Geometric Active Learning via Enclosing Ball Boundary

Xiaofeng Cao, Ivor W. Tsang, Jianliang Xu, Zenglin Shi, Guandong Xu

Abstract—Active Learning (AL) requires learners to retrain the classifier with the minimum human supervisions or labeling in the unlabeled data pool when the current training set is not enough. However, general AL sampling strategies with a few label support inevitably suffer from performance decrease. To identify which samples determine the performance of the classification hyperplane, Core Vector Machine (CVM) and Ball Vector Machine (BVM) use the geometry boundary points of each Minimum Enclosing Ball (MEB) to train the classification hypothesis. Their theoretical analysis and experimental results show that the improved classifiers not only converge faster but also obtain higher accuracies compared with Support Vector Machine (SVM). Inspired by this, we formulate the cluster boundary point detection issue as the MEB boundary problem after presenting a convincing proof of this observation. Because the enclosing ball boundary may have a high fitting ratio when it can not enclose the class tightly, we split the global ball problem into two kinds of small Local Minimum Enclosing Ball (LMEB): Boundary ball (B-ball) and Core ball (C-ball) to tackle its over-fitting problem. Through calculating the update of radius and center when extending the local ball space, we adopt the minimum update ball to obtain the geometric update optimization scheme of B-ball and C-ball. After proving their update relationship, we design the Local Enclosing Ball (LEB) algorithm using centers of B-ball of each class to detect the enclosing ball boundary points for AL sampling. Experimental and theoretical studies have shown that the classification accuracy, time, and space performance of our proposed method significantly are superior than the state-of-the-art algorithms.

Index Terms—Active learning, cluster boundary, minimum enclosing ball, over-fitting.

1 INTRODUCTION

Active learning [1] is a well-studied subject area in many machine learning and data mining scenarios such as text AL [17], image AL [18, 47, 58], transfer AL [19, 20], online learning [2], semi-supervised learning [3], and so on, where the unannotated resources are abundant and cheap, but collecting massive annotated data is expensive, time-consuming, and impractical. In this learning process, reducing the prediction error rate of the version space (data set) and being able to achieve this through fewer queries and little training is the goal of an active learner. To improve the performance of the classifier, the learner is allowed to sample a subset from an unlabeled data pool to select those instances that provide the main support for constructing the classification model in AL. Usually, training the optimal classification hypothesis by accessing the unlabeled data pool and querying their true labels for a certain number is essential, but this may encounter a selection difficulty because there is a large amount of unlabeled data in the pool.

To tackle this issue, uncertainty sampling [4] was proposed to guide AL by selecting the most important instances in a given sampling scheme or distribution assumption, such as Margin [5], uncertainty probability [6], maximum entropy [7], confused votes by committee [8], maximum model diameter [9], maximum unreliable [44], and so on. Therefore, the main issue for AL is to find a way to reduce the number of queries or converge the classifier quickly to reduce the total cost of the learning process. Accompanied by multiple iterations, querying stops when the defined sampling number is met or a satisfactory model is found. However, it is still necessary to traverse the huge version space repeatedly in this framework, although this technique performs well.

Querying the labels of sampled data is a reasonable approach to improving the AL prediction model when the training set is insufficient, but devising such positive evaluation rules is awkward because neither the learners nor the human annotators know which instances are the most important in the pool. In general, we seek methods with advantages of (1) high efficiency in querying the most effective or important instances; and (2) low redundancy in reducing the queries on redundant or useless instances. Intuitively, training a robust prediction model that performs well on unannotated data is the common goal of the different AL approaches and there have also been many uncertain evaluation strategies proposed to achieve this goal. However, they always suffer from one main limitation, that is, heuristically searching the complete version space to obtain the optimal sampling subset is impossible because of the unpredictable scale of the candidate set.

In practice, it might be more optimal when the optimal classification model can still be trained by a subspace without any prior experience, and this will solve the previously mentioned limitation in a different way [14, 45, 46]. For reliable space scaling, hierarchical sampling
utilizes unsupervised learning as a way of obtaining the cluster bone points to improve the sampling (see Figure 1(b)). Although this provides positive support with more informative instances, the data points within clusters always have weak or no influence on the current model because of their clear class labels. We call these data points core points. On the other hand, hierarchical sampling does sample some redundant points to annotate its subtree by its root node’s label. Interestingly, after removing the core points, a similar trained model can still be obtained, although only the cluster boundary points are retained (see Figure 1(c)).

In this paper, the cluster boundary points detection problem is considered equivalent to a geometric description problem, where boundary points are located at the geometric surface of one high dimensional enclosing space. By utilizing the geometric features of manifold space, [23] reconstructed the geometric space by local representative sampling for AL, and [24] mapped the underlying geometry of the data by its manifold adaptive kernel space for AL. Therefore, we consider the cluster boundary points detection problem as the enclosing ball boundary fitting, which is popular in a hard-margin support vector data description (SVDD) [16]. In this one-class classification problem, fitting the hyperplane of the high dimensional ball is used to improve the generalization of the training model when the trained data labels are imbalanced. To reduce the time consumption of multiple quadratic programming (QP) in large scale data, [51] [52] changed the SVM to a problem of minimum enclosing ball (MEB) and then iteratively calculated the ball center and radius in a (1+ε) approximation. Trained by the detected core sets, the proposed Core vector Machine (CVM) performed faster than the SVM and needed less support vectors. Especially in the Gaussian kernel, a fixed radius was used to simplify the MEB problem to the EB (Enclosing Ball), and accelerated the calculation process of the Ball Vector Machine (BVM) [54]. Without sophisticated heuristic searches in the kernel space, the training model, using points of high dimensional ball surface, can still be approximated to the optimal solution. However, the MEB alone could not calculate the fitting hyperplane of the
ball and nor could obtain the real boundary points of the ball. This is because the kernel data space might not be a complete ball space or the ball surface might not be tight to class within the ball (Figure 2(a)).

To obtain a tighter enclosing ball boundary, we split the MEB, which is a global optimization problem, into two types of local minimum enclosing ball (LMEB) issues, where one type is the B-ball (boundary ball), the other is C-ball (core ball), and centers of the B-balls are the enclosing ball boundary points. This approach tries to optimize the goodness of fit to obtain the whole geometric boundary points for each cluster. Figure 2(b) shows the motivation for this approach. The above observations and investigations motivated us to propose a new AL strategy - Local Enclosing Ball (LEB), which utilizes the MEB approach to obtain the enclosing ball boundary points for AL sampling. Our contributions in this paper are:

- We propose an idea of reducing the uncertainty sampling space to an enclosing ball boundary hyperplane and validate it in various settings of classification.
- We develop an AL approach termed LEB that samples independently without iteration and help from labeled data.
- We break the theoretical curse of uncertainty sampling by enclosing ball boundary in AL since LEB is neither a model-based nor label-based strategy with the fixed time and space complexities of $O(N \log N)$ and $O(N)$ respectively.
- We conduct experiments to verify that LEB can be applied in multi-class settings to overcome the binary classification limitation of many existing AL approaches.

