Applied Intelligence (2023) 53:14892–14902
https://doi.org/10.1007/s10489-022-04169-w

Adaptive active learning through k-nearest neighbor optimized local density clustering

Xia Ji[1] · WanLi Ye[1] · XueJun Li[1] · Peng Zhao[1] · Sheng Yao[1]

Accepted: 11 September 2022 / Published online: 4 November 2022

Abstract
Active learning iteratively constructs a refined training set to train an effective classifier with as few labeled instances as possible. In areas where labeling is expensive, active learning plays an important and irreplaceable role. The main challenge of active learning is to correctly identify critical samples. One of the current mainstream methods is to mine the potential data structure based on clustering and then identify key instances. However, the existing methods all adopt deterministic strategies, and the number of key samples is only related to the number of samples to be classified. The internal structure information of the sample clusters to be classified is not used. After analysis and verification, this deterministic key sample selection strategy has serious label waste. This is a serious problem that urgently needs to be solved in active learning. To this end, we propose an adaptive active learning algorithm based on density clustering (AAKC). Firstly, we introduce k-nearest neighbor information to redefine the local density of the instance. The new sample density can clearly express the local structural information of the sample. Secondly, we developed an adaptive key instance selection strategy based on the k-nearest neighbor sample density, which can adaptively select the necessary number of instance queries according to the structural information of the instance clusters to be classified, avoiding label waste. The experimental results of comparison with other algorithms show that our algorithm uses fewer labels to achieve better classification accuracy and has excellent stability.

Keywords Active learning · K-nearest neighbor · Density peak clustering · Adaptive instance selection

1 Introduction

Machine learning studies the generation of "models" from data, but the necessary prerequisite for generating effective models is sufficient high-quality data. However, in many practical applications, obtaining enough labeled instances is very time-consuming and expensive. Therefore, learning an effective model from a large quantity of data with few labels is significant. Active learning is the most popular method for solving this problem, which actively selects some samples that are "valued" for the model to be added to the training set, aiming to train the expected model with as little data labeling cost as possible. In recent years, active learning has been widely applied in object categorization [1], image retrieval and classification [2–4], speech recognition [5], multilabel annotation [6], and feature representation [7] fields.

The core of active learning is the sample selection strategy, and reasonable sample selection can effectively reduce the data labeling cost. Cluster-based sample selection is one of the most popular methods. McCallum and Nigam [8] proposed a general schema of cluster-based active learning by constructing hierarchical cluster trees where each leaf corresponds to a sample and each inner node corresponds to a cluster. The process of active learning involves pruning the main tree and randomly selecting samples from each

✉ Xia Ji
jixia1983@163.com

WanLi Ye
yewl_ahu@163.com

XueJun Li
xjli@ahu.edu.cn

Peng Zhao
zhaopeng_ad@ahu.edu.cn

Sheng Yao
fisheryao@126.com

1 The School of Computer Science and Technology, Anhui University, Hefei, China

cluster. Dasgupta and Hsu [9] proposed a method based on hierarchical clustering. Wang [10] proposed a method based on density peak clustering. Cluster-based approaches do not require additional classifiers and make full use of unlabeled data, avoiding the problem of sampling bias.

However, the performance of cluster-based active learning algorithms is very dependent on the quality of the clustering results. The density peak clustering algorithm calculates the local density by the number of samples within the cutoff distance, and the appropriate cutoff distance can greatly improve the clustering results quality. However, it is difficult to set a cutoff distance that is appropriate for all samples, especially if the dataset is unbalanced [11]. In addition, most of the existing cluster-based methods use a determined sample selection strategy to select the same number of key samples in different sample clusters to be classified. This strategy ignores the distribution information of the sample clusters to be classified, and inevitably selects some redundant samples that have little improvement in model performance.

To solve the above problems, an adaptive active learning algorithm based on optimized density clustering is proposed in this paper. Firstly, we introduce the k-neighbor information of the instance to redefine the local density of the instance. This method improves the stability of the clustering results. More importantly, it better describes the local data structure, which is convenient for finding the most representative samples. Secondly, we propose an adaptive instance selection strategy. This strategy adaptively determines the number of samples that need to be selected in each cluster, avoiding the choice of redundant data. Finally, we compare the algorithms presented in this paper with ten commonly used supervised learning algorithms and eight of the most popular active learning algorithms. The results of the comparative experiments show that AAKC achieves higher classification accuracy and good stability by using fewer labels on the datasets with different instances, dimensions and clusters.

2 Related work

2.1 The density peak clustering algorithm

Rodriguez and Liao [12] proposed a density peak clustering algorithm (DPC), which can automatically find cluster centers to achieve efficient clustering of arbitrary shape data. It is based on the straightforward idea about clustering centers: 1) its local density is greater than the local density of its neighbors; 2) the distance between the centers of different clusters is relatively large. The DPC algorithm uses a decision graph to select cluster centers. The decision graph is generated based on two fundamental attributes of each instance: ρ_i and δ_i. For an instance x_i, local density ρ_i is defined as follows:

$$\rho_i = \sum_{j=1}^{|U|} \chi(d_{ij} - d_c). \tag{1}$$

where d_{ij} is the distance between instances x_i and x_j, and d_c is the cutoff distance. d_c is usually empirically selected so that the average number of neighbors of the sample is approximately $1 \sim 2\%$ of the total number of instances. When $(\cdot) < 0$, $\chi(\cdot) = 1$, otherwise $\chi(\cdot) = 0$. δ_i is calculated as follows:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}). \tag{2}$$

δ_i denotes the distance between x_i and its nearest neighbor with a higher ρ_i. In addition, for x_i with the most significant local density ρ_i, we conventionally take

$$\delta_i = \max_j (d_{ij}). \tag{3}$$

The decision graph uses ρ_i as the x-axis and δ_i as the y-axis. According to the decision graph, instances with larger ρ_i and δ_i are selected as cluster centers, and the remaining instances are clustered.

The clustering performance of the DPC algorithm is excellent, but it still has its limitations. Firstly, it is difficult for the DPC algorithm to find a cutoff distance d_c that fits all samples, especially for unbalanced datasets. Secondly, the clustering result of the DPC algorithm is susceptible to the value of d_c. If d_c changes slightly, the clustering results may be completely different [11]. These shortcomings of the DPC algorithm will naturally be passed on to the active learning algorithm based on it.

