International Journal of Machine Learning and Cybernetics (2020) 11:1033–1046
https://doi.org/10.1007/s13042-020-01099-2

ORIGINAL ARTICLE

Three-way active learning through clustering selection

Fan Min[1] · **Shi-Ming Zhang**[1] · **Davide Ciucci**[2] · **Min Wang**[3]

Received: 24 September 2019 / Accepted: 17 February 2020 / Published online: 3 March 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract
In clustering-based active learning, the performance of the learner relies heavily on the quality of clustering results. Empirical studies have shown that different clustering techniques are applicable to different data. In this paper, we propose the three-way active learning through clustering selection (TACS) algorithm to dynamically select the appropriate techniques during the learning process. The algorithm follows the coarse-to-fine scheme of granular computing coupled with three-way instance processing. For label query, we select both representative instances with density peaks, and informative instances with the maximal total distance. For block partition, we revise six popular clustering techniques to speed up learning and accommodate binary splitting. For clustering evaluation, we define weighted entropy with 1-nearest-neighbor. For insufficient labels, we design tree pruning techniques with the use of a block queue. Experiments are undertaken on twelve UCI datasets. The results show that TACS is superior to single clustering technique based algorithms and other state-of-the-art active learning algorithms.

Keywords Active learning · Clustering · Granular computing · Three-way decision

1 Introduction

In many real-world applications such as image classification [1], information extraction [2] and text classification [3], unlabeled data are abundant and easy to obtain, while labels are costly. Active learning [4, 5] aims to achieve higher classification accuracy with fewer labels through human–computer interaction. The key issue is to select critical instances to label. There are at least two views and respective approaches. One view is that uncertain instances are critical. Typical approaches include query-by-committee [6], fuzzy-rough based [7], and ambiguity-based [8] algorithms. The other view is that representative instances are critical. The most typical approaches are clustering-based [9, 10] algorithms. Naturally, combination approaches [11, 12] take advantage of both.

Three-way active learning [10, 13, 14] is a new type of clustering-based [9] algorithm. From the viewpoint of three-way decision [15, 16], there are three possible actions for each block. In case that there are not enough labeled data, representative instances of the block are queried. In case that there are enough labeled data with the same label, other instances in the block are classified. In case that there are labeled data with different labels, the block is clustered for further treatment. Since the process is iterative, the paradigm can be better represented by sequential three-way decision [17–19]. From the viewpoint of granular computing [20, 21], the learning process follows the coarse-to-fine scheme [22–24]. Instances are queried or classified in the appropriate granule, which is represented by a block.

Unfortunately, the performance of three-way active learning relies heavily on the base clustering technique. Different clustering techniques are applicable to different data. This is also a common situation for clustering-based active learning algorithms [5]. A natural question is: given

✉ Fan Min
minfan@swpu.edu.cn

Shi-Ming Zhang
zhangshiming@stu.swpu.edu.cn

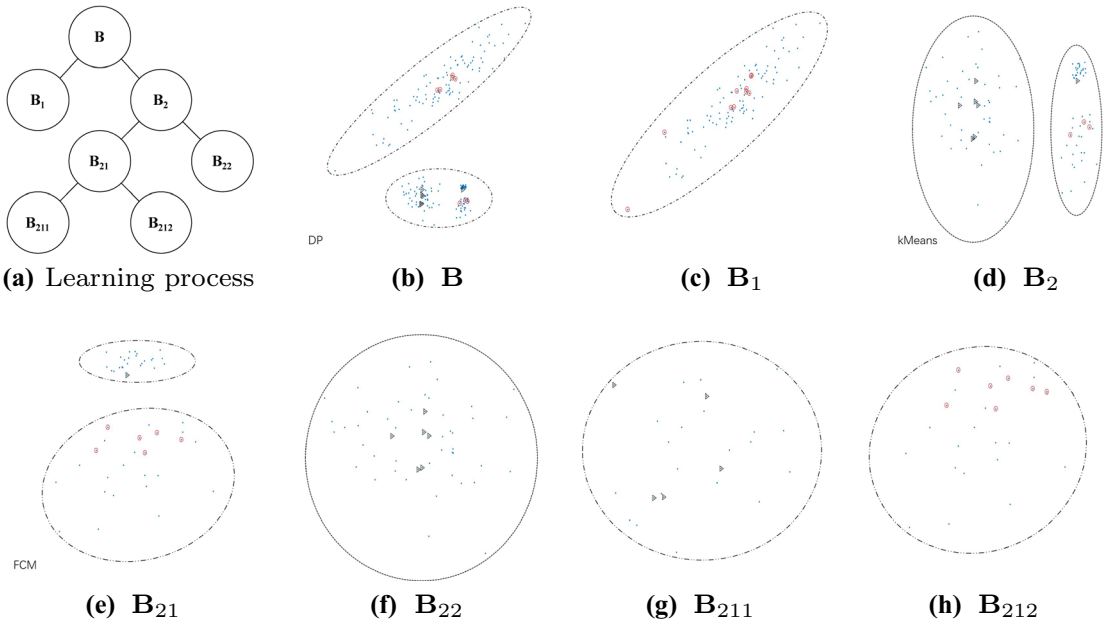
Davide Ciucci
davide.ciucci@unimib.it

Min Wang
wangmin@swpu.edu.cn

[1] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China

[2] DISCo, University of Milano-Bicocca, viale Sarca 336/14, 20126 Milan, Italy

[3] School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China



(a) Learning process **(b)** $\mathbf{B}$ **(c)** $\mathbf{B}_1$ **(d)** $\mathbf{B}_2$

(e) $\mathbf{B}_{21}$ **(f)** $\mathbf{B}_{22}$ **(g)** $\mathbf{B}_{211}$ **(h)** $\mathbf{B}_{212}$

Fig. 1 A running example of TACS **a** depicts the general clustering and learning process. **b**, **d**, **e** depict the block and the best clustering technique. **c**, **f**, **g**, **h** depict the final blocks. Triangles indicate queried positive labels, and circles indicate queried negative labels

a dataset, can we find appropriate clustering techniques in the learning process?

In this paper, we propose the three-way active learning through clustering selection (TACS) algorithm for this issue. Figure 1 illustrates a running example of the new algorithm. Some representative instances in the whole block $\mathbf{B}$ are queried at the beginning. Since there are both positive and negative labels, $\mathbf{B}$ is split in $\mathbf{B}_1$ and $\mathbf{B}_2$ using the currently best algorithm (Density Peaks, DP [25]). Then a few more instances are queried for $\mathbf{B}_1$. Since there are only positive labels, the remaining instances are classified as positive. More instances are queried for $\mathbf{B}_2$ as well. With both positive and negative labels, $\mathbf{B}_2$ is split into $\mathbf{B}_{21}$ and $\mathbf{B}_{22}$ using the currently best algorithm (kMeans). This process repeats until all instances are either queried or classified. Note that the whole process is similar to a decision tree, but the splitting technique is different. Moreover, the binary split structure works for any datasets, not only for those with binary classes.

The contribution of the paper is to address four issues of our algorithm. The first issue is: which instances should be queried? This is also the key issue of any active learning algorithm. In the beginning, we select some representative instances to query. Here representativeness is measured by both density and distance to the closest instance with higher density [10]. These instances also help clustering technique selection. During the iteration, we select either representative or informative instances. Here information is measured by the total distance to labeled instances of the same block. In this way, the number of queries is also controlled.

The second issue is: how to revise existing clustering techniques to suit our algorithm? Since our algorithm always split a block in two, general clustering techniques should be revised. For both kMeans [26] and fuzzy c-means (FCM) [27], we choose a pair of instances with large distance as the original centers. For density peaks (DP) [25], we choose the top two representative instances with different labels for splitting. For Hierarchical [28], the solution is straightforward since the cluster tree is built in a bottom-up manner. For DBScan [29], we adjust the density threshold such that at least one core exists. For random walks (RW) [30], the largest block does not change, and the other instances form the second block. We also revise these techniques to speed up the learning process.

The third issue is: how to select the most appropriate technique to cluster the current block? Naturally, this is the key issue of this paper. We design weighted entropy with 1-nearest-neighbor. It considers both labeled and unlabeled data. Labeled data are trustworthy, hence their weight is 1. Unlabeled data are first classified by 1-nearest-neighbor to obtain pseudo-labels, whose weight is less than 1. In this way, the weighted entropy can be calculated. Moreover, we design the retrospective technique to recalculate the weighted entropy after obtaining more labels.

