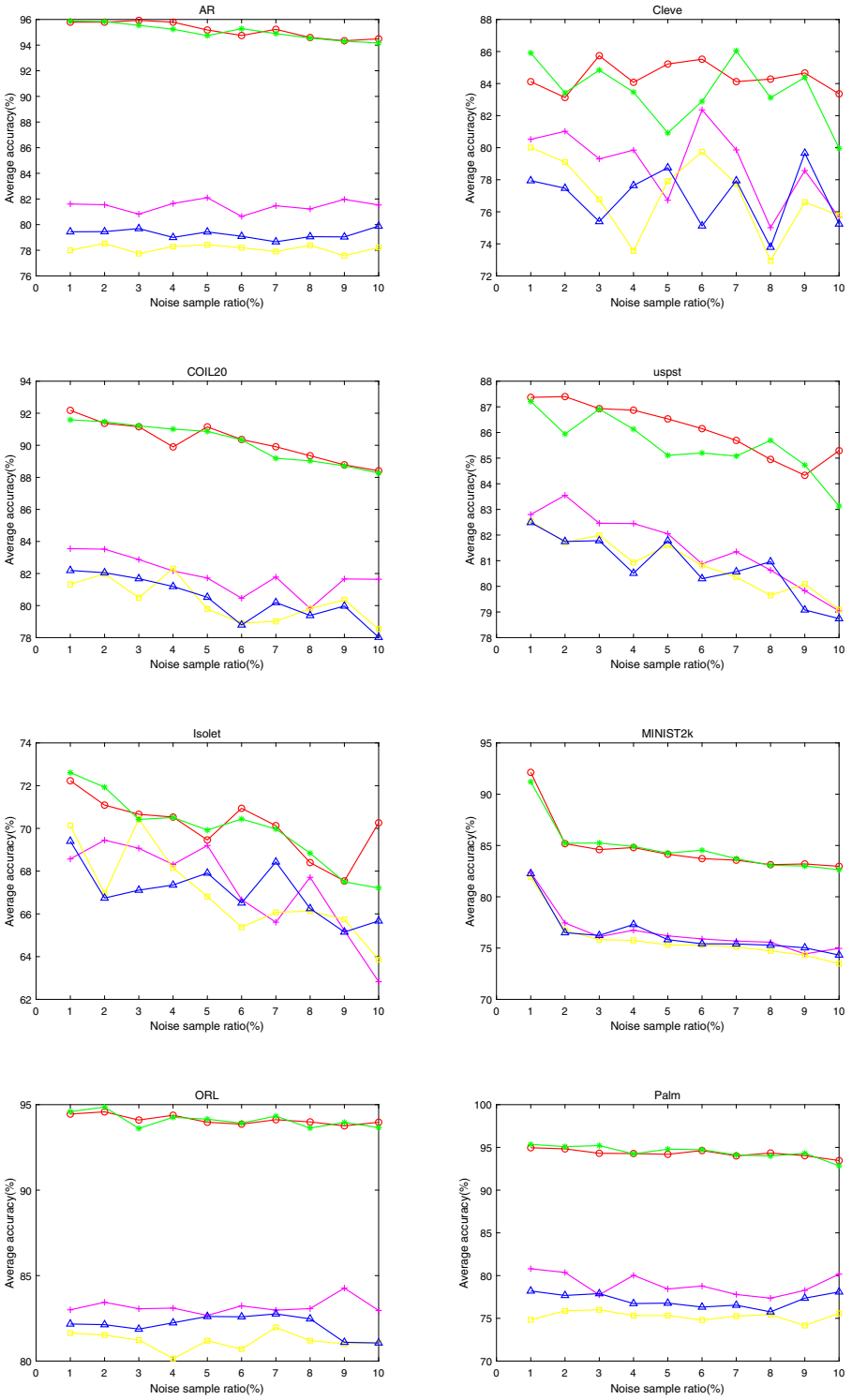


Fig. 5. The influence of noise ratio on each algorithm.