3 The proposed method

Currently, most active learning algorithms based on clustering adopt a deterministic sample selection strategy, which inevitably leads to waste of labels. In this paper, we propose a novel adaptive active learning algorithm based on optimized density clustering. Figure 1 illustrates the algorithm overview, and the pseudocode is summarized in Algorithm 1.

3.1 Initialization

Given a dataset $U = \{x_i\}_{i=1}^{n}$, n and m are the numbers of instances and features, respectively. In the initialization, we search k nearest neighbors for each sample, introduce the k nearest neighbor information, and use an exponential kernel function with a width of $\theta = 1$ to redefine the local density. This method can better represent the local structure of the data and avoid the detrimental effect of cutoff distance on

```
Require: $U$, $k$, $w$, $\lambda$, and $N$.
Ensure: $l(x)$ for all $x \in U$.
1:  $U_I = U_{II} = \emptyset$; $U_{III} = U$;
2:  search $knn(i)$ for each instance;
3:  calculate $\rho_{knn(i)}$, $\delta_i$, and $\gamma_i$;
4:  for($b = 2$ ; $(|U_I| < N) \wedge (U_{III} \neq \emptyset)$; $b$++ )
5:      $bl$ = optimizedLocalDensityClustering($U$, $b$);
6:      for($i = 1$; $i \leq b$; $i$ ++)
7:          $U_{tp}$ = SlidingWindowAlgorithm($bl_i$);
8:          if( $|U_I \cup U_{tp}| \leq N \wedge U_{III} - U_{tp} \neq \emptyset$) then
9:              $U_I = U_I \cup U_{tp}$; $U_{III} = U_{III} - U_{tp}$;
10:             if($bl_i$ is not fully labeled $\wedge$ $p_i$ is pure) then
11:                 $U'_{II}$ = classify($bl_i$ - $p_i$);
12:                 $U_{II} = U_{II} \cup U'_{II}$; $U_{III} = U_{III} - U'_{II}$;
13:             end if
14:         else
15:             break;
16:     end for
17: end for
18: if($U_{III} \neq \emptyset$) then
19:     use standard voting to classify samples in $U_{III}$;
20: end if
21: return $l(x) = \{l(x_1), ..., l(x_n)\}$
```

Algorithm 1 Adaptive active learning through k-nearest neighbor optimized local density clustering.

clustering and active learning. The k-nearest neighbor based local density $\rho_{knn(i)}$ of x_i is defined as follows:

$$\rho_{knn(i)} = \sum_{j \in knn(i)} exp(-d_{ij}). \tag{4}$$

where d_{ij} is the Euclidean distance between x_i and x_j, and $knn(i)$ is the set of k-nearest neighbor instances of x_i. The smaller the sum of the distance between sample x_i and the sample in set $knn(i)$, the greater the local density of the sample x_i, and vice versa. We also compute the minimum distance between x_i and the nearest instance with a higher local density δ_i as (2) and (3).

In cluster-based active learning, if only the local density of the sample is considered, the algorithm usually selects only the samples that are close to the center of the cluster. Although these samples are generally highly representative, they also have greater similarities between them. Selecting too many such samples does not greatly improve the performance of active learning and wastes limited labeled data, which is unacceptable for active learning tasks with limited budgets for data labeling. To solve this problem, we define an index to measure the influence of a sample within a sample cluster, which is calculated as follows:

$$\gamma_i = \rho_{knn(i)} \times \delta_i \tag{5}$$

This makes each cluster structure center area have only one sample with large intracluster influence, which ensures the representativeness and diversity of the training set and avoids the waste of labels.

3.2 Adaptive instance selection

The core of active learning is the instance selection strategy. The instance selection strategy contains two basic questions, what kind of samples to choose? And how many samples to choose? The determined instance selection strategy usually selects the same number of samples in different clusters. This strategy ignores the different requirements between clusters. As a result, too many instances are selected in some clusters, which wastes the limited label budget and reduces the number of queryable clusters, but cannot greatly improve the algorithm performance.

To solve this problem, we propose an adaptive instance selection strategy. Firstly, we perform an optimized local density clustering algorithm to divide the entire dataset into

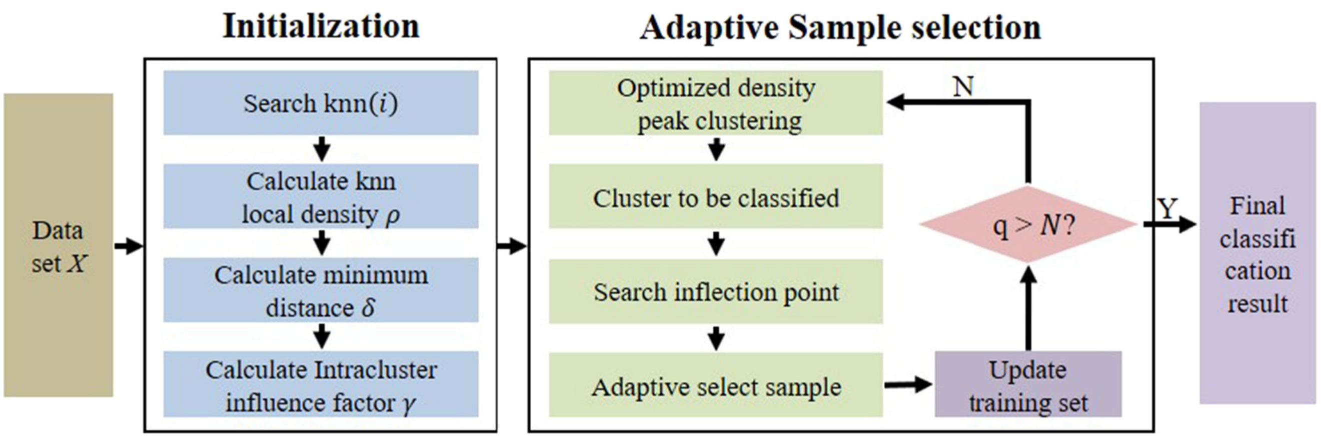
Initialization
Adaptive Sample selection
Data set X
Search knn(i)
Calculate knn local density ρ
Calculate minimum distance δ
Calculate Intracluster influence factor γ
Optimized density peak clustering
Cluster to be classified
Search inflection point
Adaptive select sample
Update training set
q > N?
N
Y
Final classifi cation result

Fig. 1 The Overview of the AAKC algorithm

b clusters. The initial value of b is two, and its value is increased by one for each iteration.