The preliminaries are described in Section 2 and the performance of cluster boundary is defined in Section 3.1 (Theorem 1). To prove it, we discuss the model distance (Lemma 1 of Section 3.2) and inclusion relation of classifiers (Lemma 2 of Section 3.3) between cluster boundary and core points in binary classification, multi-class of low and high dimensional settings, respectively. The background for the MEB problem is presented in Section 4.1. Then, we optimize the geometric update of radius (Section 4.2) and center (Section 4.3) when extending the local ball space, in which the established update optimization equation is analyzed in Lemmas 3-5 of Section 4.4. Based on the above findings, we design the LEB algorithm in Section 4.5, analyze its time and space complexities in Section 4.6, and discuss its advantages in Section 4.6. The experiments and results, including eight geometric clustering data sets and one unstructured letter recognition data set, are reported in Sections 5.1-5.3. Then Section 5.4 further discusses the time performance of different AL approaches. Finally, we conclude this paper in Section 6.

2 Preliminary

In this section, we firstly describe the general AL problem, and then classify the unlabeled data into two kinds of objects involved with evaluating whether a sampled data point will benefit the classifier training. As we define the AL sampling issue as geometric cluster boundary detection problem, we introduce some related geometric structures for geometric AL, where the related definitions, main notations and variables are briefly summarized in Table 1.

Given $\mathcal{X}$ represents data space $\{x_1, x_2, x_3, ..., x_n\} \in \mathbb{R}^{n \times m}$, where $x_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{im})$ and the label space $\mathcal{Y} = \{y_1, y_2, y_3, ..., y_n\}$, considering the classifier:

$$h_w := w^T x + b \quad (1)$$

where $w$ is the parameter vector and $b$ is the constant vector, here gives:

**Definition 1. Active Learning.** Optimizing $w$ to get the minimum RSS (residual sum of squares) $\sum_{i=1}^{n} (w^T x_i - y_i)^2$ (2)

i.e.,

$$w^* = (X^T X)^{-1} X^T Y \quad (3)$$

$$s.t. \ X_i = [X_i x_q]$$

where $X_i$ is the labeled data, $x_q$ is the queried data, and $X_i$ is the updated training set.

Given hypothesis $h_w^+$, the error rate change of predicting after adding the queried data $x_i$ is

$$\Delta(x) = \text{err}(h_w) - \text{err}(h_w^+) \quad (4)$$

$s.t. \ X_i = X_i^+ \ for \ h_w^+ \ X_i = [X_i x_i] \ for \ h_w^+$

where $\text{err}(\cdot)$ represents the prediction error rate of $\mathcal{X}$ when training the input classification model.

**Definition 2. Effective point.** If $\Delta(x_i) > \alpha$, $x_i$ is an effective point that shows positive help for the next training after adding it to $X_q$. Here $\alpha \in (0, 1)$, and it is an impactor factor that decides whether the data point will affect $h_w$.

**Definition 3. Redundant point.** If $\Delta(x_i) \leq \alpha$, $x_i$ is a redundant point that has weak or negative influence on the current $h_w$ and future model $h_w^+$.

In an enclosing geometric space, we scale the AL sampling issue as cluster boundary point detection problem. Here we introduce some related geometric structures in AL.

**Definition 4. Cluster boundary point** [11]:

A boundary point $p$ is an object that satisfies the following conditions:

1. It is within a dense region $\mathbb{R}$.
2. $\exists$ region $\mathbb{R}^r$ near $p$, Density($\mathbb{R}^r$) $\gg$ Density($\mathbb{R}$) or Density($\mathbb{R}^r$) $\ll$ Density($\mathbb{R}$).

**Definition 5. Core point.** A core point $p$ is an object that satisfies the following conditions:

1. It is within a dense region $\mathbb{R}$.
2. $\exists$ an expanded region $\mathbb{R}^r$ based on $p$, Density($\mathbb{R}^r$) $-$ Density($\mathbb{R}$) $\to 0$.

**Definition 6. Enclosing ball boundary.** An enclosed high dimensional hyperplane connects all the boundary points. Let $\beta$ define the boundary points of one class, and $\beta = (\beta_1, \beta_2, ..., \beta_{N_{\beta}})$, where $N_{\beta}$ is the number of boundary points, then the closed hyperplane $H$ satisfies the following conditions:

1. Most of the boundary points are distributed in the hyperplane.
3 Motivation

In clustering-based AL work, core points are redundant points because of their clear class labels, and provide a little help for parameter training of classifiers. Considering cluster boundary points may provide decisive factors for the support vectors, CVM and BVM use the points distributed on the hyperplane of an enclosing ball to train fast core support vectors in large-scale data sets. Their significant success motivates the work of this paper.

To further show the importance of cluster boundary points, we (1) clarify the performance of training cluster boundary points in Section 3.1, (2) discuss the model distance to the classification line or hyperplane of boundary and core points in Section 3.2, and (3) analyze the inclusion relation of classifiers when training boundary and core points in Section 3.3, where the discussion cases of (2) and (3) are binary, and multi-class classifications of low and high dimensional space.

### 3.1 Performance of cluster boundary

In this paper, we consider the performance of classification model is determined by cluster boundary points. Therefore, we have

**Theorem 1.** The performance of classification model by training cluster boundary points are similar with that of boundary points, that is to say,

\[
\Delta' = \text{err}(h^{\Xi}) - \text{err}(h^{\beta}) \to 0
\]

where \(\zeta\) represents the core points, \(\beta\) represents the cluster boundary points, and \(\Xi\) represents the cluster boundary points.

Theorem 1 aims to show which core points are redundant and have little influences on training \(h\). The objective function is supported by **Lemma 1** and 2 in the next subsections. One of them is that cluster boundary points are closer to classification model compared with other data and the other is that the trained models based on core points are a subset of the boundary points. Then, the detailed proofs of the two Lemmas are discussed in settings of binary, multi-class settings of low and high dimension space, respectively.

### 3.2 Model distance

Model distance function \(L(:, \cdot)\) is defined as the distance to the classification line or hyperplane of one data point. The model distance relations of boundary points and core points are described in the following **Lemma 1**.

**Lemma 1.** The model distance of boundary points are bigger than that of core points, that is to say,

\[
L(\beta, h) < L(\zeta, h)
\]

Lemma 1 is divided into three different cases:

- **Corollary 1**: binary classification in low dimensional space, where Corollaries 1.1 and 1.2 prove Theorem 1 in the adjacent classes and separation classes, respectively.
- **Corollary 2**: multi-class classification problem in low dimensional space.
- **Corollary 3**: high dimensional space.

**Corollary 1**: Binary classification in low dimensional space

Given two facts in the classification: (1) the data points far from \(h\) usually have clear assigned labels with a high prediction class probability; (2) \(h\) is always surrounded by noises and a part of the boundary points. Based on these facts, the proof is as follows.