5.6. Runtime analysis

The SNNRCE Algorithm mainly consists of constructing the edge-cut proximity graph, and then uses the nearest neighbor (NN) rule for classification, the time complexity of computing the NN is $O(n^3)$, and SNNRCE's overall Algorithm complexity is

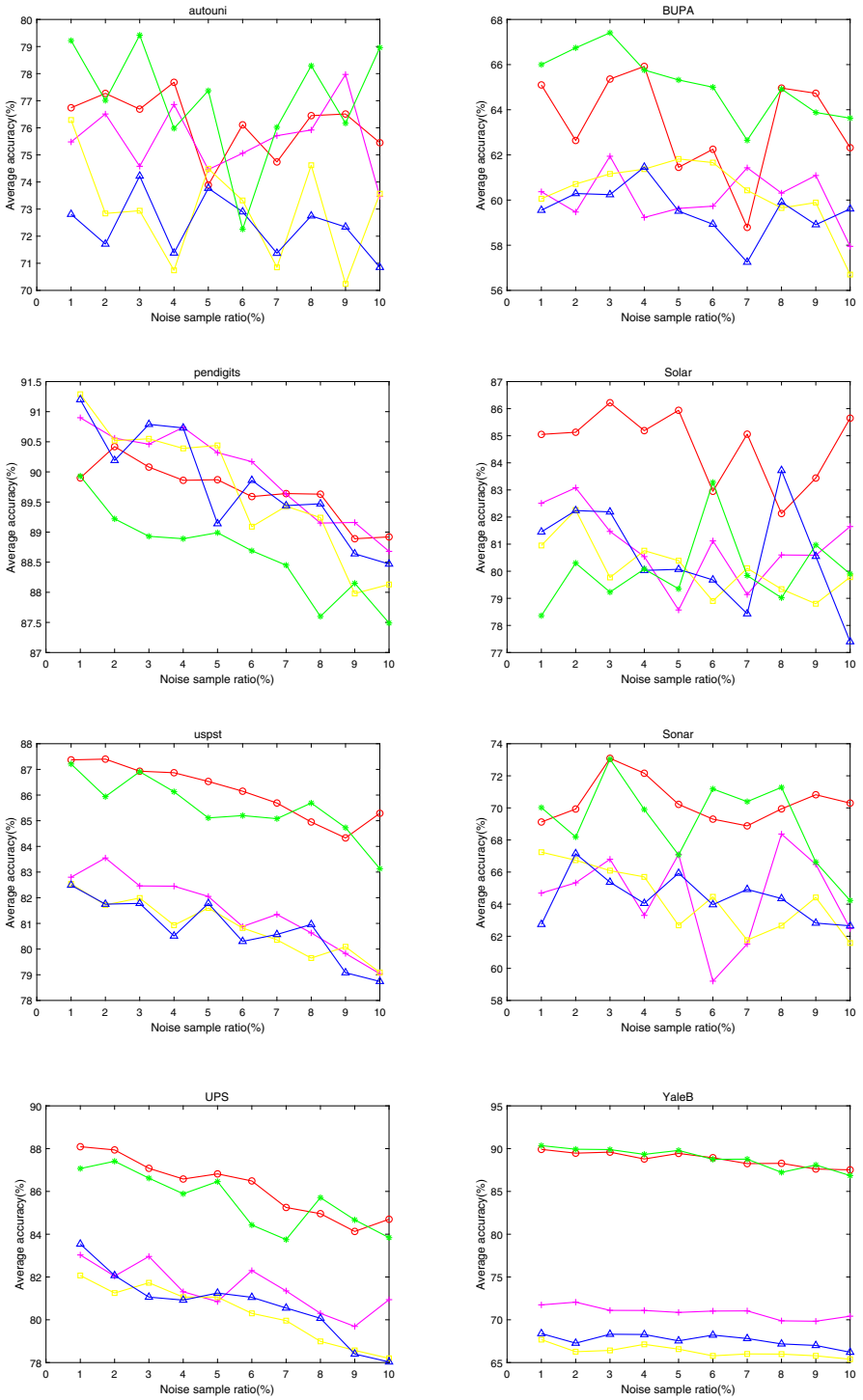


Fig. 5 (*continued*)

$O(tn^3)$, where n denotes the number of samples, t denotes the number of iterations. The MLSTE Algorithm uses the nearest neighbor editing technique(ENN) for data editing, the time complexity of computing the ENN is $O(n^3)$, and then the K-nearest neighbors(KNN) base classifier is used for self-training, and the overall complexity is $O(tn^3)$. The above analysis reveals that the current semi-supervised self-training algorithms need to calculate the distance between each pair-wise sample and have high time complexity, which limits the application of these algorithms in the area of big data.