The fourth issue is: how to deal with the lack of labels? In real applications, the expert usually cannot provide enough queries due to limited budget. Consequently, we would like to query labels in larger blocks, which is more appropriate for the clustering-based algorithm. For this purpose, we design the breadth-first visit of the tree

Table 1 Notations

Notation	Meaning	Comments
$\mathbf{X} = (x_{ij})_{n\times m}$	The data	Input/data model
$\mathbf{x}_i$	The ith instance	Data model
$\mathbf{y} = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$	Class labels	Data model
c	The number of classes	Data model
$\mathbf{U} = [1..n]$	The whole dataset	Data model
$\mathbf{B}, \mathbf{B}_1, \mathbf{B}_2 \subseteq \mathbf{U}$	Blocks in the learning process	Algorithm variable
$\mathbf{N}_i$	The neighborhood of $\mathbf{x}_i$	Algorithm variable
ρ_i	The local density of $\mathbf{x}_i$	Algorithm variable
δ_i	The distance to its master of $\mathbf{x}_i$	Algorithm variable
γ_i	The representativeness of $\mathbf{x}_i$	Algorithm variable
$\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^{\mathrm{T}}$	The queried/classified labels	Algorithm output
ST	Small block threshold	Algorithm setting
rt	Cutoff distance ratio	Algorithm setting
d_c	Cutoff distance	Algorithm setting
qi	The proportion of queried instances	Algorithm input
qr	The proportion of representative instances	Algorithm setting
$\mathbf{Q}$	The set of queried instances	Algorithm output

depicted in Fig. 1h. This is implemented by a block queue, in which blocks smaller than a threshold are discarded. Compared with the depth-first visit, different blocks of the same level have more balanced queries. When the budget runs off, only some instances in relatively small blocks remain unclassified. At last, they are classified using kNN.

Experiments are undertaken on twelve UCI datasets. Results show that in most cases, TACS can find out appropriate base clustering techniques. It is more accurate than supervised classification algorithms such as C4.5 [31] and Naïve Bayes (NB) [32], active learning algorithms such as Query-by-committee (QBC) [6] and manifold adaptive experimental design (MAED) [33], and single clustering technique based algorithms such as ALEC [10].

The rest of the paper is organized as follows. Section 2 introduces the basis of our algorithm, including the problem statement and revisions to some clustering techniques. Section 3 presents our algorithmic framework and some key issues along with corresponding techniques. Section 4 describes the experimental process and lists some results. Section 5 discusses some related work. Finally, Sect. 6 presents conclusions and outlines further research trends.

2 Basis

This section presents some basis of the work, including the data model, the problem statement, and some issues of the clustering techniques. Table 1 lists some notations used throughout the paper.

2.1 The data model

Let $\mathbf{X} = (x_{ij})_{n\times m}$ be the data matrix, where n is the number of instances, m is the number of conditional attributes, and $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ is the ith instance. Let further $\mathbf{y} = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ be the class label vector. In our active learning scenario, the value of $y_i \in [1..c]$ $(1 \le i \le n)$ is unknown. It should be either queried or classified.

The set of all instances is $\mathbf{U} = \{\mathbf{x}_i | 1 \le i \le n\}$. For brevity, we also denote it using the indices, i.e., $\mathbf{U} = [1..n]$. In this way, any block is a subset of $[1..n]$.

2.2 Problem statement

Since a query requires an oracle's effort, we assume that the number of queries is limited. The problem, which is the same as that of [10], is stated as follows.

Problem 1 Active learning with fixed number of labels.
Input: The data $\mathbf{X}$, and the proportion of queried instances qi.
Output: The set of queried instances $\mathbf{Q} \subset \mathbf{U}$, the classified labels for $\mathbf{U} \setminus \mathbf{Q}$.
Optimization objective: Maximize the classification accuracy.

First, we assume that no label is known at the beginning of the learning process. In this way, it is easy to compare the classification accuracy of different algorithms. Second, it is required that $|\mathbf{Q}| \le \lfloor n \times qi \rfloor$. When $|\mathbf{Q}| < \lfloor n \times qi \rfloor$, the queries are not used up. This situation rarely occurs in

applications because the number of queries is usually insufficient. Third, the classification accuracy only accounts for classified instances.

2.3 Revised clustering techniques

Our algorithm is based on six popular clustering techniques. These include two prototype-based techniques (*k*Means [26] and FCM [27]), two density-based techniques (DBScan [29] and DP with Guassian kernel [25]), one hierarchical technique (Hierarchical [28]), and one graph-based technique (RW [30]). Instead of explaining them from the very beginning, we will focus on the following issues:

(1) How to revise the techniques for binary splitting shown in Fig. 1b?
(2) For big datasets, how to decrease the space and/or time complexity?
(3) How to initialize clustering centers if needed? and
(4) How to balance clusters if needed?

In the following context, the current block will be denoted by $\mathbf{B}$, and the sub-blocks will be denoted by $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively.

2.3.1 Prototype-based techniques

*k*Means and fuzzy c-means (FCM) are popular prototype-based clustering techniques. Both require some initial centers, which often affect the results. Since two initial centers are required, we would like to choose the pair of instances with the maximal distance for this purpose. That is

$$(\mathbf{x}_{i^*}, \mathbf{x}_{j^*}) = \underset{(i,j)\in\mathbf{B}\times\mathbf{B}}{\operatorname{argmax}} d_{ij}, \tag{1}$$

where d_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. The time complexity of computing d_{ij} is $O(m)$ if we use simple measures such as Manhattan and Euclidean distance. Hence the time complexity of Eq. (1) is $O(m|\mathbf{B}|^2) = O(mn^2)$, which is rather high.

In reality, a pair of far enough instances usually helps to obtain good clusters. It is unnecessary to find the instance pair with the maximal distance. Here we design an alternative approach. We randomly choose some pairs of instances from $\mathbf{B}$ to form

$$\mathcal{P} \subseteq \mathbf{B} \times \mathbf{B}. \tag{2}$$

The pair with the maximal distance in $\mathcal{P}$ is

$$(\mathbf{x}_{i^*}, \mathbf{x}_{j^*}) = \underset{(i,j)\in\mathcal{P}}{\operatorname{argmax}} d_{ij}. \tag{3}$$

In this way, the time complexity is reduced to $O(m|\mathcal{P}|)$. In our experimentation, $|\mathcal{P}| = 10|\mathbf{B}|$, and the time complexity is $O(mn)$.

2.3.2 Density-based techniques

Density-based techniques [25, 29] require the computation of the density of each instance. There are two popular kernels, one is cutoff, and the other is Gaussian. With the cutoff kernel, the neighborhood of $\mathbf{x}_i$ is

$$\mathbf{N}_i = \{j \neq i | d_{ij} \leq d_c\}, \tag{4}$$

where d_c is the neighborhood distance threshold. The local density of $\mathbf{x}_i$ is

$$\rho_i = |\mathbf{N}_i|, \tag{5}$$

where $|\cdot|$ indicates the cardinality of a set.

With the Gaussian kernel, the local density of $\mathbf{x}_i$ is redefined as

$$\rho_i = \sum_{j\neq i} e^{-\left(\frac{d_{ij}}{d_c}\right)^2}. \tag{6}$$

The cutoff kernel only considers the neighborhood, while the Gaussian kernel mainly considers the neighborhood. The cutoff kernel produces an integer value density, while the Gaussian kernel produces a real value density. The cutoff kernel often produces the same density for different instances, making the master tree sensitive to data order, which is rarely faced by the Gaussian kernel.

Now we consider the time complexity of density computation. The time complexity of computing d_{ij} is $O(m)$. According to Eqs. (4), (5) and (6), the time complexity of computing the instance density is $O(mn)$ with either kernel. To compute the density of all instances, the time complexity will be $O(mn^2)$, which is rather high.

In applications, there is essentially no need to consider instances far from the current one. Hence we propose the following approach. First, the dataset is clustered into $\lfloor n/n' \rfloor$ blocks using the *k*Means algorithm. Since *k*Means only iterates a few rounds, the time complexity is $O(mnn/n') = O(mn^2/n')$. Second, when computing the density of an instance, we only consider instances in the same block. The time complexity is $O(mn')$. So the time complexity of computing the density of all instances is $O(mnn')$. If we set $n' = \sqrt{n}$, the total time complexity will be reduced to $O(mn(n/n' + n')) = O(mn^{1.5})$.