Secondly, we build a decision graph for each cluster and use the sliding window algorithm to adaptively find turning points, and the detailed process is shown in Algorithm 2. The y-axis of the decision graph represents the γ_i value of the instance and the x-axis represents the instances ordered descending by the γ value. From the decision graph, we find that the value of γ decreases significantly before the turning point, and decreases flat after the turning point. This indicates that the sample's representativeness after the turning point is relatively weak, we don't have to spend expensive time and cost to query these instance labels. To find the turning point in the dosicion graph, we design a sliding window algorithm. In the silding window algorithm, we use the variance of γ within a sliding window of to measure the flatness of the decline in sample representativeness. The larger the variance in the sliding window, the more rapidly the sample representativeness declines. We sort the samples in bl_i by γ descending order to get the subscript array s, and let p start from 0 to $s.length - w$, where w is the width of the sliding window. The turning point is defined as the minimum point where the variance within the sliding window is less than λ, as follows:

$$tp = \min_p [var(\gamma_{s_{[p]}}, ..., \gamma_{s_{[p+w-1]}}) < \lambda],\ p \in [0, s.length - w]. \tag{6}$$

Finally, we selected tp samples with largest γ to be labeled.

```
Require: $w$, $\lambda$, and $bl_i$.
Ensure: The number of critical samples.
1: $s$ = sort the samples in $bl_i$ by $\gamma$ descending order;
2: for($p = 0$; $p \le s.length - w$; $p$++)
3:     if ($var(\gamma_{s_{[p]}}, ..., \gamma_{s_{[p+w-1]}}) < \lambda$) then
4:         $tp = p$;
5:         break;
6:     end if
7: end for
8: return $U_{tp} = \{s_0, ..., s_{tp}\}$;
```

Algorithm 2 Sliding window algorithm.

3.3 Clasify

After selecting and querying the labels of these critical samples, we classify other instances in the cluster based on the existing labels. If the labeled sample set is pure (i.e., the labels of the samples in p_i are all the same, where p_i is the label set in the sample cluster bl_i), this cluster is considered pure. The remaining unlabeled instances are classified with the same label. Otherwise, we choose critical instances based on the current information in the next iteration until the cluster is divided into pure or the label budget N is exhausted. Assume there are still impure and unlabeled data after N labels are used up. We use the standard voting methods to assign the label with the most instances in the cluster to the remaining samples. Assume that a cluster to be classified contains 20 samples, including 5 samples to be classified, 10 samples of class A, 2 samples of class B, and 3 samples of class C. The class A samples in the cluster are the most abundant, so the standard voting method classifies the remaining 5 samples as class A.

3.4 Illustrative example

To illustrate our algorithm, we analyze a running example on the Aggregation dataset. The Aggregation dataset is a two-dimensional dataset from the UCI machine learning repository [18], with 7 clusters of 788 samples. The experimental parameters were set to $k = 5$, $w = 5$, and $\lambda = 0.001$, respectively. The label budget for AAKC is $N = 78$, accounting for10% of the total number of samples. Table 1 shows the size information of each cluster, the current label information, and the number of newly selected samples. Figures 2 and 3 show the decision graph of bl_1 and bl_2, respectively. To facilitate showing the turning point location, we marked its location with a red dotted line.

We can see that the AAKC algorithm clusters the dataset into two instance clusters after initialization, and generates decision graphs, respectively. The turning point in the two decision graphs are the eleventh point and the seventh point. At this time, there are no labeled instances in bl_1 and bl_2, so we need to select 11 and 7 instances in bl_1 and bl_2 respectively. After labeling, the label set in bl_1 and bl_2 are $p_1 = \{4, 1, 3, 6, 3, 1, 1, 3, 1, 3, 1\}$ and $p_2 = \{5, 0, 2, 2, 5, 2, 5\}$, respectively. Since none of them are pure clusters, we also make them wait for the next iteration. The number of labels used is 11+7 = 18, which is less than 78, so AAKC performs the next iteration. In the second iteration, the AAKC algorithm first divides the entire dataset into bl_1, bl_2, and bl_3 and builds a decision graph for each cluster. The turning points in the three decision graphs are the 7th, 7th, and 6th points. There are 6 labeled instances in bl_1, 7 labeled instances in bl_2, and 5 labeled instances in bl_3, so we select 1 instance in bl_1 and bl_3. After labeling, the label set in bl_1 and bl_2 are $p_1 = \{4, 3, 6, 3, 3, 3, 3\}$ and $p_2 = \{5, 0, 2, 2, 5, 2, 5\}$, respectively. Since none of them are pure clusters, we also made them wait for the next iteration. The label set in bl_3 is $p_3 = \{1, 1, 1, 1, 1, 1\}$, and all the labels in the cluster are the same, so it is a pure cluster. We directly put the remaining instances in bl_3 into one category. The number of labels used after the second iteration is 18+1+1 = 20, less than 78, so the iteration

Table 1 Active learning process information of AAKC where "AS" represents the number of samples that have been selected, "NS" represents the number of newly selected samples, and the total number of selected instances is 32

Iteration	ID	Size	TP	AS	NS
One	bl_1	510	11	0	11
	bl_2	277	7	0	7
Two	bl_1	341	7	6	1
	bl_2	277	7	7	0
	bl_3	169	6	5	1
Three	bl_1	34	3	1	2
	bl_2	277	7	7	0
	bl_3	169	6	6	0
	bl_4	307	6	6	0
Four	bl_1	34	3	3	0
	bl_2	232	6	6	0
	bl_3	169	6	6	0
	bl_4	307	6	6	0
	bl_5	45	2	1	1
Five	bl_1	34	3	3	0
	bl_2	127	7	3	4
	bl_3	169	6	6	0
	bl_4	307	6	6	0
	bl_5	45	2	2	0
	bl_6	105	7	3	4
Six	bl_1	34	3	3	0
	bl_2	127	7	7	0
	bl_3	169	6	6	0
	bl_4	273	6	5	1
	bl_5	45	2	2	0
	bl_6	105	7	7	0
	bl_7	34	2	1	1

continues. The remaining iterations are performed in the same way, and finally, the AAKC algorithm uses 32 labels, iterates six times dividing the dataset into seven clusters, and completes the classification of all instances.