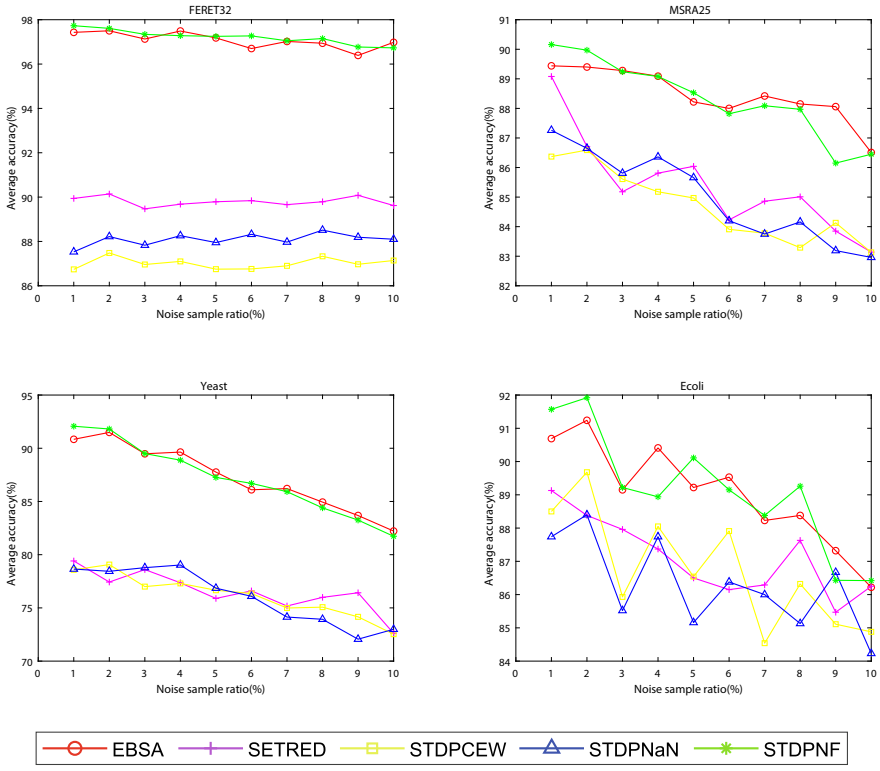


Fig. 5 (*continued*)

Table 6
Accuracy of each Algorithm on 20 data sets with noise (mean ± std).