**Corollary 1.1**: Adjacent classes

**Proof.** For the binary classification of the adjacent classes problem (see Figure 3(a)) with \(Y' \in \{-1, 1\}\), we get the result:

\[
\begin{aligned}
SSR(\zeta^+) &= \sum_{i=1}^{N_+} (w^T x - 1)^2 \to 0, X_i = \zeta^+ \\
SSR(\zeta^-) &= \sum_{i=1}^{N_-} (w^T x + 1)^2 \to 0, X_i = \zeta^- \\
SSR(\beta^+) &= \sum_{i=1}^{N^+} (w^T x^2 - 0)^2 \to 0, X_i = \beta^+ \\
SSR(\eta) &= \sum_{i=1}^{N_\eta} (w^T x - 0)^2 \to 0, X_i = \eta
\end{aligned}
\]

where \(\zeta^+\) represents the core points located inside the positive class, \(\zeta^-\) represents the core points located inside the negative class, \(\beta^+\) represents the cluster boundary
points near \( h \), and \( \eta \) represents the noises near \( h \). Here \( \mathcal{N}_c, \mathcal{N}_{\zeta}, \mathcal{N}_{\beta^+}, \) and \( \mathcal{N}_{\eta} \) represent their numbers of the four types of points.

Because noises always have wrong guidance on model training, we only focus on the differences between the core and boundary points, that is to say,

\[
|h_{\beta^+}(\beta^+) - |h_{\beta^-}(\zeta)| = (wx_{\beta^+})^2 - (wx_{\beta^-})^2 = w^2(x_{\beta^+} - x_{\beta^-}) \rightarrow \epsilon_1 < 0
\]

The distance function between \( x_i \) and \( h \) in \( \mathbb{R} \) space is:

\[
\mathcal{L}(x_i, h) = \frac{|w_1x_{i1} + w_2x_{i2} + b|}{\sqrt{w_1^2 + w_2^2}}
\]

Because the classifier definition is \( h_{\beta}(x_i) = w_1x_{i1} + w_2x_{i2} + b \), \( \mathcal{L}(\beta, h) < \mathcal{L}(\zeta, h) \), then Lemma 1 is established when \( \beta = \beta^+ \) (see Figure 3(b)).

**Corollary 1.2: Separation classes**

**Proof.** In the separation classes problem (see Figure 3(c)), the trained model based on any data points will lead to a strong classification result, that is to say, all AL approaches will perform well in this setting since:

\[
\begin{align*}
|w_{\beta^+}x_{\beta^+} - w_{\beta^-}x_{\beta^-}| &= |w(x_{\beta^+} - x_{\beta^-})| \\
&= \epsilon_2 > 0 \\
|w_{\beta^-}x_{\beta^-} - w_{\beta^+}x_{\beta^+}| &= |w(x_{\beta^-} - x_{\beta^+})| \\
&= \epsilon_3 > 0
\end{align*}
\]

where \( \beta^+ \) represents the boundary points near \( h \) in the positive class, \( \beta^- \) represents the boundary points near \( h \) in the negative class, \( x_{\beta^+} \in \beta^+ \), and \( x_{\beta^-} \in \beta^- \). Let \( \beta^+ = \beta^+ \cup \beta^- \), \( \zeta = \zeta^+ \cup \zeta^- \), we can still have the results of Eq. (8) and (9).

**Corollary 2: Multi-class classification in low dimensional space**

**Proof.** In this setting, \( \mathcal{Y} \in \{0, 1, 2, ..., k\} \), the classifier set \( H = \{h_{\beta^1}, h_{\beta^2}, ..., h_{\beta^k}\} \), and cluster boundary points are segmented into \( k \) parts \( \{\beta^1, \beta^2, ..., \beta^k\} \), where \( \beta^i \) represents the data points close to \( h^i \), \( i \in (1, k) \) (see Figure 3(d)). Based on the result of Case 1, dividing the multi-class classification problem into \( k \) binary classification problems, we can obtain:

\[
|h_{\beta^i}(\beta^i)| < |h_{\beta}((\zeta^i)|, \forall i
\]

and

\[
\mathcal{L}(\beta, h) < \mathcal{L}(\zeta, h), \forall i
\]

where \( \zeta^i \) represents the core points near \( h^i \). Then, the following holds:

\[
\mathcal{L}(\beta, h) < \mathcal{L}(\zeta, h)
\]

**Corollary 3: High dimensional space**

**Proof.** In high dimensional space, the distance function between \( x_i \) and hyperplane \( h_w \) is

\[
\mathcal{L}(x_i, h_w) = |wx_i + C|(|w|)^{-1/2}
\]

**3.3 Inclusion relation of classifiers**

Inclusion relation of classifiers is the collection relation of training different data sub sets. Lemma 2 shows this relation of training boundary and core points, respectively.

**Lemma 2.** Training models based on \( \beta \) is the subset of models based on \( \zeta \), that is to say,

\[
h_{\beta} \subseteq h_{\zeta}
\]

It shows training models based on \( \beta \) can predict \( \zeta \) well, but the model based on \( \zeta \) may sometimes not predict \( \beta \) well. To prove this relation, we discuss it in three different cases:

- Corollary 4: binary classification in low dimensional space, where Corollary 4.1 and Corollary 4.2 prove Lemma 2 in one-dimension space and two-dimension space, respectively.
- Corollary 5: binary classification in high dimensional space.
- Corollary 6: multi-class classification.

**Corollary 4: Binary classification in low dimensional space**

**Corollary 4.1: Linear one-dimension space**

**Proof.** Given point classifier \( h_{\zeta}^i, h_{\beta}^i \) in the linear one-dimension space as described in Figure 4(a),

\[
h_{\zeta}^i = C, \gamma \in (\zeta_1, \zeta_2) \quad \text{or} \quad h_{\beta}^i = C, \gamma \in (\beta_1, \beta_2)
\]
where \( z_1, z_2 \) are core points. In comparison, the boundary points of \( \beta_1, \beta_2 \) have smaller distances to the optimal classification model \( h_w^\beta \), i.e., \( z_1 < \beta_1, z_2 < \beta_2 \). Therefore, it is easy to conclude: \( (\beta_1, \beta_2) \subseteq (z_1, z_2) \). Then, classifying \( z_1 \) and \( z_2 \) by \( h_w^\beta \) is successful, but we cannot classify \( \beta_1 \) and \( \beta_2 \) by \( h_w^\gamma = \gamma \in (z_1, \beta_1) \), or \( h_w^\gamma = \gamma \in (\beta_2, z_2) \), respectively. \( \blacksquare \)

**Corollary 4.2: Two dimensional space**

**Proof.** Given two core points \( z_1 = \{z_{11}, z_{12}\}, z_2 = \{z_{21}, z_{22}\} \) in the two dimensional space, the line segment \( L_\xi \) between them is described as follows:

\[
y - z_{12} = \frac{x - z_{12}}{z_{21} - z_{11}}, x \in (z_{11}, z_{21})
\]

(17)

Training \( z_1 \) and \( z_2 \) can get the following classifier:

\[
h_w^\zeta(x_i) = w_1^\zeta x_{i1} + w_2^\zeta x_{i2} + b, \{w_1^\zeta, w_2^\zeta, b\} \in (-\infty, +\infty)
\]

\[
s.t. \quad h_w^\zeta \cap L_\xi, t\tan \theta^\zeta = \left| \frac{z_{12} - z_{22}}{z_{11} - z_{21}} \right| = 0
\]

(18)

where \( \theta^\zeta \) is the angle between \( h_w^\zeta \) (see Figure 4(b)).