With the cutoff kernel, this acceleration method may produce a density value that is slightly different from the original definition. With the Gaussian kernel, there will always be small differences. Fortunately, for our learning task, such

difference rarely influences the final results especially when n is big.

Now we discuss some implementation issues of DBScan. Instance $\mathbf{x}_i$ is called a *core* iff

$$\rho_i \geq MinPts, \tag{7}$$

where $MinPts$ is the density threshold.

The setting of d_c and $MinPts$ is an important issue. In our implementation, $MinPts = \sqrt{|\mathbf{B}| + 1}$ where $\mathbf{B}$ is the current block. Similar to Eq. (3), to save the runtime, we set

$$d_c = rt \times \max_{(i,j)\in\mathcal{P}} d_{ij}, \tag{8}$$

where $rt \in (0, 1)$ is a distance ratio with an initial value such as 0.1. If there is no core, we will increase rt to $1.5rt$ until at least one core exists.

DBScan generates a number of clusters. To obtain exactly two clusters, the largest cluster forms $\mathbf{B}_1$, and $\mathbf{B}_2 = \mathbf{B} \setminus \mathbf{B}_1$.

For density peaks [25], we have some more issues to handle. Let $ms_i = \{j|\rho_j > \rho_i\}$ be the set of instances with higher density than $\mathbf{x}_i$. The master of $\mathbf{x}_i$ is

$$m_i = \operatorname*{argmin}_{j\in ms_i} d_{ij}. \tag{9}$$

With m_i, we organize all instances using a tree, where the master of $\mathbf{x}_i$ is its parent. This tree, which will be called the *master tree*, will be used in the algorithm. The distance to master is

$$\delta_i = d_{im_i} = \min_{j\in ms_i} d_{ij}. \tag{10}$$

The representativeness of $\mathbf{x}_i$ is measured by [10]

$$\gamma_i = \rho_i \delta_i. \tag{11}$$

To split $\mathbf{B}$ into $\mathbf{B}_1$ and $\mathbf{B}_2$, we sort labeled instances in $\mathbf{B}$ in descending order according to γ_i. Let the label of the first instance be lf. Let further the first instance with label different from lf be $\mathbf{x}_f$. $\mathbf{x}_f$ and all its offsprings in $\mathbf{B}$ form $\mathbf{B}_1$. Naturally, $\mathbf{B}_2 = \mathbf{B} \setminus \mathbf{B}_1$.

Note that for each block, we will build a new master tree. However, the density of each instance will not be recomputed. The reason is that pairwise distance computation is very time consuming, and recomputing is unnecessary.

We have tested two versions of the density peaks clustering technique based on cutoff and Gaussian kernels, respectively. It is observed that the cutoff kernel rarely defeats the Gaussian kernel. Hence we will only use the Gaussian kernel in our experimentation.

2.3.3 Hierarchical technique

The hierarchical clustering technique works in a bottom-up manner. Let $\mathbf{B}_i$ and $\mathbf{B}_j$ be two blocks, and their centers be $\mathbf{x}'_i$ and $\mathbf{x}'_j$, respectively. Here $\mathbf{x}'_i$ and $\mathbf{x}'_j$ may not be true data points. The distance between $\mathbf{B}_1$ and $\mathbf{B}_2$ is defined as the distance between $\mathbf{x}'_i$ and $\mathbf{x}'_j$.

Let $\mathcal{B} = \{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{n'}\}$ be the current round set of blocks. At the beginning, $\mathbf{B}_i = \{\mathbf{x}_i\}$ for $1 \leq i \leq n$ and $n' = n$. Let $\mathcal{B}'$ be the next round set of blocks initialized as $\emptyset$. One round of merging repeats the following two steps $\lfloor \frac{n'}{2} \rfloor$ times: Step 1. Take out $\mathbf{B}_i$ and $\mathbf{B}_j$ from $\mathcal{B}$ such that their distance is minimal. $\mathcal{B} = \mathcal{B} \setminus \{\mathbf{B}_i, \mathbf{B}_j\}$. Step 2. $\mathcal{B}' = \mathcal{B}' \cup \{\mathbf{B}_i \cup \mathbf{B}_j\}$. If $\lfloor \frac{n'}{2} \rfloor \neq \lceil \frac{n'}{2} \rceil$, the last element of $\mathcal{B}$ is added to $\mathcal{B}'$ directly.

In the first round, the n nodes are merged into $\lceil \frac{n}{2} \rceil$ blocks. In the following rounds, the current n' blocks are merged into $\lceil \frac{n'}{2} \rceil$ bigger ones. In our implementation, the process stops when there are exactly 2 blocks.

2.4 Graph-based technique

In graph-based techniques, the data is organized as a weighted graph, where each edge represents a direct connection between nodes (i.e., instances). To use this technique, we should first construct a weighted graph for a data matrix, as defined in the beginning of this section. During the learning process, we should construct a weighted graph for the current block $\mathbf{B}$.

For instance $\mathbf{x}_i$, we find out its k nearest neighbors to build direct connections. In our experiments, we set $k = 2$. Let instance $\mathbf{x}_j$ be one of the k nearest neighbors. The edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ has the weight

$$w_{ij} = \begin{cases} MW, & \text{if } d_{ij} = 0; \\ 1/d_{ij}, & \text{otherwise,} \end{cases} \tag{12}$$

where MW is a constant which is set to 10,000 in our experimentation. By assigning 0 to all other weights, we obtain the initial adjacent matrix $W = (w_{ij})_{|\mathbf{B}|\times|\mathbf{B}|}$.

Next, we let $w_{ij} = \max\{w_{ij}, w_{ji}\}$. In this way, W becomes a symmetric matrix, which is the standard input for graph-based techniques.

We adopt the simple version of random walks described in [30]. The number of blocks cannot be controlled by the algorithm. Hence in our implementation, the largest block is assigned to $\mathbf{B}_1$, and the other instances form $\mathbf{B}_2 = \mathbf{B} - \mathbf{B}_1$.

3 The algorithm

In this section, we first present the algorithm framework. Then we analyze some key issues of the algorithm, including instance selection strategies, clustering quality evaluation, and other special case handling techniques.

3.1 Algorithm description

Algorithm 1 Active learning through clustering selection (TACS)

Input: A data matrix $\mathbf{X}$, cutoff distance ratio rt, the proportion of queried instances qi, the proportion of representative instances qr, small block threshold ST.
Output: Queried/classified labels $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$.

1: $\mathbf{B} = [1..n]$; // Initialized as the block containing all instances
2: $\hat{\mathbf{y}} = (-1, \ldots, -1)_n$; //Initialized as unknown
3: $N_q = \lfloor n \times qi \rfloor$; //Number of queries
4: $N_r = \lfloor N_q \times qr \rfloor$; //Number of representatives
// Step 1. Select representative instances to label
5: Compute ρ_i for $1 \le i \le n$ according to Eqs. (6) and 8;
6: Compute δ_i for $1 \le i \le n$ according to Eq. (10);
7: $\gamma \leftarrow (\gamma_1, \cdots, \gamma_N) \leftarrow \rho \cdot \delta$; //Hadamard product for representativeness
8: Query $N = N_r$ instances with the highest γ_i; // Current number of queries
//Step 2. Query informative instances and classify
9: q.enqueue($\mathbf{B}$); // The original queue with only one block
10: **while** ($q \neq \emptyset$ && $N \le N_q$) **do**
11: $\mathbf{B} = q$.dequeue(); //Take a block from the head
12: **if** ($|\mathbf{B}| \le ST$) **then**
13: continue; //Process small blocks later
14: **end if**
15: Query up to $\sqrt{|\mathbf{B}|}$ instances according to Eq. (14) or (15) and update N;
16: **if** ($\mathbf{B}$ is pure) **then**
17: For all $i \in \mathbf{B}$, l_i = the label of queried instances in the block;
18: continue; //No need to split further
19: **end if**
20: Cluster $\mathbf{B}$ to $(\mathbf{B}_1, \mathbf{B}_2)$ using the current best algorithm;
21: q.enqueue($\mathbf{B}_1$);
22: q.enqueue($\mathbf{B}_2$);
23: **end while**
24: **if** ($N < N_q$) **then**
25: query $(N_q - N)$ unlabeled instances according to γ_i;
26: **end if**
27: Classify unlabeled instances using kNN;
28: **return** $\hat{\mathbf{y}} \leftarrow [\hat{y}_i]_{n \times 1}$;

Algorithm 1 lists the TACS algorithm. The initialization is implemented in Lines 1–3. Line 1 sets the current block $\mathbf{B}$ as the whole dataset. Line 2 indicates that all instances are unlabeled. Line 3 calculates the number of queries N_q. Line 4 calculates the number of representatives N_r. The user does not specify N_q and N_r directly, since qi is easier specified by the users, and qr can be dataset independent.