Accuracy	EBSA	SETRED	STDPCEW	STDPNF	STDPNaN
AR	**95.19 ± 0.61**	81.46 ± 0.46	78.13 ± 0.31	95.04 ± 0.61	79.27 ± 0.37
COIL20	**90.26 ± 1.21**	81.92 ± 1.20	80.25 ± 1.30	90.17 ± 1.25	80.40 ± 1.40
Cleve	**84.42 ± 0.87**	78.89 ± 2.40	77.02 ± 2.41	83.5 ± 1.96	76.89 ± 1.88
Heart	**72.44 ± 2.16**	70.50 ± 1.27	70.17 ± 1.80	71.41 ± 2.69	70.61 ± 2.04
Isolet	**70.12 ± 1.36**	67.26 ± 2.15	66.97 ± 2.07	69.94 ± 1.72	67.05 ± 1.28
MINIST2k	84.75 ± 2.70	76.54 ± 2.22	75.85 ± 2.31	**84.78 ± 2.45**	76.36 ± 2.23
ORL	94.06 ± 0.33	83.18 ± 0.43	81.17 ± 0.50	**94.10 ± 0.42**	82.10 ± 0.60
Palm	94.30 ± 0.43	78.98 ± 1.25	75.26 ± 0.55	**94.47 ± 0.74**	77.14 ± 0.83
Solar	**84.68 ± 1.36**	80.93 ± 1.38	80.11 ± 1.04	80.03 ± 1.35	80.58 ± 1.89
Sonar	**70.38 ± 1.34**	64.53 ± 2.87	64.33 ± 2.08	69.19 ± 2.63	64.40 ± 1.48
UPS	**86.20 ± 1.37**	81.48 ± 1.10	80.32 ± 1.35	85.59 ± 1.34	80.69 ± 1.61
YaleB	88.78 ± 0.84	70.90 ± 0.71	66.30 ± 0.68	**88.90 ± 1.19**	67.62 ± 0.71
BUPA	63.35 ± 2.25	60.12 ± 1.17	60.35 ± 1.48	**65.13 ± 1.45**	59.57 ± 1.10
FERET32	97.08 ± 0.35	89.80 ± 0.21	87.01 ± 0.25	**97.22 ± 0.32**	88.09 ± 0.28
MSRA25	**88.46 ± 0.90**	85.39 ± 1.67	84.70 ± 1.24	88.35 ± 1.34	85.00 ± 1.53
Yeast	**87.24 ± 3.12**	76.55 ± 1.89	76.17 ± 2.01	87.15 ± 3.47	76.10 ± 2.64
autouni	76.15 ± 1.16	75.60 ± 1.31	72.99 ± 1.93	**77.07 ± 2.15**	72.41 ± 1.09
Ecoli	89.04 ± 1.55	87.11 ± 1.16	86.75 ± 1.72	**89.14 ± 1.82**	86.30 ± 1.35
pendigits	89.68 ± 0.48	**89.98 ± 0.77**	89.71 ± 1.11	88.63 ± 0.74	89.79 ± 0.93
uspst	**86.15 ± 1.06**	81.50 ± 1.42	80.88 ± 1.10	85.51 ± 1.16	80.80 ± 1.22
WSR-test	N/A	+	+	∼	+
Ave.ACC	84.64	78.13	76.72	84.27	77.06
Ave.std	1.27	1.35	1.36	1.54	1.32

The time complexity of the decision tree of the base classifier is $O(n \log n)$. EBSA needs $O(n)$ to calculate the centers of the ball-clusters and the time complexity of calculating the radii of the ball-clusters is $O(kn)$, where k is the number of clusters. Computing the distances between the samples and the centers of the clusters needs $O(kn)$ and calculate the pairwise distances of the centers needs $O\left(k^2\right)$. $k \ll n$, therefore, the total time complexity of the EBSA Algorithm is $O\left(t\left(2kn + n\log n + n + k^2\right)\right)$.

Table 7
Experimental accuracy of different base classifiers on 20 datasets.

Accuracy	SG + KNN(K = 1)	SG + RF	SG + SVM	SG + DT
AR	**99.12 ± 0.02**	99.1 ± 0.06	99.09 ± 0.09	96.23 ± 0.61
Cleve	75.71 ± 4.19	**87.29 ± 1.65**	73.21 ± 6.76	86.17 ± 2.13
COIL20	95.94 ± 0.05	95.95 ± 0.02	**95.97 ± 0.01**	92.61 ± 1.07
Heart	70.79 ± 7.57	**78.9 ± 1.05**	57.79 ± 6.70	74.55 ± 3.81
Isolet	76.31 ± 0.07	76.31 ± 0.11	**76.32 ± 0.14**	72.36 ± 1.81
MINIST2k	**91.37 ± 0.01**	**91.37 ± 0.01**	**91.37 ± 0.01**	85.94 ± 0.62
ORL	97.01 ± 0.06	97.25 ± 0.04	**97.31 ± 0.05**	95.58 ± 0.66
Palm	**99.06 ± 0.02**	99.05 ± 0.02	99.04 ± 0.14	95.46 ± 0.47
Solar	83.13 ± 3.13	**87.47 ± 2.11**	83.22 ± 0.45	86.25 ± 1.54
Sonar	77.94 ± 0.36	**78.03 ± 0.17**	53.37 ± 0.01	75.75 ± 2.81
UPS	92.04 ± 0.02	**92.64 ± 0.03**	92.30 ± 0.19	88.23 ± 1.39
YaleB	96.57 ± 0.01	**97.47 ± 0.21**	97.46 ± 0.17	91.93 ± 1.68
BUPA	65.94 ± 3.12	**71.73 ± 1.76**	42.58 ± 1.67	67.41 ± 4.72
FERET32	98.97 ± 0.16	**99.27 ± 0.12**	98.99 ± 0.13	97.75 ± 0.50
MSRA25	**92.87 ± 0.01**	92.86 ± 0.01	92.86 ± 0.01	90.63 ± 0.87
Yeast	92.51 ± 0.04	93.23 ± 0.08	93.48 ± 0.06	**93.51 ± 0.02**
autouni	70.12 ± 4.56	**83.00 ± 1.71**	58.92 ± 4.80	79.91 ± 1.36
Ecoli	91.15 ± 0.15	92.40 ± 0.64	92.31 ± 0.51	**93.03 ± 0.18**
pendigits	**92.15 ± 0.07**	92.13 ± 0.06	89.03 ± 0.46	90.37 ± 0.57
uspst	**92.65 ± 0.02**	92.64 ± 0.01	92.26 ± 0.19	88.34 ± 1.03
Ave.ACC	87.57	**89.90**	83.84	87.1
Ave.std	1.18	0.49	1.13	1.39