For comparison, we also run the ALEC algorithm on the Aggregation data. Table 2 shows the size information of each cluster, the current label information, and the number of newly selected samples. In the first iteration, ALEC divides the master tree into two clusters bl_1 and bl_2 through clustering. Since $N = 78$, ALEC selects $\sqrt{N} = 8$ instances in bl_1 and bl_2. At this time, the number of labels used is 8+8 = 16, which is less than 78. Therefore ALEC performs the next iteration. In the second iteration, ALEC first divides the entire dataset into bl_1, bl_2, and bl_3. Then, ALEC selects 8 instances for all the clusters. After the second iteration, the number of labels used is 16+8+8+8 = 40 less than 78, so ALEC performs the next iteration. The remaining

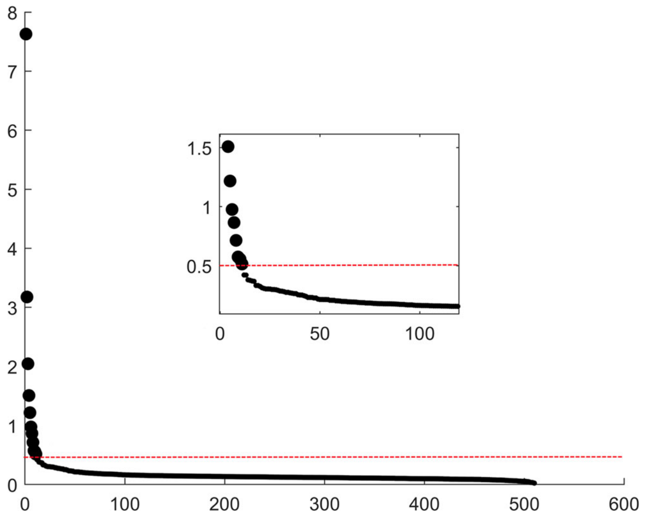

Fig. 2 The decision graph of bl_1

iteration runs in the same manner. Finally, the ALEC algorithm runs out of budget in the fourth iteration and then uses the standard voting method to classify the remaining unclassified instances. The ALEC algorithm tends to select more instances in the earlier generated clusters, which causes ALEC to run out of labels and exit the loop quickly. For example, ALEC chooses a total of 24 instances in the bl_2 cluster. The remaining instances can only use the standard voting method classification. AAKC selects only truly representative samples in each cluster, making AAKC iterate more times and finally complete the category of all instances. The result of this example is that ALEC uses all 78 labels with a classification accuracy of 0.9506, and AAKC uses only 32 labels with a classification accuracy of 0.9972.

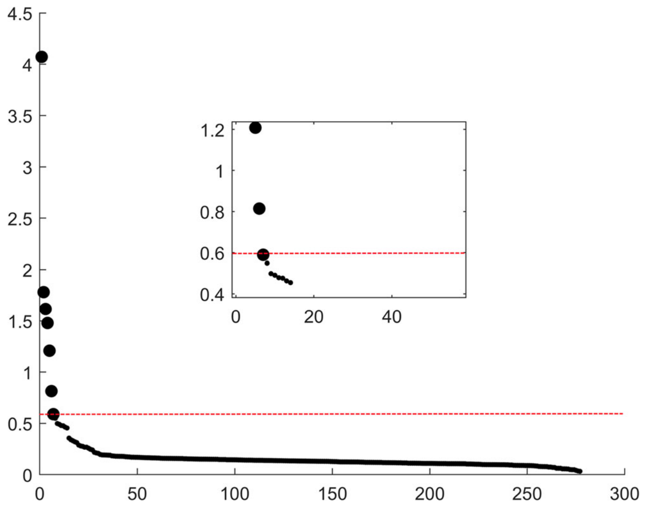

Fig. 3 The decision graph of bl_2

Table 2 Active learning process information of ALEC where “AS” represents the number of samples that have been selected, “NS” represents the number of newly selected samples, and the total number of selected instances is 78

Iteration	ID	Size	AS	NS
One	bl_1	510	0	8
	bl_2	277	0	8
Two	bl_1	510	8	8
	bl_2	177	6	8
	bl_3	100	2	8
Three	bl_1	510	16	8
	bl_2	45	9	8
	bl_3	100	10	8
	bl_4	132	5	8
Four	bl_1	169	7	6
	bl_2	45	17	0
	bl_3	100	18	0
	bl_4	132	13	0
	bl_5	341	17	0

3.5 Complexity analysis

The time complexity of the AAKC algorithm consists of two parts. Initialization. Firstly, we need to calculate γ, $\rho_{knn(i)}$, and δ for each sample, and the time complexity is $O(mn^2)$. Adaptive instance selection. This stage usually involves several iterations. In each iteration, we take $O(nlog_n)$ time to sort γ_i and select centers and take $O(n)$ time to assign the cluster index to non-center instances. We obtain several sample clusters to be classified. Asumme that the cluster size is n', we take $O(nlog_n)$ to sort and take $O(n')$ to calculate the variance. Since the size of a cluster is less than n, this part takes $O(nlog_n)$. Now we classify pure clusters, the number of instances in the pure cluster is $n' - |p_i|$, and classification takes $O(n' - |p_i|)$ time. Since $n' - |p_i|$ is less than n, this part takes $O(n)$. The algorithm usually iterates k' times, so the total time complexity of AAKC is $O(mn^2) + O(k'(nlog_n + nlog_n + n))$. Since k' is much smaller than n, the total time complexity of the AAKC algorithm is $O(mn^2)$.

4 Experiment

To verify the AAKC algorithm’s effectiveness, we compare the AAKC algorithm with the mainstream supervised classification algorithm and the latest active learning algorithm. The parameter settings of the AAKC algorithm are as follows, $w = 5$ and $\lambda = 0.001$. We use four synthetic datasets [12] and five UCI benchmark datasets [18], and Table 3 shows information about these datasets.