Instance selection strategies are implemented in Lines 5–7, 14 and 24. Inspired by the density peaks algorithm [25], the representativeness of an instance is qualified by the multiplex of its density and distance. The informativeness of an instance is qualified by the total distance from labeled instances of the same block. These measures will be discussed in more detail in Sect. 3.2.

According to the tree structure of the learning process, there are two possible implementations. One is depth-first using a stack, while the other is breadth-first using a queue. Since there is often lack of labels, the depth-first approach tends to label small blocks of some branches. In contrast, the breadth-first approach distributes labels more evenly to different big blocks. Therefore, we choose it for balanced query. Line 9 initializes the queue. Line 10 judges whether all impure blocks are processed. Line 11 takes a block out of the queue. Lines 21–22 append two sub-blocks to the queue.

There are two pruning techniques. The first one deals with small blocks. As indicated in Lines 12–14, small blocks are abandoned in the while loop. This is because small blocks are not suitable for clustering-based active learning. The second one deals with lack of labels. As indicated by the second condition of Line 10, if the number of queries reaches N_q, remaining blocks in q are suspended. Due to these pruning techniques, some instances are still not handled. They are classified using kNN, as indicated in Line 27.

3.2 Instance selection strategies

As shown in Line 15 of the algorithm, in each round, the active learner should select a number of critical instances to query. This is done by sorting unlabeled instances according to their representativeness in descending order. In the current block $\mathbf{B}$, let the set of labeled instances be $\mathbf{B}_l$ and the other part be $\mathbf{B}_u$. We propose two strategies to calculate the representativeness of each instance $\mathbf{x} \in \mathbf{B}_u$.

The local density ρ_i of $\mathbf{x}_i \in \mathbf{B}_u$ is computed in Eq. (6). As discussed in Sect. 2.3.2, it is adopted directly for reducing time complexity. The distance to master is redefined as

$$\delta_i(\mathbf{B}) = \min_{j \in \mathbf{B}, \rho_j > \rho_i} d_{ij}. \tag{13}$$

With the density peak strategy, the representativeness of $\mathbf{x}_i$ in $\mathbf{B}$ is measured by

$$\gamma_i(\mathbf{B}) = \rho_i \delta_i(\mathbf{B}). \tag{14}$$

With the distance-sum-based strategy, the representativeness of $\mathbf{x}_i$ in the current block is measured by

$$ds_i(\mathbf{B}_l) = \sum_{\mathbf{x}_j \in \mathbf{B}_l} d_{ij}. \tag{15}$$

In a word, the density peak strategy tends to query central instances, while the distance-sum-based strategy tends to query informative ones.

3.3 Clustering quality evaluation

There are both internal and external methods to evaluate the quality of a clustering algorithm. Internal methods do not rely on any class label, while external methods rely on the labels of all instances. During the active learning process, parts of the labels are queried. We now propose a new measure to evaluate the quality of clustering.

Suppose that with the clustering algorithm, a block $\mathbf{B} \subseteq \mathbf{U}$ is partitioned into a set of blocks $\mathcal{B} = \{\mathbf{B}_1, \mathbf{B}_2, \ldots \mathbf{B}_k\}$, i.e., $\cup_{i=1}^{k} \mathbf{B}_i = \mathbf{B}$ and $\forall 1 \le i < j \le k$, $\mathbf{B}_i \cap \mathbf{B}_j = \emptyset$. Let the set of labeled instances in $\mathbf{B}_i$ be $\mathbf{Q}_i$ for any $1 \le i \le k$.

Given $x_l \in \mathbf{B}_i \setminus \mathbf{Q}_i$, the label of x_l is unknown. Let $n(x_l, \mathbf{Q}_i)$ be the nearest neighbor of x_l in $\mathbf{Q}_i$, namely

$$n(x_l, \mathbf{Q}_i) = \operatorname*{argmin}_{x_m \in \mathbf{Q}_i} d_{lm}. \tag{16}$$

Inspired by 1 nearest neighbor (1NN), let $d'(x) = d(n(x, \mathbf{Q}_i))$, now every instance in $\mathbf{B}_i \setminus \mathbf{Q}_i$ has a pseudo label.

The number of actual labels of the jth class in $\mathbf{B}_i$ is

$$l(\mathbf{B}_i, j) = |\{x \in \mathbf{Q}_i | d(x) = l_j\}|, \tag{17}$$

and the number of pseudo labels of the jth class in $\mathbf{B}_i$ is

$$l'(\mathbf{B}_i, j) = |\{x \in \mathbf{B}_i \setminus \mathbf{Q}_i | d'(x) = l_j\}|. \tag{18}$$

Let

$$f_\alpha(\mathbf{B}_i, j) = \frac{l(\mathbf{B}_i, j) + \alpha l'(\mathbf{B}_i, j)}{|\mathbf{Q}_i| + \alpha|\mathbf{B}_i \setminus \mathbf{Q}_i|}, \tag{19}$$

where $\alpha \in [0, 1]$ is a weight, and

$$g_\alpha(\mathbf{B}_i, j) = \begin{cases} 0, & \text{if } f(\mathbf{B}_i, j) = 0; \\ f(\mathbf{B}_i, j) \log f(\mathbf{B}_i, j), & \text{otherwise.} \end{cases} \tag{20}$$

The weighted information entropy of $\mathbf{B}_i$ is given by

$$H_\alpha(\mathbf{B}_i) = -\sum_{j=1}^{c} g(\mathbf{B}_i, j). \tag{21}$$

Finally, the weighted information entropy of $\mathcal{B}$ is given by

$$H_\alpha(\mathcal{B}) = \sum_{i=1}^{k} \frac{|\mathbf{B}_i|}{|\mathbf{B}|} H_\alpha(\mathbf{B}_i). \tag{22}$$

Now we discuss the computation of information entropy through a running example.

Example 1 As illustrated in Fig. 2, $\mathbf{B}$ is partitioned into $\mathcal{B} = \{\mathbf{B}_1, \mathbf{B}_2\}$. There are only positive labels in $\mathbf{B}_1$, so the other 6 instances have positive pseudo labels, and $H_{0.7}(\mathbf{B}_1) = 0$.

There are 12 unlabeled instances in $\mathbf{B}_2$, 4 are close to the positive and 8 are close to the negative instance. So there will be 5 positive and 2 negative labels, including both pseudo and actual ones. We have $H_{0.7}(\mathbf{B}_2) = -\frac{1+8\times0.7}{2+12\times0.7} \log \frac{1+8\times0.7}{2+12\times0.7} - \frac{1+4\times0.7}{2+12\times0.7} \log \frac{1+4\times0.7}{2+12\times0.7} = -\frac{6.6}{10.4} \log \frac{6.6}{10.4} + -\frac{3.8}{10.4} \log \frac{3.8}{10.4} = 0.285$.

Finally, $H_{0.7}(\mathcal{B}) = 0 + \frac{2+12\times0.7}{4+18\times0.7} H(\mathbf{B}_2) = 0 + 0.178 = 0.178$.

3.4 Other issues

Three more issues are handled as follows.

(1) How to control the number of queries? According to the problem statement, the maximal number of queries is $\lfloor n \times qi \rfloor$. Therefore, the `while` loop should terminate as long as the given query runs out. In fact, Line 15 should be extended in the algorithm implementation to enable loop termination.

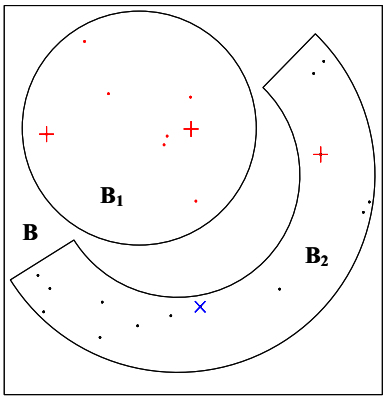


Fig. 2 An example of data and label distribution

(2) How to classify small cluster blocks? The generation of small cluster blocks indicates that our approach is not suitable for them. Hence we left them alone in the while loop, and use kNN to classify them at the end of the algorithm.
(3) How to classify the remaining unprocessed instances? With limited number of queries, after the `while` loop, there are often some unprocessed blocks. We also use kNN to classify instances in them. Note that the neighbors are searched globally instead of within the current block.