Table 8
Running time on different base classifiers.

runtime	SG + KNN(K = 1)	SG + RF	SG + SVM	SG + DT
AR	23.45	26.25	691.56	**22.53**
Cleve	**0.02**	0.49	2.05	**0.02**
COIL20	**2.58**	6.76	116.84	3.43
Heart	0.03	0.41	0.44	**0.01**
Isolet	**1.45**	4.16	0.96	1.83
MINIST2k	12.58	9.78	109.21	**2.87**
ORL	**0.20**	3.46	28.98	0.95
Palm	**1.09**	21.07	314.27	8.11
Solar	0.04	0.57	0.16	**0.03**
Sonar	**0.01**	0.27	0.03	0.02
UPS	1.07	4.61	3.72	**0.42**
YaleB	**6.11**	15.08	255.71	9.86
BUPA	**0.01**	0.41	0.14	0.03
FERET32	12.28	17.19	943.54	**11.14**
MSRA25	**0.76**	4.11	3.71	1.03
Yeast	**3.34**	6.27	19.13	4.20
autouni	**0.03**	0.40	5.45	**0.03**
Ecoli	**0.03**	0.72	0.33	0.08
pendigits	**0.15**	4.26	38.83	0.24
uspst	1.12	4.58	3.45	**0.44**
Ave.time	3.36	6.54	126.93	**3.32**

EBSA has linear time complexity in terms of the number of samples, which is substantially lower than the comparison algorithms. Given a data set with 10% labeled samples, each Algorithm was conducted the running time experiments 50 times and recorded the average running time. In Table 9, the numbers in parentheses indicate the ordering of the running time of the Algorithm under the current data set among the five algorithms (ordered from the shortest time to the longest). Runtime experiments were conducted on 20 data sets and the results are shown in Table 9.

From Table 9, we can draw the following conclusions:

(1)The running times of the proposed Algorithm EBSA are substantially lower than the comparison algorithms SETRED, STDPCEW, STDPNaN and STDPNF on all 20 data sets. Table 9 also shows that EBSA has the shortest running time and STDPCEW Algorithm has the longest running time. SETRED needs to construct related adjacency graphs and the overall time complexity is $O(tn^3)$. The time complexity of the STDPCEW Algorithm is $O(tn^3)$. The STDPNaN Algorithm and STDPNF find the spatial structure and the natural nearest neighbor using DPC with $O(n^2)$ and $O(n\log n)$, respectively. Therefore, the overall time complexity of STDPNaN is $O(tn^2)$. The overall time complexity of STDPNF is $O(tn^2)$. Therefore, the time complexity of EBSA is the lowest.