Similarly, the classifier \( h_w^\beta \) trained by \( \beta_1 = \{\beta_{11}, \beta_{12}\}, \beta_2 = \{\beta_{21}, \beta_{22}\} \) is subject to:

\[
h_w^\beta \cap L_\xi, y - \beta_{12} = \frac{x - \beta_{12}}{\beta_{22} - \beta_{12}}, x \in (\beta_{11}, \beta_{21})
\]

(19)

where \( L_\xi \) is the line segment between \( \beta_1 \) and \( \beta_2 \). Intuitively, the difference of \( h_w^\beta \) and \( h_w^\zeta \) is their constraint equation. Because \( (\beta_{11}, \beta_{21}) \subset (z_{11}, z_{21}) \), we can conclude:

\[
h_w^\beta \subset h_w^\zeta
\]

(20)

It aims to show \( h_w^\zeta \) cannot classify \( \beta_1 \) and \( \beta_2 \) when \( x \in (z_{11}, \beta_{11}) \) or \( x \in (\beta_{21}, z_{21}) \) in the constraint equation. But for any \( h_w^\beta \), it can classify \( z_1, z_2 \) correctly. \( \blacksquare \)

**Corollary 5: High dimensional space**

**Proof.** Given two core points \( z_1 = \{z_{11}, z_{12}, z_{13}, \ldots, z_{1m}\}, z_2 = \{z_{21}, z_{22}, z_{23}, \ldots, z_{2m}\} \), the Bounded Hyperplane \( S \) between them is:

\[
S := \\{x_i : x_{i1} \in (z_{11}, z_{21}), x_{i2} \in (z_{12}, z_{22}), \ldots, x_{im} \in (z_{1m}, z_{2m})\}\}

(21)

Training the two data points can get the following classifier:

\[
h_w^\zeta(x_i) = \sum_{d=1}^m w_d^\zeta x_{id} + C, \{w_d^\zeta, C\} \in (-\infty, +\infty)
\]

s.t. \( h_w^\zeta \cap S, \cos \theta^\zeta = \frac{\sum_{d=1}^m (w_d^\zeta)^2}{\sum_{d=1}^m (w_d^\zeta)^2 + C^2} \leq \frac{1}{2}
\]

(22)

where \( \theta^\zeta \) is the angle between \( h_w^\zeta \) and \( S \), \( v \) is the normal vector of \( S \). Given point \( p \), which is located on \( h_w^\zeta \), if \( p_1 \subseteq (\beta_{11}, z_{11}), p_2 \subseteq (\beta_{12}, z_{22}), \ldots, p_m \subseteq (z_{1m}, \beta_{2m}) \), in the positive class or \( p_1 \subseteq (z_{11}, \beta_{11}), p_2 \subseteq (z_{12}, \beta_{22}), \ldots, p_m \subseteq (z_{1m}, \beta_{2m}) \) in the negative class, \( h_w^\zeta \) cannot predict \( \beta_1 \) and \( \beta_2 \) correctly. It can also be described as follows: if \( h_w^\zeta \) segments the bounded hyperplane between \( z_1 \) and \( \beta_3 \), or \( z_2 \) and \( \beta_4 \), the trained \( h_w^\zeta \) can not classify \( \beta_1 \) and \( \beta_2 \). Then Lemma 2 is established. \( \blacksquare \)

**Figure 4:** (a) An example of \( h_w^\beta \subset h_w^\zeta \) in one-dimensional space. \( h_w^\beta, h_w^\zeta \) are two point classifiers. (b) An example of \( h_w^\beta \subset h_w^\zeta \) in two-dimensional space.

**Corollary 6: Multi-class classification**

**Proof.** Like the multi-class classification proof of Lemma 1, the multi-class problem can be segmented into \( k \) parts of binary classification problems. \( \blacksquare \)

**4 Enclosing Ball Boundary**

The hard-margin support data description in one-class classification is equivalent to the MEB (Minimum Enclosing Ball) problem which attempts to find the radius component of the radius-margin bound and its center. It is described in Section 4.1. To improve the fitting of the ball boundary, we split the ball of each cluster into two kinds of different small balls: C-ball (core balls) and B-ball (boundary balls), where core balls are located within the clusters, and boundary balls are located at the edge of clusters.

Our task of this section is to detect the B-ball of each cluster by calculating the increment of the ball radius (Section 4.2) and center (Section 4.3) when extending the local space, where the two types of interments of B-ball are bigger than that of C-ball. To enhance the difference of the two types of local features, we consider both radius and center updates to propose an optimization scheme in Section 4.4 and develop the LEB algorithm in Section 4.5. The time and space complexities then are analyzed in Section 4.5. Finally, the advantages of our approach are further discussed in Section 4.7.

**4.1 MEB in SVDD**

The MEB problem is to optimize the \( B(c, R) \):

\[
\min_{R,c} R^2 : ||c - x_i|| \leq R^2
\]

(23)

where \( R \) is the ball radius, and \( c \) is the ball center. The corresponding dual is to optimize:

\[
\mathcal{F}(c, R, \xi_i) = R^2 + C \sum_i \xi_i
\]

s.t. \( (x_i - c)^T (x_i - c) \leq R^2 + \xi_i, \forall i, \xi_i > 0 \)

(24)

where \( \xi_i \) is the relaxation variable and \( C \) is a user-defined parameter to describe \( \xi_i \). The optimization result is:

\[
R = \sqrt{\alpha diag(K) - \alpha'(K)\alpha}
\]

(25)

where \( \alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]^T \). According to the conclusion in [51] [52], \( diag(K) \) is close to a constant, and then the optimization task changes to:

\[
\max_{\alpha} -\alpha'(K)\alpha
\]

(26)
4.2 Update radius

Intuitively, the geometric volume of B-ball is larger than that of C-ball by the global characteristics description. Therefore, the local characteristics of radius update when adding more data to the current enclosing ball, will benefit the enhancement of characteristics scale.

Based on the global characteristics and the local characteristics of radius update when adding more data to the current enclosing ball benefits the enhancement of characteristics scale.