Table 3 Dataset information where “n” represents the number of samples, “c” represents the number of categories, and “m” represents the number of features

Dataset	n	c	m
Aggregation	788	7	2
Spiral	312	2	2
Flame	240	3	2
R15	600	15	2
Iris	150	3	4
Seeds	210	3	7
Heart-statlog	270	2	12
DCCC	3000	2	23
Ionosphere	351	2	34

The experiment is conducted under Windows 10 64-bit 32GB operating memory and an Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz processor environment. We use the accuracy index to evaluate the algorithm performance, which is calculated as follows:

$$Acc = \frac{|U_t| - e}{|U_t|}. \tag{7}$$

$|U_t|$ is the size of the testing set, and e is the number of misclassified instances. If the active learner queries N labels, $|U_t| = n - N$. Otherwise, $|U_t| \geq n - N$, indicating that the active learner chooses to predict more labels than expected.

4.1 Comparison with supervised classifiers

The AAKC algorithm is an active learning algorithm for classification, we compare it with 10 supervised learning algorithms on nine datasets. (K Nearest Neighbors (KNN) [19], C4.5 [20], Naive Bayes (NB) [21], Random Forest (RF) [22], AdaBoostM1 (ABM) [23], Classification Via Regression (CVR) [24], Logit Boost (LB) [25], Bagging [26], Multi Class Classifier (MCC) [27], and Filtered Classifier (FC) [28]) These algorithms are implemented in the Weka platform [29]. All datasets are normalized, and the number of labels N and the training set size are both $0.1|U|$.

Table 4 lists the experimental results, with the best results obtained in bold. Among the ten algorithms, AAKC has the highest accuracy in six datasets, with the highest average precision of 0.9227 and the lowest average ranking of 2.11. It is worth noting that AAKC achieved an accuracy of 1.000 on Spiral and Flame datasets that the other nine algorithms cannot achieve. AAKC is superior to the classical supervised classification algorithm.

We select three classical supervised learning algorithms (KNN, C4.5, and NB) for further experimental analysis of label number N. During the experiment, the number of labels N and the size of the training set range from

Table 4 Classification accuracy of the supervised learning algorithm

Dataset	KNN	C4.5	NB	RF	ABM	CVR	LB	Bagging	MCC	FC	AAKC
Aggregation	**0.9985**	0.9732	0.9957	0.9957	0.5133	0.9774	0.9605	0.9971	0.9449	0.9704	0.9972
Spiral	0.8106	0.4510	0.3735	0.6267	0.3646	0.3842	0.6367	0.4502	0.3453	0.3442	**1.0000**
Flame	0.9722	0.8657	0.9583	0.8935	0.8796	0.8703	0.8889	0.8657	0.8657	0.8703	**1.0000**
R15	0.9814	0.7389	0.8944	0.8833	0.1203	0.7129	0.7500	0.7389	0.6352	0.8296	**0.9982**
Iris	0.9037	0.9185	0.8963	0.8889	0.9259	0.7037	0.9259	0.4974	0.8593	0.8889	**0.9710**
Seeds	0.8824	0.8122	0.8366	0.8425	0.8212	0.8138	0.8532	0.8076	0.8750	0.8255	**0.9275**
Heart-statlog	0.7477	0.7329	0.7865	0.7860	0.7572	0.7292	0.7514	0.7269	0.7329	0.7205	**0.8015**
DCCC	0.7125	0.7639	0.5479	0.7931	0.7929	0.7821	0.7879	0.7948	0.7803	**0.7957**	0.7663
Ionosphere	0.7930	0.8288	0.8047	**0.8820**	0.8253	0.8094	0.8499	0.7999	0.7848	0.8336	0.8430
Mean Acc	0.8669	0.7872	0.7882	0.8435	0.6667	0.7537	0.8227	0.7421	0.7581	0.7865	**0.9227**
Mean Rank	4.33	6.67	5.56	3.78	6.56	7.33	4.44	7.22	8.00	6.56	**2.11**

$0.01|U|$ to $0.1|U|$. Figure 4 presents the experimental results in the form of a broken line graph. The x-axis represents the number of labeled samples or the size of the training set, and the y-axis represents the classification accuracy of the algorithm. The AAKC algorithm has the following advantages over the other algorithms. Firstly, AAKC is the fastest to peak accuracy in seven data sets. It is only slightly slower than NB and KNN algorithms on flame and ionosphere datasets. AAKC only queries the labels of $0.01|U|$ samples on the six datasets, and the accuracy almost reaches the peak. Secondly, on all data sets, the accuracy of the AAKC algorithm increases steadily with increasing N, while the accuracy of the other three algorithms fluctuates. For example, on the DCCC dataset, when the training set size increases from $0.01|U|$ to $0.02|U|$, the performance of the NB algorithm decreases by 0.0848. In the Spiral dataset, when the training set size increases from $0.05|U|$ to $0.06|U|$, the prediction accuracy of the C4.5 algorithm decreases by 0.2279. In the Seeds dataset, when the training set size increases from $0.06|U|$ to $0.07|U|$, the prediction accuracy of the C4.5 algorithm decreases by 0.0791.

4.2 Comparison with active learning classifiers

In this section, we verify the effectiveness of the adaptive instance selection strategy. We compare the AAKC algorithm with eight state-of-the-art active learning algorithms, including the committee-based algorithms QBC [14] and KQBC [15], the uncertain sampling algorithm MAED [30], and the cluster-based algorithms QUIRE [13], ABD [31], ALEC [10], TASC [16], and ALTA [17]. They are the most popular approaches for active learning. code and optimal parameters of all comparison algorithms are from the author.

Table 5 shows the experimental results of the AAKC and the other eight active learning algorithms when the number of labels is $0.1|U|$, and the bold numbers indicate the best results obtained. The AAKC algorithm achieves the highest accuracy on seven data sets. On the DCCC and Ionosphere datasets, it is only slightly lower than the QBC algorithm on DCCC and Ionosphere datasets. The mean accuracy of AAKC is 0.9227, and the mean rank is 1.44. Compared with ALEC, the mean accuracy and the mean rank are increased by 0.0393 and 2.56, respectively.

Figure 5 compares the accuracy of the AAKC algorithm with eight state-of-the-art active learning algorithms. For all datasets, the proportion of queries ranges from $0.01|U|$ to $0.1|U|$. We observed that in nine datasets, the AAKC algorithm has the following advantages over the other algorithms. Firstly, the AAKC algorithm achieves peak accuracy using fewer labels than other active learning algorithms.