4 Experiments

In this section, we report the results of experiments to analyze the effectiveness of the TACS algorithm. Through the experiments, we aim to answer

(1) Is the TACS algorithm more accurate than popular supervised classification algorithms?
(2) Is the TACS algorithm more accurate than popular active learning algorithms?
(3) Is the TACS algorithm more accurate than single clustering technique based algorithms?
(4) Can the TACS algorithm select appropriate base clustering techniques?
(5) Is the TACS algorithm efficient?

The experimental environment is a Windows 10 64-bit operating system, 8 GB memory, Intel (R) Core 2 Quad CPU Q9500@2.83 GHz. The source code with a graphical user interface (GUI) written in Java is available at github.com/fansmale/tacs.

Table 2 Dataset information

Dataset	|U|	|C|	Area	*rt*	*qi*
Seeds	210	8	Life	0.19	0.1
Thyroid	215	6	Life	0.25	0.1
Heart	270	14	Life	0.05	0.1
Spiral	312	3	Synthetic	0.3	0.1
Ionosphere	350	35	Physical	0.04	0.1
R15	600	3	Synthetic	0.2	0.1
USps_2_6	2200	257	Image	0.2	0.1
Waveform	5000	22	Physical	0.1	0.1
Credit	5987	66	Financial	0.35	0.1
Twonorm	7400	21	Synthetic	0.3	0.1
DLA0.1	16,563	18	Society	0.05	0.1
Letter	20,000	17	Computer	0.4	0.1

4.1 Dataset and parameter setting

Table 2 summarizes 12 datasets used for our experiments. Three of the synthetic datasets come from [25], and the other 10 are from the UC Irvine ML repository [34]. The application area ranges from biological to life. The number of attributes for a dataset ranges from 2 to 10,000.

The initial value of *rt* is also presented in the table. It was discussed in Eq. (8). Some datasets are sorted by class label. For example, in the Iris dataset, the first 50 instances have the label "Iris-setosa", the second 50 instances have the label "Iris-versicolor", and the last 50 instances have the label "Iris-virginica". Other datasets such as Spiral have the same characteristics. Unfortunately, the result may be influenced by the order. One reason is that when multiple instances have the highest (lowest) value, we tend to use the first instance. To break this bias, we reordered the dataset before the learning process. This is fulfilled by randomly swapping some instance pairs. Through this operation, the results tend to be different in different runs. Memory overflow occurs while testing TACS and comparing algorithms on large datasets. Hence for DLA, we sampled 1% of the instances to form new datasets. Class distribution is maintained after sampling.

4.2 Comparison with supervised classification algorithms

Table 3 compares TACS with nine popular supervised classification algorithms. These algorithms were implemented in the Weka platform[1]. C4.5 is the implementation of the C4.5 algorithm [31, 35]. NB is the implementation of the Naïve Bayes algorithm [32]. CVR is the implementation of Classification Via Regression algorithm [36]. RF is the implementation of Random Forest algorithm [37]. ABM is the implementation of Discrete AdaBoost algorithm [38]. LB is the implementation of LogitBoost algorithm [39]. Bagging is the implementation of Bagging algorithm [31]. MCC is the implementation of Multiple Class Classifier algorithm [40]. FC is the implementation of Filtered Classifier algorithm [41].

For these algorithms, we randomly select the training set with the size indicated in Table 2. The average ranks were obtained by applying a Friedman test [42], which is the most well-known non-parametric test. The Friedman test analyzes whether there are significant differences among the algorithms. The best results are highlighted in boldface. Here are some detailed observations. (a) TACS generally outperformed existing supervised classification algorithms. In particular, the TACS algorithm had the highest accuracy on seven datasets. Through statistical analysis, the mean rank is 2.16. For the Spiral datasets, the accuracy reached 100%. For the other eleven algorithms, the accuracy did not reach 100% with only 10% labels. (b) There were a few cases in which TACS performed worse than some of the other algorithms. For example, the NB algorithm had a better accuracy than TACS on the Twonorm and DLA datasets. However, the difference in the classification accuracy was less than 1%.

4.3 Comparison with other active learning algorithms

In this section, we compare the TACS algorithm with three state-of-the-art active learning algorithms, including Query-by-committee (QBC) [6], kernel version of QBC algorithm (KQBC) [43], and MAED [33]. QBC is probably the most noticeable algorithm in active learning [6]. It is an integrated approach that requires the establishment of a voting committee [5]. Gilad-Bachrach et al. [43] presented the KQBC algorithm to decrease the runtime. MAED [33] is a uncertainty-based active learning algorithm. It minimizes the expected error to select the most discriminative instances for labeling.

For QBC, KQBC, and MAED, we adopted the source code provided by the authors and the best settings given in reference [6, 33, 43] to ensure the best performance. For KQBC and MAED, the code was run on the Matlab platform, while for QBC, the code was on the JAVA platform. QBC and KQBC use integrated classifiers, including KNN, J48 and NB. The final result is determined by the relevant strategic vote.

Figure 3 describes the learning curves of the TACS algorithm and three active learning algorithms. Each value is the average of 50 runs of the corresponding algorithm. Some detailed observations are described as follows. (a) TACS converges faster than other algorithms. For example, for the

[1] www.weka.com.

Table 3 Accuracy comparison with nine supervised algorithms (in %)

Dataset	C4.5	NB	CVR	RF	ABM	LB	Bagging	MCC	FC	TACS
Seeds	50.26	60.84	82.01	84.65	79.89	82.53	57.67	*85.18*	50.26	**91.53**
Thyroid	76.68	*93.78*	77.72	87.56	86.52	90.15	90.67	84.97	69.43	**95.20**
Heart	73.66	74.48	72.83	*78.60*	76.13	74.48	70.78	74.48	71.60	**79.36**
Spiral	55.51	34.87	40.21	*70.10*	32.38	68.32	49.11	34.87	32.38	**100.00**
Ionosphere	86.34	84.76	80.95	**90.79**	*87.93*	86.03	75.87	85.39	86.03	87.01
R15	73.88	89.44	71.29	*92.59*	12.03	75.00	88.88	63.51	82.96	**98.58**
USps_2_6	88.08	93.48	88.28	*95.90*	93.98	92.92	93.08	95.85	86.46	**96.36**
Waveform	81.64	85.08	85.51	**88.60**	82.75	85.64	86.00	86.80	83.71	*87.83*
Credit	77.72	68.55	**84.66**	*84.07*	78.65	84.00	82.90	82.20	81.03	82.54
Twonorm	80.76	**97.67**	86.24	95.90	85.58	85.57	90.16	*97.52*	79.60	96.67
DLA0.1	91.42	77.78	91.99	**96.90**	54.22	90.42	92.61	83.31	86.28	*94.36*
Letter	71.43	62.83	77.43	**85.07**	6.80	70.79	75.67	69.88	65.99	*80.06*
Meanrank	7.08	5.66	5.75	**2.16**	7.50	4.75	5.50	5.91	7.16	**2.16**

The best and second best results are highlighted in boldface and italic, respectively

Flame dataset, the classification accuracy reached 100% accuracy by labeling only 10% of the instances, while the QBC, MAED and KQBC algorithms were unable to reach 100% accuracy. This indicates that TACS is more appropriate for the cold-start situation. (b) In 9 out of 12 datasets, TACS is more accurate than other algorithms. (c) There are also some cases that TACS performs worse. For example, MAED outperforms TACS in the R15, Heart and USps_2_6 datasets.

4.4 Comparison with clustering-based active learning algorithms

We compare the new approach with the ones based on only one clustering technique. Table 4 shows the accuracy of the TACS algorithm with six clustering-based active learning algorithms. Each algorithm runs ten times to obtain average accuracy and variance. These six methods adopt the same active learning strategy as TACS. The difference is that the six methods adopt a single classical clustering algorithm, while the TACS adopts a clustering selection strategy. Table 5 shows the average number of times each clustering technique was selected. It not only shows how often each clustering technique is used, but also indicates the actual number of iterations of the active learning process.

From Tables 4 and 5, we observe that the TACS algorithm selects the appropriate base clustering techniques. According to the Friedman test, TACS generally outperforms the other algorithms, with an average rank of 1.58, which is the highest on these datasets. For each of the 12 datasets, TACS achieved the best performance on eight of them, and *k*Means achieved good performance on the Twonorm, and Ionosphere datasets. In some cases, different clustering techniques produce the best result for the given block and respective counters are all added by 1.