Table 9
The running time of each Algorithm on 20 data sets.

time	EBSA	SETRED	STDPCEW	STDPNF	STDPNaN
AR	**22.53(1)**	705.05(5)	320.15(4)	130.03(3)	52.29(2)
Cleve	**0.02(1)**	1.27(4)	0.55(3)	0.12(2)	0.12(2)
COIL20	**3.43(1)**	74.61(4)	145.03(5)	23.95(3)	17.99(2)
Heart	**0.01(1)**	1.24(4)	0.32(5)	0.09(2)	0.16(3)
Isolet	**1.83(1)**	26.86(4)	77.74(5)	53.75(2)	11.21(2)
MINIST2k	**2.87(1)**	380.67(4)	1965.18(5)	332.93(3)	161.29(2)
ORL	**0.95(1)**	28.24(5)	6.07(4)	6.60(3)	1.91(2)
Palm	**8.11(1)**	253.86(4)	281.93(5)	32.40(2)	33.15(3)
Solar	**0.03(1)**	2.14(4)	0.43(3)	0.20(2)	0.20(2)
Sonar	**0.02(1)**	1.29(5)	0.27(4)	0.23(3)	0.14(2)
UPS	**0.42(1)**	45.67(4)	208.04(5)	39.73(3)	18.18(2)
YaleB	**9.86(1)**	312.61(4)	446.26(5)	164.95(3)	57.53(2)
BUPA	**0.03(1)**	1.65(5)	0.62(4)	0.11(2)	0.26(3)
FERET32	**11.14(1)**	472.56(5)	111.73(4)	76.92(3)	20.88(2)
MSRA25	**1.03(1)**	35.15(4)	91.09(5)	9.37(2)	15.42(3)
Yeast	**4.20(1)**	55.19(5)	25.09(4)	11.44(3)	8.00(2)
uspst	**0.44(1)**	37.91(3)	201.14(5)	40.66(4)	18.55(2)
autouni	**0.03(1)**	1.67(5)	0.23(4)	0.13(3)	0.10(2)
Ecoli	**0.08(1)**	2.96(5)	0.98(4)	0.90(3)	0.25(2)
pendigits	**0.24(1)**	143.37(4)	1514.17(5)	9.45(2)	91.85(3)
Ave.time	3.36	129.20	269.85	52.63	26.8
Ave.rank	1	4.35	4.4	2.65	2.25

(2) On the 20 data sets, the average running time of the EBSA Algorithm is significantly smaller than the other four comparison algorithms. The running time of the EBSA Algorithm is 2.6%, 1.2%, 12.5% and 6.3% of that of the SETRED, STDPCEW, STDPNaN and STDPNF algorithms, respectively. The running time order of the EBSA Algorithm is all 1, which proves that the EBSA Algorithm is ahead of other comparison algorithms with respect to running time, and verifies the correctness of the theoretical analysis.
(3) Since the Cleve, Heart, Solar, Sonar, BUPA, autouni and Ecoli data sets are relatively small compared to the other 13 data sets, the running time of the proposed Algorithm EBSA on these seven data sets don't exceed 0.05s, which is considerably faster than the other four compared algorithms.
(4) The running time of the EBSA Algorithm is only 16.67%, 15%, 14.29%, 30%, 32%, 11.11% and 27% on Cleve, Solar, Sonar, autouni, Ecoli, Heart and BUPA of that of the second ranking algorithm, respectively. On the image data sets MINIST2k and UPS, the running time of EBSA is 1.78% and 2.31% of that of the STDPNaN algorithm, respectively. Obviously, the experimental results are consistent with the theoretical analysis.

6. Conclusion

The current semi-supervised machine learning algorithms with self-training Algorithm framework have high computational complexity and are not suitable for big data scenarios. To address this problem, we propose a novel data editing method, which can not only handle noisy data conveniently, but also select high-confidence samples for iterative training quickly. The proposed Algorithm divides the data set into stable and disputed areas by partitioning the ball-clusters. The method requires less computational cost because only the distances between the sample points in the disputed area and the centers need to be calculated in each iteration of self-training. Moreover, data editing is used to identify and delete the mislabeled samples, which improves the quality of the selected high-confidence samples. The experimental results of the Wilcoxon signed ranks test at the level of confidence of 95% indicate that the classification performance of EBSA is significantly better than that of SETRED, STDPCEW and STDPNaN, and slightly higher than that of STDPNF.

In general, the proposed EBSA has the following advantages: (1) Compared with other related algorithms based on the self-training framework, the training speed of the EBSA is significantly improved. Therefore, EBSA can handle largescale data. (2) EBSA can edit noise data and select high-confidence samples with high quality. As a result, the performance of the learned classifier is improved. (3) EBSA does not require hyperparameters, which makes it more convenient for practical applications.