In addition, AAKC performs exceptionally well on the Aggregation, Spiral, Seeds, Heart-stat log, and Ionosphere datasets. When the proportion of queries is $0.02|U|$, the accuracy is almost at the peak. From the perspective of minimizing query costs, there is no need to continue to query instances. Secondly, the accuracy of AAKC maintains a steady increase as N increases. In contrast, the accuracy of QUIRE and MAED fluctuates. For example, on the DCCC dataset, the accuracy of the QUIRE algorithm fluctuates constantly. When the roportion of queries is $0.02|U|$, the accuracy rate reaches the maximum value, and then the accuracy rate decreases slightly as the number of tags increases. On the Heart-stat log and Seeds datasets, when the number of labels increases from $0.04|U|$ to $0.05|U|$, the accuracy of the MAED algorithm decreases by 0.0774 and 0.0656, respectively. AAKC is consistently more accurate

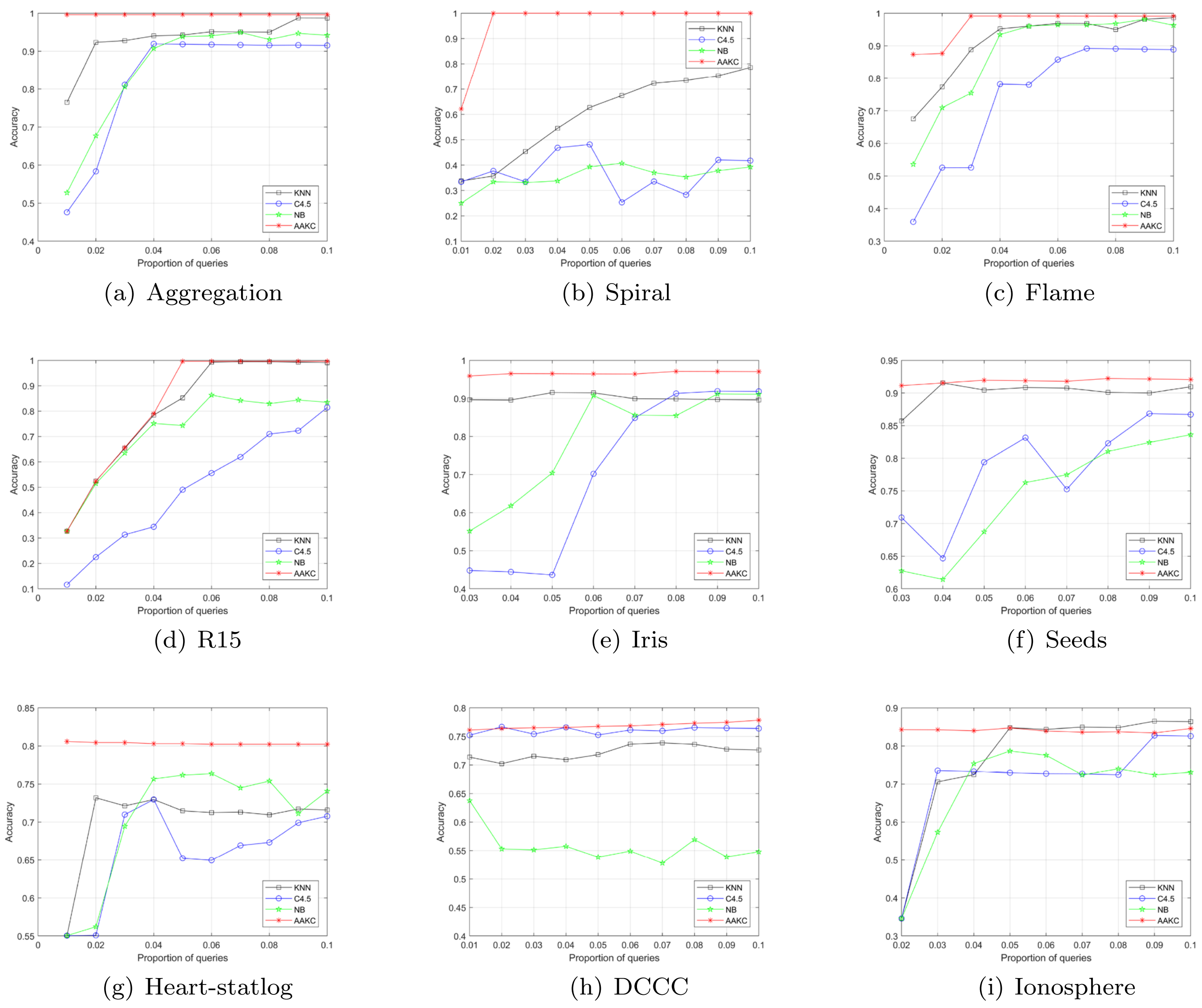

Fig. 4 Comparison of classification accuracy with supervised learning algorithm

Table 5 Classification accuracy of the active learning algorithm

Dataset	ABD	QBC	KQBC	MADE	QUIRE	ALEC	TACS	ALTA	AAKC
Aggregation	0.9964	0.8922	0.7023	0.9944	0.9971	0.9506	0.9872	0.9855	**0.9972**
Spiral	0.9326	0.7918	0.3309	0.7393	0.7295	**1.0000**	0.8089	0.6093	**1.0000**
Flame	0.9791	0.9513	0.7383	0.9861	0.9537	**1.0000**	0.9704	0.9191	**1.0000**
R15	0.9812	0.3583	0.7674	0.9907	0.9778	0.9962	0.9787	0.9808	**0.9982**
Iris	0.9600	0.9666	0.9462	0.9185	0.9407	0.9629	0.9096	0.8000	**0.9710**
Seeds	0.8928	0.9206	0.9027	0.8571	0.8624	0.8835	0.8545	0.9045	**0.9275**
Heart	0.7292	0.7514	0.7269	0.7329	0.7205	0.7654	0.7564	0.7373	**0.8015**
DCCC	0.7770	**0.7977**	0.7761	0.7030	0.7415	0.7577	0.7044	0.6729	0.7663
Ionosphere	0.8094	**0.8499**	0.7999	0.7848	0.8336	0.6347	0.8127	0.7464	0.8430
Mean acc	0.8953	0.8089	0.7434	0.8563	0.8619	0.8834	0.8647	0.8173	0.9227
Mean rank	3.89	4.11	6.56	5.56	5.67	4.00	5.44	6.56	1.44