4.5 Algorithm efficiency

The efficiency of TACS depends mainly on the base clustering techniques. As shown in Table 5, for DLA and Letter datasets, TACS only needs to run less than 100 rounds of clustering techniques. Therefore, the time complexity of TACS can be regarded as approximately the same as the sum of base clustering techniques. Specifically, in our implementation, the time complexity is $O(mn^2)$ when RW is not selected.

Figure 4 depicts the runtime of the DLA dataset with the change of the number of instances n. TACS indicates that these five clustering techniques are employed for selection. Other versions are for specific clustering techniques. For example, TACS-kMeans indicates that only kMeans is available for splitting each block. Note that RW was not selected for this experiment for two reasons. (a) RW is very time consuming since we need to construct a graph from the original data. Hence it is inappropriate for large datasets. (b) RW rarely outperforms other clustering techniques. This issue has been analyzed in more detail in Sect. 4.4.

Here are some detailed observations. (a) When the number of instances is doubled, the runtime will increase to about 4 times. This shows that the time complexity is proportion to $O(n^2)$. In other words, the TACS algorithm is scalable. (b) Different versions share the same trend, showing the effectiveness of the acceleration techniques presented in Sect. 2. (c) The runtime of TACS is smaller than the sum of other versions. This is due to the fact that some overhead is shared.

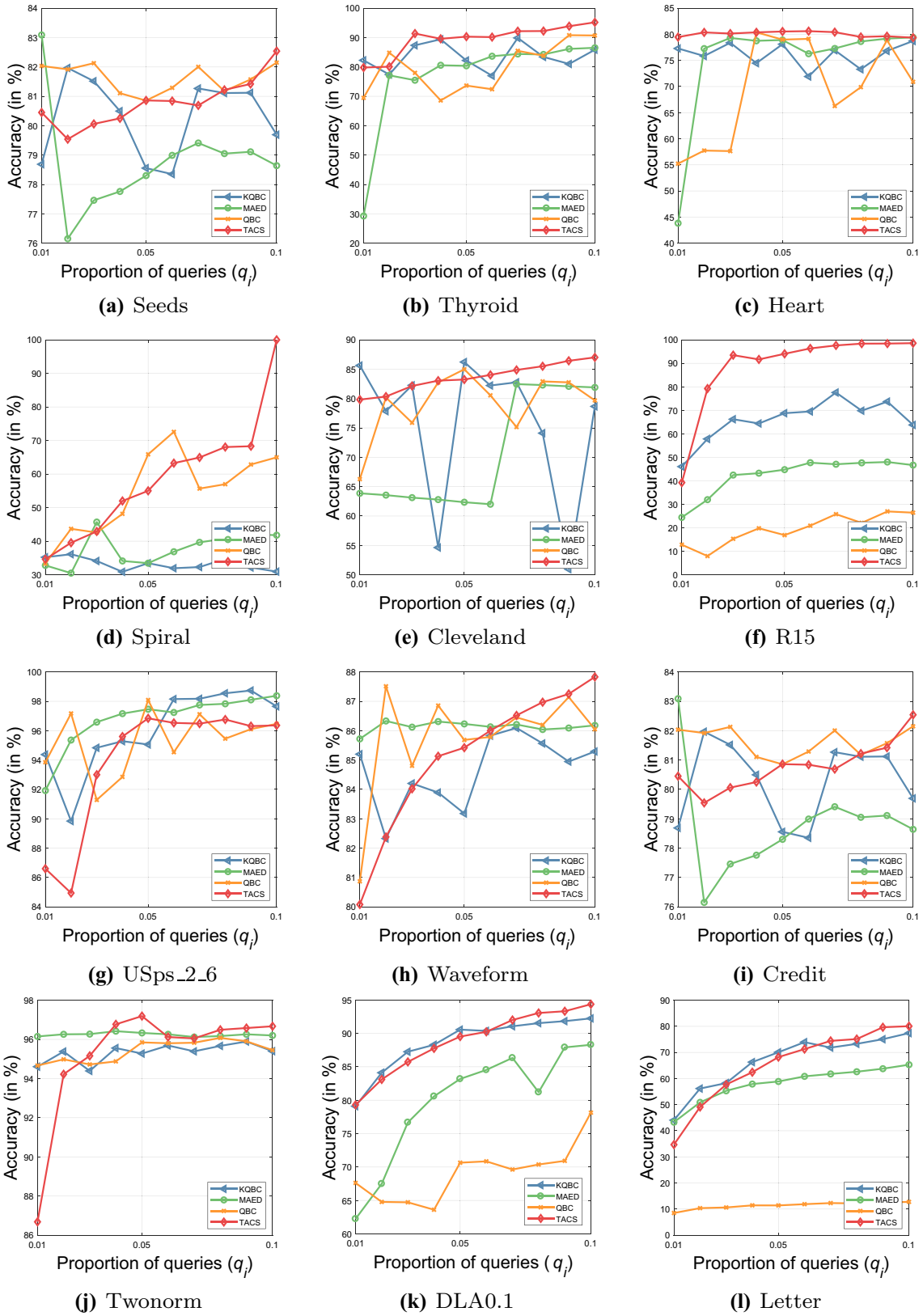


Fig. 3 Comparison of the learning curve of TACS and three popular active learning algorithms

Table 4 Accuracy comparison with algorithms based on single clustering technique (mean ± std in %)

Dataset	DP-Gaussian	kMeans	Hierarchical	DBScan	FCM	RW	TACS
Seeds	*90.47* ±0.00	90.31±0.94	89.36 ±1.55	90.31±0.24	85.18 ±1.91	90.31±0.24	**91.53**±1.56
Thyroid	93.91 ±0.44	*95.00*±0.23	93.35 ±0.90	93.81±0.46	94.22±1.27	93.71±0.50	**95.20**±1.30
Heart	76.83 ±0.86	76.83±0.86	76.83 ±0.86	76.25±0.78	**79.13**±1.07	78.47±1.47	*78.93*±0.73
Spiral	**100.00** ±0.00	79.60±1.87	78.57 ±4.16	69.78±1.85	*81.77*±3.44	58.71±0.00	**100.00**±0.00
Ionosphere	85.39 ±0.00	**87.04**±0.98	86.44 ±2.16	83.68±0.29	85.39±2.00	83.74±0.31	*87.01*±0.55
R15	**98.88** ±0.00	97.49±0.49	97.81 ±0.89	96.46±1.15	97.50±0.78	98.20±0.67	*98.57*±0.69
USps_2_6	*96.34* ±0.40	94.70±0.49	91.05 ±4.23	86.18±1.55	81.33±12.59	96.01±1.88	**97.62**±1.37
Waveform	85.51 ±0.66	*86.96*±0.35	86.20 ±0.44	84.11±0.37	85.79±0.83	84.69±0.34	**87.75**±0.51
Credit	80.39 ±1.50	80.43±0.61	*80.68* ±0.75	80.25±1.40	79.75 ±1.51	80.47±1.22	**82.54**±0.95
Twonorm	94.26 ±1.48	**97.54**±0.22	94.62 ±0.94	88.25±2.20	94.55±0.33	90.22±4.48	*96.45*±1.63
DLA0.1	90.69 ±2.02	92.63±1.22	*94.60* ±0.86	91.30±0.36	87.86±3.40	87.96±0.94	**94.36**±0.67
Letter	69.45 ±2.35	*76.49*±0.78	68.66 ±0.65	65.85±1.76	65.01 ±1.93	62.79±1.57	**80.06**±1.03
Meanrank	3.33	*3.16*	4.00	6.00	4.83	4.66	**1.33**

The best and second best results are highlighted in boldface and italic, respectively

Table 5 The average number of times each clustering technique was selected

Dataset	DP-Gaussian	kMeans	Hierarchical	DBScan	FCM	RW
Seeds	0.1	*0.3*	**0.4**	0.0	*0.3*	0.0
Thyroid	**0.6**	0.2	*0.4*	0.0	0.1	0.0
Heart	0.2	**0.9**	*0.4*	0.0	0.3	0.0
Spiral	**2.0**	0.0	0.0	0.0	0.0	0.0
Ionosphere	**0.9**	*0.5*	0.1	0.0	*0.5*	0.0
R15	*1.6*	**1.7**	0.5	0.8	1.1	0.0
USps_2_6	*0.9*	0.5	0.1	0.0	0.2	**1.2**
Waveform	**31.6**	25.4	*26.1*	0.6	23.3	0.2
Credit	**30.6**	16.2	*25.9*	2.7	17.5	3.6
Twonorm	**11.4**	7.6	*8.9*	0.0	7.0	0.6
DLA0.1	*62.8*	28.6	**81.8**	14.2	20.4	0.0
Letter	**34.4**	*31.2*	26.2	10.2	13.0	0.3