When the data \( \Phi(x_i) \) is added to \( B \) on time \( t \), the new radius is:

\[
R = \sqrt{\hat{I} \text{diag}(K^c) - \hat{I}'(K')\hat{I}'}
\]

where \( \hat{I} = [I_1, I_2, ..., I_t]_{T \times t} \), and \( K' \) is the updated kernel matrix after adding. Then, the square increment of the radius is:

\[
\Delta R^2_t = \hat{I}' \text{diag}(K') - \hat{I}'(K')\hat{I}' - \text{diag}(K) + IK I \quad \text{A} \quad \text{B}
\]

\[
= K'_t - (K'_t + K'_{nt}) \hat{I}'
\]

where \( \hat{I} = [I_1, I_2, ..., I_t]_{T \times t} \), \( K'_t = K(\Phi(x_i), \Phi(x_i)) \), \( K'_t \) is the \( t \)-th row of matrix \( K' \) and \( K'_{nt} \) is the \( t \)-th column of matrix \( K' \). Therefore, after \( k \)-times of adding, the kernel matrix changes to:

\[
T = \begin{bmatrix}
K_{tx} & D_{tx} (k-t) \\
E_{t-kx} & F_{t-kx} (k-t)
\end{bmatrix}
\]

Let \( \delta = [1, 1, ..., 1]_{T \times (k-t)} \), \( \nu = [1, 1, ..., 1]_{T \times t} \) and \( \nu = [1, 1, ..., 1]_{T \times (k-t)} \). The square increment of adding \( k-t \) features to \( x_i \) is close to:

\[
\ell(D,E,F) = \nu^T \text{diag}(F) - (\delta D \nu + \nu \text{E} \nu + \nu^T \text{F} \nu)
\]

In the kernel matrix, \( T(x_i, x_i) = \kappa \), therefore the optimization task changes to:

\[
\min_{\tau, \nu, \nu} (\tau \text{D} \nu + \nu \text{E} \nu + \nu^T \text{F} \nu)
\]

4.3 Update center

Path change of ball center when adding more data to the current enclosing ball is another important local characteristics to distinguish between B-ball and C-ball, where the length of path update of B-ball is bigger than that of C-ball.

Given \( c_t \) as the ball center of \( t \)-time, the optimization objective function is [54]:

\[
\min : ||c - c_t||^2
\]

s.t. \( R^2 \geq ||c - \Phi(x_i)||^2 \)

and the Lagrange equation is:

\[
\hat{L}(c, c_t, \Phi(x_i)) = ||c - c_t||^2 - \phi(R^2 - ||c - \Phi(x_i)||^2)
\]

On setting its derivative to zero, we obtain:

\[
\phi = \frac{||c - \Phi(x_i)||}{R} - 1
\]

\[
c_t = \phi c + (1 - \phi)\Phi(x_i)
\]

where \( \phi = \frac{R}{||c - \Phi(x_i)||} \). As such,

\[
\Delta c^2 = ||c_t - c||^2
\]

\[
= ||\phi c + (1 - \phi)\Phi(x_i) - c||^2
\]

\[
= ||1 - \phi||^2||\Phi(x_i) - c||^2
\]

\[
= \xi ||\Phi(x_i) - c||^2
\]

where \( \xi \) is a constant. Therefore, the increment of adding \( k-1 \) features to \( \Phi(x_i) \) can be written as:

\[
\Delta c^2 = \sum_{t \leq k} ||\Phi(x_i) - c||^2
\]

The matrix form is:

\[
\Delta c^2 = \hat{I} \text{diag}(T_t^T T_t)
\]

where \( \hat{I} = [1, 1, ..., 1]_{1 \times k} \).

4.4 Geometric update optimization

To enhance the difference of local geometric feature of B-ball and C-ball, we consider both the radius and center update and discuss the properties of the optimized objective function.

The kernel update of \( R \) and \( c \) is:

\[
\min_{\delta, \nu, \nu, t} \hat{I} \text{diag}(T_t^T T_t) + (\delta \text{D} \nu + \nu \text{E} \nu + \nu^T \text{F} \nu)
\]

Let \( t = 1 \) to calculate the update:

\[
J(T, K_{11}) = \hat{I} \text{diag}(T_1^T T_1) + (\delta \text{D} \nu + \nu \text{E} \nu + \nu^T \text{F} \nu)
\]

\[
= K_{11} + \tau \text{D} \nu + \nu \text{E} \nu + \nu^T \text{F} \nu + \Theta
\]

\[
= \mu \text{T} \mu + \Theta
\]

where \( \Theta = \hat{I} \text{diag}(T_1^T T_1) - K_{11}, \mu = [1, 1, ..., 1]_{k \times 1} \). Next, let us produce some properties for this objective function. The detailed proofs of the following lemmas are presented in the Appendix.

Lemma 3. Suppose that \( K(x_i, x_j) = \Re > K(x_i, x_j), \forall j \in (1, n), \) where \( \Re \geq 1, kR < J(T, K_{11}) < kR^2 + (k^2 - 1)R \). Otherwise, \( J(T, K_{11}) \rightarrow +\infty \) when \( k \rightarrow +\infty \).

Lemma 4. \( J(T, K_{11}) \) is a monotonically increasing function on \( k \).

Lemma 5. \( J(\beta) < J(\zeta) \).

4.5 LEB algorithm

Based on the conclusion of Lemmas 3-5, we find: \( J(\beta) < J(\zeta) \) and \( J(T, K_{11}) \) increases with the extension of local ball volume. Therefore, we propose an AL sampling algorithm called LEB (see Algorithm 1).

To calculate the update of radius and center, we need to capture the \( k \) neighbors of each data point. After initialization in Lines 1-3, Lines 6 calculates the kNN matrix \( C \) of \( X \) using the Kd-tree with a time consumption of \( O(n \log(n)) \). Then, Line 7-8 iteratively calculate \( J(\zeta(i), K(x_i, x_j)) \) by Eq. (39) and stores it in \( Z(i) \), where the used kernel function is RBF kernel.

However, radius and center updates of noises sometimes may be larger than that of ball boundary points. To smooth noises, Line 10 sorts \( 2 \lambda N + N_0 \) times by descending, where \( N_0 \) is the querying number. After sorting, the sorted
values of $Z$ are stored in matrix $V$ and their corresponding positions are stored in matrix $P$.

Intuitively, the data with an update value located in the interval of $[V(E(\lambda N^2)), V(1)]$ and $[V(E(\lambda N + N_q^2)), V(\lambda N^2)]$ is a noise and ball boundary point respectively, where $E(\cdot)$ is the round down operation. In other words, the input parameter $\lambda$ is an effective liner segmentation of noises and querying data by their update values of Eq. (39).

After capturing the update range of ball boundary points, Line 11 finds the position of queried data from matrix $P$ and Line 12-14 then return the queried data accordingly. Finally, the expert gives label annotation help for the queried data in Line 15.

### 4.6 Time and space complexities

In model-based approaches, the time complexity of training classifiers determines the time consumption of sampling process. Studying the time complexity of SVM is $O(N^2)$ to $O(N^3)$, we predict that Margin’s time cost will rise to $O((N_q + N_l)^3)$ to $O((N_q + N_l)^4)$ with a given query number of $N_q$, where $N_l$ is the number of labeled data. For Hierarchical AL approach, hierarchical clustering is its main time consumption process that costs $O(N^2)$. Similarly, calculating the kernel matrix also costs the time price of $O(N^2)$ in TED. Although Re-active is a novel idea, it still needs to visit the whole version space to select a data point by approximately $N$ times of SVM training. It means that the time complexity of one selection will cost $O(N^2)$ to $O(N^4)$ and the time consumption of sampling $N_q$ data points will be $O(N_q N^2)$ to $O(N_q N^4)$. (The detailed descriptions of Hierarchical, TED, and Re-active are present in Section 5.1.)