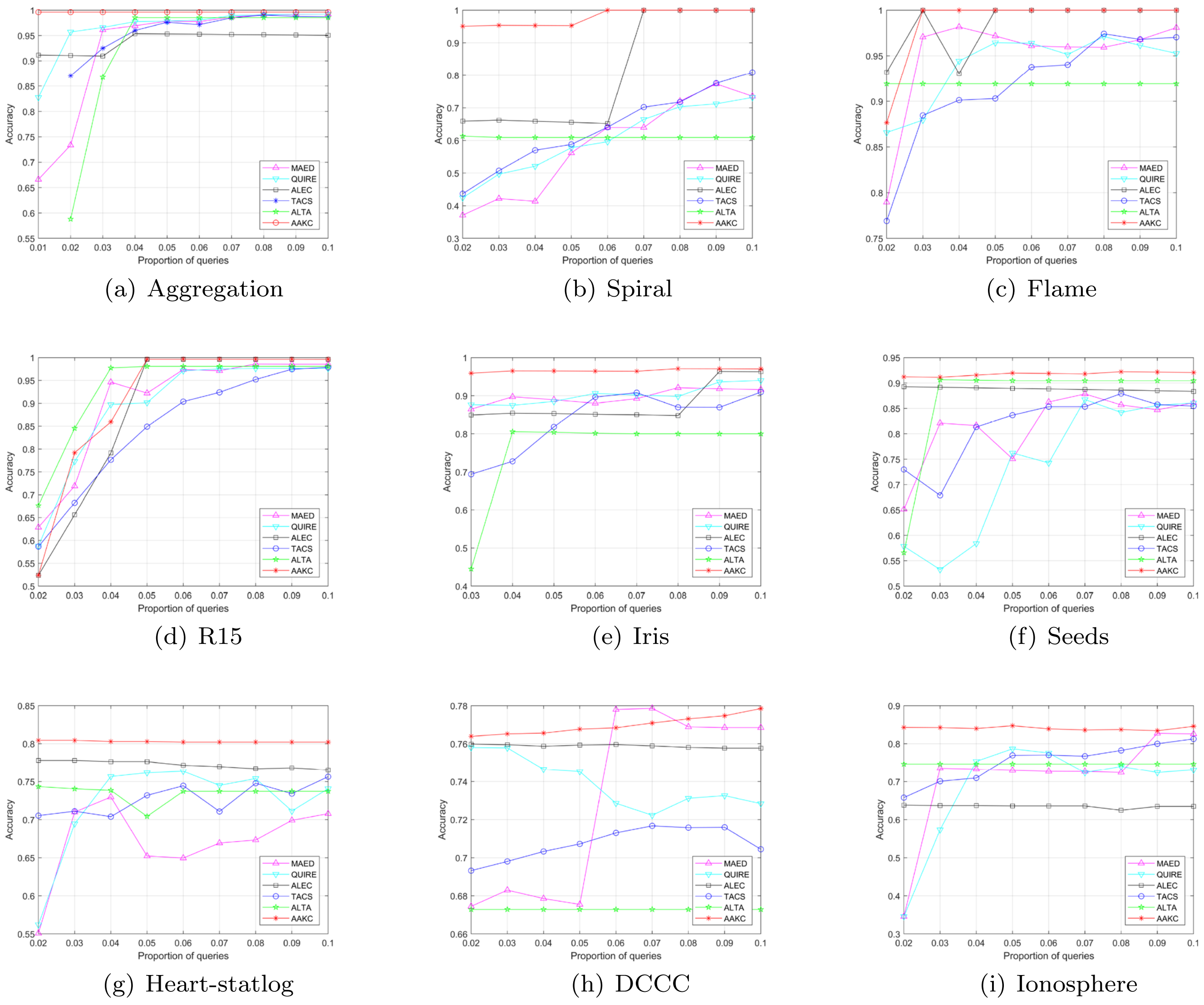

Fig. 5 Comparison of classification accuracy with active learning algorithm

and more consistent than ALEC. In conclusion, the AAKC algorithm is superior to other active learning algorithms in general.

4.3 Effects of parameters

The critical parameter in AAKC is k, which determines the neighbors of each instance. To evaluate the effect of k and the stability of our algorithm, we set k from 3 to 10. Figure 6 presents the results of the experiment, and Table 6 shows the statistics of the experimental results. We observe that AAKC is very robust concerning the parameter k. Additionally, although the value of k is not always the same when the AAKC algorithm obtains the highest accuracy on different datasets, it is always in the range of 3 to 6. This proves the stability of our algorithm with respect to the parameter k.

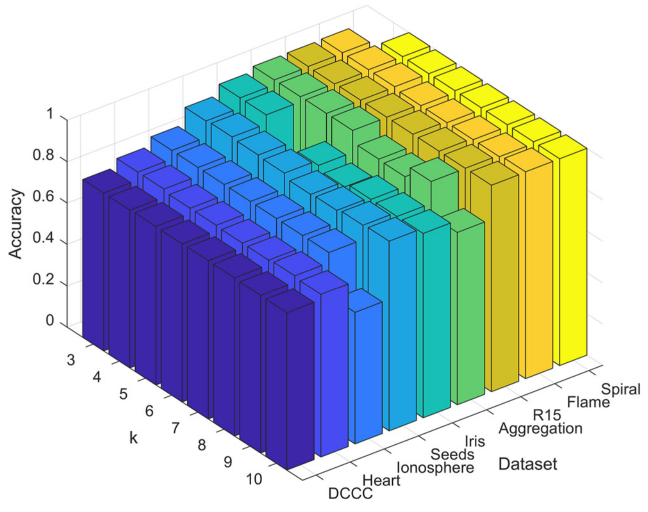

Fig. 6 The experimental results of AAKC on different k

Table 6 The effects of k on nine datasets, where k ranges from 3 to 10, and the "k" column represents the k value for maximum accuracy

Dataset	Max	Min	Mean	Var	k
Aggregation	0.9972	0.8331	0.9955	0.003368	5
Spiral	1.0000	0.6063	0.9508	0.019378	5
Flame	1.0000	0.9910	0.9617	0.000023	5
R15	0.9982	0.9946	0.9975	0.000002	5
Iris	0.9710	0.6522	0.8786	0.009868	4
Seeds	0.9275	0.8964	0.9145	0.000115	5
Heart-statlog	0.8015	0.7871	0.7928	0.000037	4
DCCC	0.7663	0.7612	0.7640	0.000003	5
Ionosphere	0.8430	0.6366	0.8172	0.005325	5

5 Conclusion

This paper proposed the AAKC algorithm, which uses the k-nearest neighbor information to redefine the local density of instances and adopts an adaptive instance selection strategy to select samples automatically. Experimental results on nine datasets confirm that our algorithm is better than the classic supervised learning algorithm and the latest active learning algorithm. The time complexity of AAKC is $O(mn^2)$, which is the same as that of DPC algorithms. However, it is not efficient enough when facing high-dimensional and large datasets. In future research, we plan to further improve the efficiency of the algorithm.