The highest and second highest values are highlighted in boldface and italic, respectively

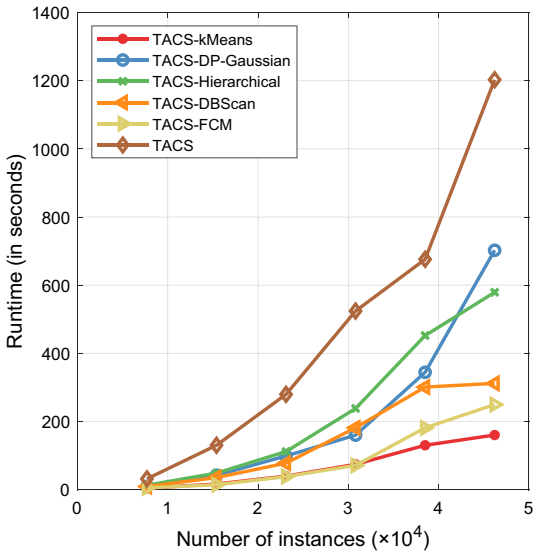


Fig. 4 The runtime of TACS with different base clustering techniques

4.6 Discussions

Now we can answer the questions proposed at the beginning of this section.

(1) TACS is more accurate than popular supervised classification algorithms, including C4.5, NB, RF, etc. This is validated by Table 3. Unfortunately, on some datasets such as Ionosphere, it is significantly worse than some other algorithms such as RF. The reason may be that clustering techniques do not perform well on those datasets.

(2) TACS is more accurate than popular active learning algorithms, including QBC, MAED, and KQBC. This is validated by Fig. 3. It was also defeated by MAED on the Heart dataset. The reason may be that for some

datasets, informative instances are more important than representative ones.

(3) TACS is more accurate than single clustering technique based algorithms. This is validated by Table 4. It is the best, or the second best one on all datasets.

(4) In most cases, TACS can find out the appropriate base clustering techniques. This is validated by Table 5.

(5) TACS is efficient as long as we do not employ inefficient clustering techniques. This is validated by Table 5 and Fig. 4.

5 Related work

This section discusses some related work, including active learning, clustering-based active learning and three-way decision. Their relationships with this work will be emphasized.

5.1 Active learning

Active learning [4, 44] is a special form of semi-supervised learning [45, 46], which in turn is a special form of classification. Classification tasks typically require a large amount of labeled data to form a training set for building a classifier. However, labeled data are difficult, expensive, or time consuming to obtain [46]. Semi-supervised learning uses abundant unlabeled data with a few labeled ones to build a better classifier. When man–machine interaction is available, active learning obtains key tags to further improve classification accuracy.

The key issue for active learning is: which instances are critical and therefore should be labeled? General answers include informative ones [6, 47], diverse ones [7, 48], representative ones [10, 49], and their combinations [11, 12, 48]. Uncertain sampling [50], Query-by-committee [6], and query by margin [51] are classical approaches to find informative instances. Fuzzy-rough approaches [7, 48] have been developed to find diverse ones. Clustering-based active learning [10, 49], as the basis of this work, tends to find representative ones.

There are a number of typical active learning scenarios. The classical scenario [10] specifies the number of labels provided by oracle. It is explicitly defined in Problem 1. The cost-sensitive active learning scenario [13] involves query cost and misclassification costs. The learning task is to achieve a trade-off between them to minimize the total cost. The active learning from data streams scenario [52] involves dynamic nature of the data. The learner should query instances based on the data observed so far to maximize the classification accuracy on future instances. Moreover, instances usually should be queried immediately to save the storage.

5.2 Three-way decision

Three-way decision [15] is a general methodology for problem solving especially in data mining and machine learning. It was rooted in decision-theoretic rough set model [53], while be more independent since 2010 [15], and quite matured in 2018 with the TAO model [16]. Liu et al. [54] enriched the theory to deal with incomplete information systems. Hu [55] established three-way decision space based on the axiomatic definitions for decision measurement, decision condition and evaluation function. Fang et al. [56] discussed cost-sensitive approximate attribute reduction with three-way decisions. Zhang et al. [57] introduced a general model of decision theoretic three-way approximations of fuzzy sets based on the particle swarm optimization algorithm.

It has been connected with other theories to form new ones. Li et al. [58] investigated three-way cognitive concept learning from the perspective of multigranularity information fusion. Shivhare et al. [59] explored the process of three-way cognitive concept learning to achieve a more precise recall. Qi et al. [60] proposed three-way formal concept analysis to reveal more detailed information from formal contexts. Zhi et al. [61] extended dual formal concept with three-way decision.

It has also inspired the presentation or solution of machine learning tasks. Zhang et al. [62] developed a three-way regression approach for recommender systems. Yu et al. [63] proposed an active three-way clustering method via low-rank matrices for multi-view data. Li et al. [17] designed deep neural network based sequential granular feature extraction method for image processing. Jia et al. [64] applied the three-way decision-theoretic rough set model to address multi-class cost-sensitive learning problems. Min et al. [65] introduced tri-patterns and provided an efficient mining algorithm for different types of sequential data.

Three-decision was first applied to active learning in [66]. Then three-way active learning served as a special form of clustering-based active learning. Wang et al. [10] introduced the first three-way active learning algorithm. It can be viewed as a special case of the TACS algorithm where the DB-Gaussian clustering technique is available. Wu et al. [13] studied the cost-sensitive active learning scenario under the uniform label distribution assumption. Wang et al. [14] explored different assumptions further and established a statistical model. As mentioned in the introduction part, there are three possible actions for each block. Due to the iterative process, the paradigm can be better represented by sequential three-way decision [17, 18].

6 Conclusions and further work

This study has proposed the TACS algorithm to dynamically select the appropriate clustering in the active learning process. A number of techniques were discussed, including cluster selection, query balancing, and tree pruning. Experimental results verify the effectiveness of the algorithm.

The following research topics deserve further investigation:

(1) More clustering techniques in the algorithm framework. Currently, only a few clustering techniques have been incorporated into TACS. Other techniques can be incorporated to accommodate data with different shapes. In addition, these techniques should be fine-tuned or modified to fit the framework of the algorithm.
(2) Better evaluation measures to select clustering techniques. TACS uses weighted entropy to evaluate the quality of clustering. New measures based on the Gini indicator might be good alternatives. More sophisticated measures can be designed to consider other information such as block size ratios and data distribution.
(3) Clustering ensemble techniques for active learning. TACS only selects the currently "best" clustering technique. By designing cluster ensemble techniques, new blocks can be obtained from these different techniques. Hence it is possible to obtain more *stable* blocks of better quality. Moreover, different classification strategies can be employed for instances inside or outside stable blocks.

In summary, TACS is a general algorithm framework that can be enriched in the future.

Acknowledgements This work is in part supported by the Natural Science Foundation of Sichuan Province under Grant number 2019YJ0314, and the Sichuan Province Youth Science and Technology Innovation Team under Grant number 2019JDTD0017.