Our Algorithm has two main disadvantages: (1) The initial labeled data distribution significantly impacts the proposed algorithm's performance. (2) EBSA uses ball-cluster partition and editing Algorithm to edit noise data, making it unable to deal with the non-spherical distribution data sets.

Because EBSA uses the euclidean distance to measure the distance between the samples, it does not apply to categorical data and mixed numeric data. On the other hand, the EBSA Algorithm is not suitable for non-spherical distribution data sets.

In the future, we will expand our Algorithm from the following aspects: (1) By increasing the number of ball-clusters, we will extend the EBSA Algorithm to the non-spherical distribution data sets. (2) For categorical and mixed-type data sets, we

will employ appropriate distance calculation methods to measure the distance of the samples so that our Algorithm can be applied to these scenarios.

CRediT authorship contribution statement

Bing Li: Software, Writing - original draft. **Jikui Wang:** Conceptualization, Methodology, Writing - review & editing. **Zhengguo Yang:** Data curation. **Jihai Yi:** Visualization. **Feiping Nie:** Supervision.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work was partially supported by NSFC (61772427), Foundation of State Key Laboratory of Public Big Data (No.GZU-PBD2021-101), Natural Science Foundation of Gansu Province (21JR11RA132,22JR5RA554), Gansu University Innovation Fund Project (2022A-092) and Key R & D projects in Gansu Province (21YF5FA087).

References

[1] X. Gu, A self-training hierarchical prototype-based approach for semi-supervised classification, Inf. Sci. 535 (2020) 204–224.
[2] J. Tanha, M. van Someren, H. Afsarmanesh, Semi-supervised self-training for decision tree classifiers, Int. J. Mach. Learn. Cybern. 8 (1) (2017) 355–370.
[3] M. Zhou, Y. Li, H. Lu, C. Nengbin, Z. Xuejun, Semi-supervised meta-learning via self-training, in: 2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS), IEEE, 2020, pp. 1–7.
[4] Y. Zou, Z. Yu, X. Liu, B. Kumar, J. Wang, Confidence regularized self-training, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 5982–5991.
[5] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proceedings of the ninth international conference on Information and knowledge management, 2000, pp. 86–93.
[6] F. Ma, D. Meng, Q. Xie, Z. Li, X. Dong, Self-paced co-training, in: International Conference on Machine Learning, PMLR, 2017, pp. 2275–2284.
[7] X. Ning, X. Wang, S. Xu, W. Cai, L. Zhang, L. Yu, W. Li, A review of research on co-training, Concurrency and computation: practice and experience (2021) e6276.
[8] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, A. Torralba, Rewriting a deep generative model, in: European Conference on Computer Vision, Springer, 2020, pp. 351–369.
[9] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499.
[10] Y. Tang, G. Yang, D. Ding, G. Cheng, Multi-level amplified iterative training of semi-supervision deep learning for glaucoma diagnosis, in: 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2021, pp. 747–750.
[11] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, Y. Song, Metagan: An adversarial approach to few-shot learning, Advances in neural information processing systems 31.
[12] Z.-Q. Wei, H.-X. Bi, X. LIU, A graph-based semi-supervised polsar image classification method using deep convolutional neural networks, Acta Electonica Sinica 48 (1) (2020) 66.
[13] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C. Raffel, Mixmatch: A holistic approach to semi-supervised learning, arXiv preprint arXiv:1905.02249 (2019) 5050–5060.
[14] X. Zhou, M. Belkin, Semi-supervised learning, Vol. 1, Elsevier, 2014, pp. 1239–1269.
[15] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, Synthesis lectures on artificial intelligence and machine learning 3 (1) (2009) 1–130.
[16] J. Liu, Y. Liu, X. Luo, Semi-supervised learning method, Chin. J. Comput. 38 (8) (2015) 1592–1617.
[17] Y. Wang, X. Xu, H. Zhao, Z. Hua, Semi-supervised learning based on nearest neighbor rule and cut edges, Knowl.-Based Syst. 23 (6) (2010) 547–554.
[18] Z. Wei, H. Wang, R. Zhao, Semi-supervised multi-label image classification based on nearest neighbor editing, Neurocomputing 119 (2013) 462–468.
[19] D. Wu, M. Shang, G. Wang, L. Li, A self-training semi-supervised classification Algorithm based on density peaks of data and differential evolution, in: 2018 IEEE 15th international conference on networking, Sensing and Control (ICNSC), IEEE, 2018, pp. 1–6.
[20] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, H. Yan, A highly accurate framework for self-labeled semisupervised classification in industrial applications, IEEE Trans. Industr. Inf. 14 (3) (2017) 909–920.
[21] H. Gan, N. Sang, R. Huang, X. Tong, Z. Dan, Using clustering analysis to improve semi-supervised classification, Neurocomputing 101 (2013) 290–298.
[22] G.J. McLachlan, Mahalanobis distance, Resonance 4 (6) (1999) 20–26.
[23] S. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, W. Wei, Z. Chen, A fast adaptive k-means with no bounds, IEEE Trans. Pattern Anal. Mach. Intell.
[24] S. Zhang, X. Li, M. Zong, X. Zhu, D. Cheng, Learning k for knn classification, ACM Trans. Intell. Syst. Technol. (TIST) 8 (3) (2017) 1–19.
[25] M. Li, Z.-H. Zhou, Setred: Self-training with editing, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2005, pp. 611–621.
[26] J. Li, Q. Zhu, 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. 184 (2019) 104895.
[27] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[28] Y. Liu, Self-training Algorithm combining density peak and cut edge weight, J. Vis. Lang. Comput. 2020 (1) (2020) 11–16.
[29] S. Zhao, J. Li, A semi-supervised self-training method based on density peaks and natural neighbors, J. Ambient Intell. Humaniz. Comput. 12 (2) (2021) 2939–2953.
[30] B. Charbuty, A. Abdulazeez, Classification based on decision tree Algorithm for machine learning, J. Appl. Sci. Technol. Trends 2 (01) (2021) 20–28.
[31] C. Manapragada, G.I. Webb, M. Salehi, Extremely fast decision tree, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1953–1962.