In our LEB approach, Line 6 uses the Kd-tree to calculate the $k$NN matrix of data set $X$ and the time complexity is $O(N \log(N))$, Line 7-8 cost $O(N)$ to calculate the radius and center update of each data point, Line 10 costs $O(N N_q)$ for sorting, and Line 11-14 return the boundary points of $X$. After that, we will train the boundary points within a short time $\varepsilon$. Therefore, the total time complexity $T$ is

$$
T = O(N \log(N)) + O(N^2) + O(N N_q) + O(N_q) + O(\varepsilon)
$$

Standard SVM training has $O(N^2)$ space complexity when the training set size is $N$. It is thus computationally expensive on data sets with a large amount of samples. By observing the iterative sampling process of model-based AL approaches, we conclude these approaches cost $O((N_q + N_l)^2)$ space complexities. However, our LEB approach uses the tree structure to calculate the $k$NN matrix, which costs cheaply with a space consumption of $O(N)$. Therefore, the space complexity of LEB is lower than that of other model-based AL approaches.

### 4.7 Advantages of LEB

Our investigation finds that many existing AL algorithms which need labeled data for training are model-based and suffer from the model curse. To describe this problem, we have summarized the iterative sampling model in Algorithm 2. In its description, Line 6-10 calculate the uncertainty function, Line 11 finds the position of the data with the maximum uncertainty, where $i \Leftarrow$ represents this operation, and Line 12-13 update the labeled and unlabeled set. After $N_q$ times iterations, Line 15-16 train the classifier and returns the error rate of predicting $X$.

Interestingly, different labeled data will lead to various iterative sampling sets because of $h$ is always retrained after updating $X_l$ and $X_u$. Then the $\rho$ matrix must be recalculated in each iteration. In addition, some AL algorithms work in special scenarios, for example: (1) the margin-based AL approaches only work under the SVM classification; (2) Entropy-based AL only works under the probabilistic classifier or probability return values. Table II lists the properties summary of different AL approaches.

From the analysis results, we can find the reported approaches all need iterative sampling, the support of labeled data, and the high time consumption. We can observe that many AL algorithms pay too much attention on the uncertainty of the classification model since the unfamiliarity of which data are their main sampling objects. However, our proposed LEB algorithm does not need any iteration and labeled data to sample, and also can be trained by any available classifier whatever in binary or multi-class settings.

### 5 Experiments

To demonstrate the effectiveness of our proposed LEB algorithm, we evaluate and compare the classification performance with existing algorithms in eight clustering data sets (structured data sets) since they have clear geometry boundary, and a letter recognition data set (unstructured data set) to observe its performance. The structure of this section is: Section 5.1 and 5.2 describe the related baselines and tested data sets, respectively; Section 5.3 describes the experimental settings and analyzes the results, and Section 5.4 discusses the time and space performance of different AL approaches.

---

**Algorithm 1. LEB**

**Input:** data set $X$ with $N$ samples, number of queries $N_q$, nearest neighbor number $k$, noise ratio $\lambda$

**Output:** Queried data $X_l$

1: Initialize: $i \leftarrow 0$
2: $\omega \leftarrow 0$
3: $X_l, C, Z, V, P \leftarrow \varnothing$
4: **Begin:**
5: **for** $i=1$ to $N$ **do**
6: Calculate the $k$NN of $x_i$ by Kd-tree and store them in $C(i)$
7: Let $T \leftarrow C(i)$, then calculate $\mathcal{J}(C(i), K(x_i, x_i))$ using Eq. (39)
8: and store in $Z(i)$
9: **endfor**
10: $[V, P] \leftarrow $ sort($Z$, descending, $\lambda N + N_q$)
11: $\omega \leftarrow P(E(\lambda N) : E(\lambda N + N_q))$
12: **for** $i=1$ to $N_q$ **do**
13: add $X(\omega(i))$ to matrix $X_l$
14: **endfor**
15: Query the labels of all data of $X_l$
Table 2: Properties of different active learning strategies. Y’ represents ‘Yes’, ‘N’ represents ‘No’, ‘Uncertain’ represents the time consumption is hard to evaluate since it relates to the sampling number or time complexity of classifier.

| Approach      | Model       | Iteration | Label support | Classifier | Multi-class | Time consumption |
|---------------|-------------|-----------|---------------|------------|-------------|-----------------|
| Margin        | SVM         | Y         | Y             | Y          | Uncertain   | Uncertain       |
| Entropy       | Uncertain probability | Y | Y | Probability classifier | Y | Uncertain | O(1/N²) |
| Hierarchical  | Clustering  | Y         | Y             | Any        | Y           | O(1/N²)         |
| TED           | Experimental optimization | Y | Y | Any | Y | Uncertain | O(1/N²) |
| Re-active     | Maximize the model difference | Y | Y | Any | N | Uncertain, but high | O(1/Nlog(N)) |
| LEB           | Enclosing ball boundary | N | N | Any | Any | O(1/Nlog(N)) |

Algorithm 2. Iterative sampling

**Input:** X
- number of queries N_q
- labeled data X_l

**Output:** prediction error rate \( \epsilon \)

1: **Initialize:** uncertainty function \( \mathcal{P}(x) \),
2: \( i, j \leftarrow 0 \),
3: \( \rho \leftarrow \emptyset \),
4: unlabeled data \( X_u \leftarrow X - \mathcal{P} \) and it has \( N_u \) data
5: while \( i < N_q \)
6: \( \text{for } i=1:N_u \)
7: \( h = \text{train}(X_l) \)
8: \( \text{calculate the } \mathcal{P}(x_i) \text{ based on } h \)
9: \( \text{store it in matrix } \rho(i) \)
10: \( \text{endfor} \)
11: \( \Lambda \leftarrow \rho \bowtie \text{argmax}(\rho(u)) \)
12: \( X_l(A) \leftarrow \emptyset; X_u \leftarrow [X_u, x_i] \)
13: \( \text{update } N_u \)
14: \( \text{endwhile} \)
15: \( h = \text{train}(X_l) \)
16: **return** \( \epsilon = \text{err}(h, X) \)

5.1 Baselines

Several algorithms have been proposed in the literature [3, 10, 22, 24], and will be compared with LEB, where Random is an uncertainty sampling without any guidance, Margin is based on SVM, Hierarchical is a clustering-based AL approach, TED is a statistical experimental optimization approach, and Re-active is an idea of maximizing the model differences:

- **Random**, which uses a random sampling strategy to query unlabeled data, and can be applied to any AL task but with an uncertain result.
- **Margin** [5], which selects the unlabeled data point with the shortest distance to the classification model, only can be supported by the SVM [26, 27, 28].
- **Hierarchical** [10] sampling is a very different idea, compared to many existing AL approaches. It labels the subtree with the root node’s label when the subtree meets the objective probability function. But incorrect labeling always leads to a bad classification result.
- **TED** [22] favors data points that are on the one side hard to predict and on the other side representative for the rest of the unlabeled data.
- **Re-active** [21] learning finds the data point which has the maximum influence on the future prediction result after annotating the selected data with positive and negative labels. This novel idea does not need to query the label information of unlabeled data when relabeling, but needs a well-trained classification model at the beginning. Furthermore, its reported approach cannot be applied in multi-class classification problems without extension.