Acknowledgements This work was supported in part by the Natural Science Foundation of China under Grant 61972001, in part by the General Project of Anhui Natural Science Foundation under Grant 1908085MF188 and 2108085MF212, and in part by the Key Projects of Natural Science Foundation of Anhui Province Colleges and Universities under Grant KJ2020A0041.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Li Y, Fan B, Zhang W, Ding W, Yin J (2021) Deep active learning for object detection. Inf Sci 579:418–433
2. Deng C, Liu X, Li C, Tao D (2018) Active multi-kernel domain adaptation for hyperspectral image classification. Pattern Recogn 77:306–315
3. Cao X, Yao J, Xu Z, Meng D (2020) Hyperspectral image classification with convolutional neural network and active learning. IEEE Trans Geosci Remote Sens 58(7):4604–4616
4. Haut JM, Paoletti ME, Plaza J, Li J, Plaza A (2018) Active learning with convolutional neural networks for hyperspectral image classification using a new Bayesian approach. IEEE Trans Geosci Remote Sens 56(11):6440–6461
5. Kansizoglou I, Bampis L, Gasteratos A (2019) An active learning paradigm for online audio-visual emotion recognition. IEEE Trans Affect Comput 13(2):756–768
6. Reyes O, Ventura S (2018) Evolutionary strategy to perform batch-mode active learning on multi-label data. ACM Trans Intell Syst Technol 9(4):1–26
7. Guo J, Pang Z, Bai M, Xie P, Chen Y (2021) Dual generative adversarial active learning. Appl Intell 51(8):5953–5964
8. McCallumzy AK, Nigamy K (1998) Employing EM and pool-based active learning for text classification. In: Proceedings of the international conference on machine learning, pp 359–367
9. Dasgupta S, Hsu D (2008) Hierarchical sampling for active learning. In: Proceedings of the 25th international conference on machine learning, pp 208–215
10. Wang M, Min F, Zhang ZH, Wu YX (2017) Active learning through density clustering. Expert Syst Appl 85:305–317
11. Xie J, Gao H, Xie W (2016) K-nearest neighbor optimized density peak fast searching clustering algorithm. Chin Sci Inf Sci 46(2):258–280
12. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
13. Huang SJ, Jin R, Zhou ZH (2010) Active learning by querying informative and representative examples. Adv Neural Inf Process Syst 23:892–900
14. Seung HS, Opper M, Sompolinsky H (1992) Query by committee. In: Proceedings of the fifth annual workshop on computational learning theory, pp 287–294
15. Gilad-Bachrach R, Navot A, Tishby N (2003) Kernel query by committee (KQBC). Leibniz Cent Hebr Univ Jerus Israel Tech Rep 88:2004
16. Min F, Zhang SM, Ciucci D, Wang M (2020) Three-way active learning through clustering selection. Int J Mach Learn Cybern 11(5):1033–1046
17. Wang M, Zhang YY, Min F, Deng LP, Gao L (2020) A two-stage density clustering algorithm. Soft Comput 24:17797–17819
18. Blake C (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository. Accessed 01 Dec 2021
19. Han J, Pei J, Tong H (2022) Data mining: concepts and techniques. Morgan Kaufmann
20. Xiang Z, Zhang L (2012) Research on an optimized C4. 5 algorithm based on rough set theory. In: 2012 international conference on management of e-commerce and e-government, pp 272–274
21. Rish I (2001) An empirical study of the naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, vol 3. no. 22, pp 41–46
22. Liaw A, Wiener M (2002) Classification and regression by randomForest. R News 2(3):18–22
23. Cortes EA, Martinez MG, Rubio NG (2007) Multiclass corporate failure prediction by Adaboost. M1. Int Adv Econ Res 13(3):301–312

24. Ruan YX, Lin HT, Tsai MF (2014) Improving ranking performance with cost-sensitive ordinal classification via regression. Inf Retr 17(1):1–20
25. Cai YD, Feng KY, Lu WC, Chou KC (2006) Using LogitBoost classifier to predict protein structural classes. J Theor Biol 238(1):172–176
26. Quinlan JR (1996) Bagging, boosting, and C4. 5. In: AAAI/IAAI, vol 1. pp 725–730
27. Afshar S, Mosleh M, Kheyrandish M (2013) Presenting a new multiclass classifier based on learning automata. Neurocomputing 104:97–104
28. Suoliang Z, Tianshu Z, Ming L, Kunlun L, Baozong Y (2010) An experimental study of classifier filtering, 361–364
29. Frank E, Hall MA, Witten IH (2016) The WEKA Workbench. Online appendix for "Data mining: practical machine learning tools and techniques", Morgan Kaufmann, Fourth Edition, 2016
30. Cai D, He X (2011) Manifold adaptive experimental design for text categorization. IEEE Trans Knowl Data Eng 24(4):707–719
31. Munoz-Mari J, Tuia D, Camps-Valls G (2012) Semisupervised classification of remote sensing images with active queries. IEEE Trans Geosci Remote Sens 50(10):3751–3763

Xia Ji (1982.11) Ph.D., lecturer at the School of Computer Science and Technology, Anhui University. Her research interest covers rough set theory and granular computing.

WanLi Ye (1998,5) is currently pursuing the master's degree with the School of Computer Science and Technology, Anhui University. His current research interest includes active learning, clustering and machine learning.

XueJun Li received the Ph.D. degree from Anhui University, Hefei, China, in 2008. He is currently a Professor with the School of Computer Science and Technology, Anhui University. His major research interests include workflow systems, cloud computing, and intelligent software.

Peng Zhao Ph.D., associate professor at the School of Computer Science and Technology, Anhui University. Her research interest covers machine learning and intelligent information processing.

Sheng Yao (1979.10) Ph.D., lecturer at the School of Computer Science and Technology, Anhui University. Her research interest covers rough set theory and granular computing.