[32] A.J. Myles, R.N. Feudale, Y. Liu, N.A. Woody, S.D. Brown, An introduction to decision tree modeling, J. Chemometr.: J. Chemometr. Soc. 18 (6) (2004) 275–285.
[33] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Syst., Man, Cybern. 21 (3) (1991) 660–674.
[34] Z. He, X. Xu, S. Deng, Clustering mixed numeric and categorical data: A cluster ensemble approach, arXiv preprint cs/0509011.
[35] A. Martínez, R. Benavente, The ar face database, 1998, Computer vision center, technical report 3 (5).
[36] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, D. Cremers, Clustering with deep learning: Taxonomy and new methods, arXiv preprint arXiv:1801.07648.
[37] J. Xu, J. Han, K. Xiong, F. Nie, Robust and sparse fuzzy k-means clustering, IJCAI, in, 2016, pp. 2224–2230.
[38] M.S. Hasan, X. Wu, L.T. Watson, L. Zhang, Ups-indel: a universal positioning system for indels, Sci. Rep. 7 (1) (2017) 1–13.
[39] A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman, From few to many: Illumination cone models for face recognition under variable lighting and pose, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 643–660.
[40] R. Cole, M. Fanty, Spoken letter recognition, in: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990, 1990.
[41] H. Li, M. Wang, X.-S. Hua, Msra-mm 2.0: A large-scale web multimedia dataset, in: 2009 IEEE International Conference on Data Mining Workshops, IEEE, 2009, pp. 164–169.
[42] W. Zhao, A. Krishnaswamy, R. Chellappa, D.L. Swets, J. Weng, Discriminant analysis of principal components for face recognition, Face Recognition, Springer, in, 1998, pp. 73–85.
[43] L. Guo, H. Gan, S. Xia, X. Xu, T. Zhou, Joint exploring of risky labeled and unlabeled samples for safe semi-supervised clustering, Expert Syst. Appl. 176 (2021), https://doi.org/10.1016/j.eswa.2021.114796, https://www.sciencedirect.com/science/article/pii/S0957417421002372 114796.
[44] I.E. Livieris, A. Kanavos, V. Tampakas, P. Pintelas, An auto-adjustable semi-supervised self-training algorithm, Algorithms 11 (9) (2018) 139.