References

1. Tuia D, Ratle F, Pacifici F, Kanevski MF, Emery WJ (2009) Active learning methods for remote sensing image classification. IEEE Trans Geosci Remote Sens 47(7):2218–2232
2. Thompson CA, Califf ME, Mooney RJ (1999) Active learning for natural language parsing and information extraction. In: ICML, pp 406–414
3. Tong S, Koller D (2002) Support vector machine active learning with applications to text classification. J Mach Learn Res 2(1):45–66
4. Angluin D (1988) Queries and concept learning. Mach Learn 2(4):319–342
5. Settles B (2010) Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison
6. Seung HS, Opper M, Sompolinsky H (1992) Query by committee. In: Proceeding of the fifth workshop on computational learning theory, vol 284, pp 287–294
7. Wang R, Chen DG, Kwong S (2014) Fuzzy-rough-set-based active learning. IEEE Trans Fuzzy Syst 22(6):1699–1704
8. Wang R, Chow CY, Kwong S (2016) Ambiguity-based multiclass active learning. IEEE Trans Fuzzy Syst 24(1):242–248
9. Nguyen HT, Smeulders A (2004) Active learning using pre-clustering. In: ICML, pp 79–90
10. Wang M, Min F, Zhang ZH, Wu YX (2017) Active learning through density clustering. Expert Syst Appl 85:305–317
11. Du B, Wang ZM, Zhang LF, Zhang LP, Liu W, Shen JL, Tao DC (2017) Exploring representativeness and informativeness for active learning. IEEE Trans Cybern 47(1):14–26
12. Huang SJ, Jin R, Zhou ZH (2014) Active learning by querying informative and representative examples. IEEE Trans Pattern Anal Mach Intell 36(10):1936–1949
13. Wu YX, Min XY, Min F, Wang M (2019) Cost-sensitive active learning with a label uniform distribution model. Int J Approx Reason 105:49–65
14. Wang M, Lin Y, Min F, Liu D (2019) Cost-sensitive active learning through statistical methods. Inf Sci 501:460–482
15. Yao YY (2012) An outline of a theory of three-way decisions. In: RSCTC. Springer, Berlin, pp 1–17
16. Yao YY (2018) Three-way decision and granular computing. Int J Approx Reason 103:107–123
17. Li HX, Zhang LB, Zhou XZ, Huang B (2017) Cost-sensitive sequential three-way decision modeling using a deep neural network. Int J Approx Reason 85:68–78
18. Yang X, Li TR, Fujita H, Liu D (2019) A sequential three-way approach to multi-class decision. Int J Approx Reason 104:108–125
19. Qian J, Liu CH, Yue XD (2019) Multigranulation sequential three-way decisions based on multiple thresholds. Int J Approx Reason 105:396–416
20. Yao JT, Vasilakos AV, Pedrycz W (2013) Granular computing: perspectives and challenges. IEEE Trans Syst Man Cybern C Appl Rev 43(6):1977–1989
21. Yao YY (1999) Granular computing using neighborhood systems. In: Advances in soft computing. Springer, London, pp 539–553
22. Dai JH, Hu QH, Hu H, Huang DB (2018) Neighbor inconsistent pair selection for attribute reduction by rough set approach. IEEE Trans Fuzzy Syst 26(2):937–950
23. Sun L, Zhang XY, Qian YH, Xu JC, Zhang SG (2019) Feature selection using neighborhood entropy-based uncertainty measures for gene expression data classification. Inf Sci 502:18–41
24. Zhao H, Wang P, Hu QH, Zhu PF (2019) Fuzzy rough set based feature selection for large-scale hierarchical classification. IEEE Trans Fuzzy Syst 27:1891–1903
25. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
26. Hartigan JA, Wong MA (1979) Algorithm AS 136: A k-Means clustering algorithm. Appl Stat 28(01):100–108
27. Bezdek JC, Ehrlich R, Full W (1984) FCM: the fuzzy *c*-means clustering algorithm. Comput Geosci 10(2):191–203
28. Johnson SC (1967) Hierarchical clustering schemes. Psychometrika 32(3):241–254
29. Ester M, Kriegel HP, Sander J, Xu XW (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. Morgan Kaufmann Publishers Inc., San Francisco, pp 226–231
30. Harel D, Koren Y (2001) On clustering using random walks. In: FSTTCS. Springer, Berlin, pp 18–41
31. Quinlan R (1996) Bagging, Boosting, and C4.5. In: AAAI/IAAI, pp 725–730
32. Irina R (2001) An empirical study of the Naïve Bayes classifier. In: IJCAI workshop on empirical methods in artificial intelligence, pp 41–46

33. Cai D, He XF (2012) Manifold adaptive experimental design for text categorization. IEEE Trans Knowl Data Eng 24(4):707–719
34. Blake C, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html
35. Xiang ZY, Zhang L (2012) Research on an optimized C4.5 algorithm based on rough set theory. In: International conference on management of e-Commerce and e-Government, pp 272–274
36. Ruan YX, Lin HT, Tsai MF (2014) Improving ranking performance with cost-sensitive ordinal classification via regression. Inf Retr 17(02):133
37. Liaw A, Wiener M (2002) Classification and regression by randomForest. R News 2–3:18–22
38. Cortés EA, Martínez MG, Rubio NG (2007) Multiclass corporate failure prediction by Adaboost.M1. Int Adv Econ Res 13(02):301–312
39. Cai YD, Feng KY, Lu WC, Chou KC (2006) Using LogitBoost classifier to predict protein structural classes. J Theor Biol 238(2):172–176
40. Afshar S, Mosleh M, Kheyrandish M (2013) Presenting a new multiclass classifier based on learning automata. Neurocomputing 104:97–104
41. Zhang SL, Zhang TS, Liu M, Li KL, Yuan BZ (2010) An experimental study of classifier filtering. In: ICWMMN, pp 361–364
42. Reyes O, Altalhi AH, Ventura S (2018) Statistical comparisons of active learning strategies over multiple datasets. Knowl-Based Syst 145:274–288
43. Gilad-Bachrach R, Navot A, Tishby N (2004) Kernel query by committee (KQBC). Leibniz Center Technical Report 88, Hebrew University
44. Cohn DA, Ghahramani ZB, Jordan MI (1996) Active learning with statistical models. J Artif Intell Res 4(1):129–145
45. Blum A, Chawla S (2001) Learning from labeled and unlabeled data using graph mincuts. In: ICML, pp 1–8
46. Belkin M, Niyogi P (2004) Semi-supervised learning on Riemannian manifolds. Mach Learn 56(1–3):209–239
47. Lewis DD, Gale WA (1994) A sequential algorithm for training text classifiers. In: SIGIR, pp 3–12
48. Wang R, Wang XZ, Kwong S, Chen X (2017) Incorporating diversity and informativeness in multiple-instance active learning. IEEE Trans Fuzzy Syst 25(6):1460–1475
49. Dasgupta S, Hsu D (2008) Hierarchical sampling for active learning. In: ICML, pp 208–215
50. Lewis DD, Catlett J (1994) Heterogeneous uncertainty sampling for supervised learning. In: ICML, pp 148–156
51. Campbell C, Cristianini N, Smola A (2000) Query learning with large margin classifiers. In: ICML, pp 111–118
52. Zhu X, Zhang P, Lin X, Shi Y (2007) Active learning from data streams. In: ICDM, pp 757–762
53. Yao YY, Wong S (1992) A decision theoretic framework for approximating concepts. Int J Man Mach Stud 37:793–809
54. Liu D, Liang DC, Wang CC (2016) A novel three-way decision model based on incomplete information system. Knowl-Based Syst 91:32–45
55. Hu BQ (2014) Three-way decisions space and three-way decisions. Inf Sci 281:21–52
56. Fang Y, Min F (2019) Cost-sensitive approximate attribute reduction with three-way decisions. Int J Approx Reason 104:148–165
57. Zhang QH, Xia DY, Liu KX, Wang GY (2020) A general model of decision-theoretic three-way approximations of fuzzy sets based on a heuristic algorithm. Inf Sci 507:522–539
58. Li JH, Huang CC, Qi JJ, Qian YH, Liu WQ (2017) Three-way cognitive concept learning via multi-granularity. Inf Sci 378(1):244–263
59. Shivhare R, Cherukuri AK (2017) Three-way conceptual approach for cognitive memory functionalities. Int J Mach Learn Cybern 8:21–34
60. Qi JJ, Qian T, Wei L (2016) The connections between three-way and classical concept lattices. Knowl-Based Syst 91:143–151
61. Zhi HL, Qi JJ, Qian T, Wei L (2019) Three-way dual concept analysis. Int J Approximate Reasoning 114:151–165
62. Zhang HR, Min F, Shi B (2017) Regression-based three-way recommendation. Inf Sci 378:444–461
63. Yu H, Wang XC, Wang GY, Zeng XH (2020) An active three-way clustering method via low-rank matrices for multi-view data. Inf Sci 507:823–839
64. Jia XY, Li WW, Shang L (2019) A multiphase cost-sensitive learning method based on the multiclass three-way decision-theoretic rough set model. Inf Sci 485:248–262
65. Min F, Zhang ZH, Zhai WJ, Shen RP (2020) Frequent pattern discovery with tri-partition alphabets. Inf Sci 507:715–732
66. Min F, Liu FL, Wen LY, Zhang ZH (2019) Tri-partition cost-sensitive active learning through kNN. Soft Comput 23:1557–1572

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.