5.2 Data sets

We compare the different algorithms’ best classification results on some structured data sets [30], and one unstructured letter recognition data set _letter_.

- g2-2-30 [31]: 2048 × 2. There are 2 adjacent clusters in the data set.
- Flame [34]: 240 × 2. It has 2 adjacent clusters with similar densities.
- Jain [35]: 373 × 2. It has two adjacent clusters with different densities.
- Path-based [33]: 300 × 2. Two clusters are close and surround by an arc cluster.
- Spiral [33]: 312 × 2. There are three spiral curve clusters which are linear inseparable.
- Aggregation [22]: 788 × 2. There are 7 adjacent clusters in the data set and noises connect them.
- R15 [34]: 600 × 2. There are 7 separate clusters and 8 adjacent clusters.
- D31 [34]: 3100 × 2. It has 31 adjacent Gaussian clusters.
- letter [37, 38]: 20000 × 16. It is a classical letter recognition data set with 26 English letters. We select 5 pairs letters which are difficult to distinguish from each other to test the above AL algorithms in a two-class setting. They are DvsP, EvsF, IvvsJ, MvsN, UvsV, respectively. For multi-class test, we select A-D, A-H, A-L, A-P, A-T, A-X, A-Z, respectively. Of these, A-D is the letter set A to D, and A-H is the letter set A to H, ... , A-Z is the letter set A to Z. The seven multi-class sets have 4, 8, 12, 16, 20, and 26 classes respectively.

In addition to the introduction for the tested data sets, all two-dimensional data sets are shown in Figure 5.

5.3 Results

We report the experimental results of the compared algorithms over the synthetic and real data sets in this section. Figure 6 marks the cluster boundary points on data sets of Aggregation and Flame. It is used to show the definition of enclosing ball boundary points. Figure 7 shows the sampling process in Flame data set. Observe that of sub figure (a) shows the LEB algorithm can detect the optimal training data point of each class when the querying number is 2. In the querying process of (b) and (c), LEB selects the effective
Table 3: The statistical results (mean±std) of different AL algorithms on classical cluster data sets.

| Data sets | Num_C | Algorithms | 1% | 5%  | 10%  | 15%  | 20%  | 30%  | 40%  | 50%  | 60%  |
|-----------|-------|------------|----|-----|------|------|------|------|------|------|------|
| g2-2-30  | 2     | Random     | 0.96 | 0.99 | 1.00 | 0.99 | 0.98 | 0.98 | 0.97 | 0.96 | 0.95 |
|           |       | Margin     | 0.50 | 0.50 | 0.51 | 0.51 | 0.51 | 0.50 | 0.50 | 0.50 | 0.50 |
|           |       | Hierarchical | 0.50 | 0.50 | 0.51 | 0.51 | 0.51 | 0.50 | 0.50 | 0.50 | 0.50 |
|           |       | TEB        | 0.63 | 0.65 | 0.66 | 0.66 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 |
|           |       | Re-active LEB | 0.56 | 0.52 | 0.54 | 0.53 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 |
|           |       | Flame      | 0.72 | 0.72 | 0.72 | 0.72 | 0.72 | 0.72 | 0.72 | 0.72 | 0.72 |
|           |       | Jain       | 0.69 | 0.70 | 0.71 | 0.71 | 0.71 | 0.71 | 0.71 | 0.71 | 0.71 |
|           |       | Pathbased  | 0.44 | 0.44 | 0.44 | 0.44 | 0.44 | 0.44 | 0.44 | 0.44 | 0.44 |
|           |       | Spiral     | 0.38 | 0.38 | 0.38 | 0.38 | 0.38 | 0.38 | 0.38 | 0.38 | 0.38 |
|           |       | Aggregation | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 |
|           |       | R15        | 0.37 | 0.37 | 0.37 | 0.37 | 0.37 | 0.37 | 0.37 | 0.37 | 0.37 |
|           |       | D31        | 0.40 | 0.40 | 0.40 | 0.40 | 0.40 | 0.40 | 0.40 | 0.40 | 0.40 |

Figure 5: The classical clustering data sets. (a) g2-2-30 (b) Flame (c) Jain (d) Pathbased (e) Spiral (f) Aggregation (g) R15 (h) D31.

Figure 6: The marked cluster boundary points of Aggregation and Flame are in blue circles.

data sets and some special results are bold. The detailed experiment settings are as follows: (1) we use the MATLAB random function to implement the Random algorithm and calculate the mean and std (standard deviation) values after running it 100 times; (2) as the Margin, Hierarchical and Re-active algorithms all need the labeled data points to enclosing ball boundary points to obtain good training models, in which their prediction results are promised.

Table 3 shows the classification results on some synthetic
algorithms will show poorer performance if we use random
includes all the different label kinds of data set, or the
and std values in order to guarantee that the labeled set
test the algorithms 100 times and then calculate the mean
each class and query their labels, respectively. Similarly, we
guide the training process, we select one data point from
beginning number of queries in each group experiment. In all sub figures, Hiera is the abbreviation of Hierarchical, REAL
settings. (f)-(l) are the multi-class settings. The class number are 2,2,2,2,4,8,12,16,20,24, and 26 respectively, which are the
Figure 7: The AL process on Flame, where $A_i$ represents the $i$th sampling data point, Acc represents the prediction Accuracy.

Figure 8: The SVM classification results of different AL approaches on the letter data set. (a)-(e) are the binary classification
settings. (f)-(l) are the multi-class settings. The class number are 2,2,2,2,4,8,12,16,20,24, and 26 respectively, which are the beginning number of queries in each group experiment. In all sub figures, Hiera is the abbreviation of Hierarchical, REAL is the abbreviation of Re-active.

guide the training process, we select one data point from each class and query their labels, respectively. Similarly, we
test the algorithms 100 times and then calculate the mean
and std values in order to guarantee that the labeled set includes all the different label kinds of data set, or the
algorithms will show poorer performance if we use random
selection; (3) there are two important parameters for the TED algorithm: the kernel function parameter $\sigma$ and the
regularization parameter for the kernel ridge regression $\lambda$. We use a super parameter $\sigma=1.8$ to generate the kernel matrix and train $\lambda$ from 0.01:0.01:1. The reason for this is that this parameter will provide an important guidance for
the sampling selection. After we test it many times, we limit its correct and stable range; (4) for our LEB algorithm, we train the parameters $k$ form $2:1:5\%N$ and noise ratio $\lambda=1:0.01:0.3$ to record the classification result. Because $\lambda$ segments the noises and points within a cluster, we use a super parameter $\lambda=30\%N$ as the maximum [12]. The classifier trained in the classification experiment is LIBSVM [39][40] and we use its default parameters settings without any tuning action.

We marked some specific results in bold to analyze the performance of the algorithms. The observation shows: (a) Random provides a fast sampling strategy which is not sensitive to data number and dimensions or class number. But its performance is always bad for the first query as it cannot select valuable data points using a random strategy. (b) Margin is a popular AL approach which selects the data points that are closest to the current classification line or plane. The results in the literature [5] show it is a good AL approach. But, this paper is the first to use the challenging two-dimension clustering data sets in AL and the experiment results show a drawback of Margin. That is, it has separation class bias, as it always selects the noises and boundary points between adjacent classes since their calculated distances are small. Therefore, an unfair and unreasonable sampling strategy always selects the data points distributed in the most adjacent area in Jain, then returns a bad classification result (refer to the bold results for Margin in the Jain data set in Table 3). (c) Hierarchical is a special AL approach which uses pre-clustering to judge whether the subtree nodes can be labeled with the same label as the root node. The experimental results show this AL approach can obtain good classification results when the data sets have structured clusters, such as the R15 data set, and it outperforms the other algorithms when labeling 1% data points. (d) Selecting the most uncertain data points to label is applied in the TED approach, which also pays attention to representative data points. But our experiment shows it is very slow and sensitive to parameters. (e) Re-active uses an innovate idea of selecting the data points which have the greatest influence on the prediction model after adding the queried data points to the labeled set, where noises may be their main sampling objects since they will change the performance of current prediction model greatly. (f) The experiments using LEB show that it can get a very strong classification result with fast accuracy acceleration at the beginning since the queried boundary points are effective.

Figure 9 reports a group of optimal classification results for different algorithms on real data sets under training their parameters. In high dimensional space, the performance of these AL algorithms is interesting: (a) Random is still stable as discussed in the previous analysis. (b) Margin becomes stable in the high dimension space since the data points are distributed sparsely and no adjacent classes with high density attracts the selection process. (c) Hierarchical performs poorly in the high dimensional space in this set of tests. After rechecking the algorithm, we find the real reason which leads to this phenomenon is that there is no obvious hierarchical clustering results. Especially for some multi-class data sets, most of the data points are clustered into one class. Then, the algorithm will wrongly label the large class using its label. Wrong clustering results make the algorithm degrading. (d) TED is still stable due to its good sampling strategy. (e) Re-active’s sensitivity to noises disappears since there is no noises in the letter data set. Then, strong classification results are generated. (f) For our LEB algorithm, its performance is still relatively good.

### 5.4 Time performance

Intuitively, the time complexity of LEB is neither label-based nor model-based. Compared with the general AL approaches, its time complexity is fixed and less. When $N_q$ increases, the time complexity of general iterative model will dramatic increase. To analyze the time performance of above compared AL approaches, here we show a set of experiments involved to the running time test on data size, dimension, class number in Figure 9. The parameter settings are $N_q=100$, $\lambda=0.01$ in TED, $\lambda=0.7$ in LEB. In Figure 9(a), we set Dimension= 2, Class= 2, and vary the Gaussian synthetic data set number from 1,000 to 40,000. In Figure 9(b), we set Number= 1,000, Class= 2 and vary the dimension from 2 to 700. In Figure 9(c) we set Number= 1,000, Dimension= 2 and vary the class number from 2 to 30. The result shows that our proposed algorithm’s running time is shorter than others’ and not sensitive to dimension or class number.
6 Conclusion

Querying the labels of unlabeled data is essential in many AL problems. However selecting the most informative and representative instances from the massive number of arbitrary candidates by a limited number of querying times can be problematic due to the selection difficulty. To address this issue, we proposed an AL sampling approach which reduced the uncertainty sampling space to an enclosing surface of a high dimensional ball in each cluster. To prove the boundary points located on the hyperplane are more effective compared with other data, various cases including binary classification, multi-class of low and high dimensional space were discussed. Then, we set the enclosing boundary points detection task as the MEB problem. Observing an over-fitting issue that the ball boundary hypothesis rate of LEB in low dimensional space shows that re-breaking the uncertainty sampling curse. Viewing the redundant rate of LEB in low dimensional space were discussed. Then, we set the enclosing boundary points detection task as the MEB problem. Extending the local space helped us to detect the boundary balls. Therefore, $J(T, K_{11}) < k\mathbb{R}$ is set up. Meanwhile, $\mu^T T \mu = \sum_{i,j} K(t_i, t_j) < k^2 K(t_i, t_i) = k^2 \mathbb{R}$ and $\Theta = K_{11}K_{11} + \sum_{j \neq 1} ||K(t_1, t_j)||^2 - K_{11}$ (44)

Therefore, $J(r, v, \nu) < k\mathbb{R}^2 + (k^2 - 1)\mathbb{R}$ is set up. $\square$

Lemma 4.

Proof. According Eq. (39), we rewrite $J(T, K_{11})$ as:

$$J(k) = \sum_{i \neq j} K(t_i, t_j) + k\mathbb{R} + \sum_{j \neq 1} ||K(t_1, t_j)||^2 + K_{11}K_{11} - K_{11}$$

where $\mu^T T \mu = \sum_{i \neq j} K(t_i, t_j) + k\mathbb{R}$. Then, we obtain:

$$\Delta = J(k) - J(k - 1) \sum_{j \neq 1} K(t_1, t_j) + \sum_{i \neq 1} K(t_1, t_1) + ||K(t_1, t_k)||^2 + \mathbb{R} > 0$$

(46)

Therefore, Lemma 4 is set up. $\square$

Lemma 5.

Proof. Given a $k$-clique $G = (V_1, V_2, ..., V_k)$ in graph structure, and the used kernel density estimation function is:

$$f(G) = \sum_{i,j} K(V_i, V_j)$$

Then, $J(T, K_{11}) = f(G) + \sum_{j \neq 1} f^2(V_1, V_j) \rightarrow f(G)$

(48)

For B-balls, the density of them are lower than that of C-balls. Therefore, $J(\beta) < J(\zeta)$. $\square$

Acknowledgment

The authors would like to thank Kai Yu who provided the TED code, Chih-Jen Lin who provided the LIBSVM package, and Dongming Li who provided the help of proofreading.

Appendix

Lemma 3.

Proof. Of the $\mu^T T \mu$, its matrix type is:

$$\mu^T T \mu = \text{tr}(T) + \sum_{i \neq j} K(t_i, t_j)$$

$$= k\mathbb{R} + \sum_{i \neq j} K(t_i, t_j)$$

(41)

where $T_{ij} = K(t_i, t_j), \sum_{i \neq j} K(t_i, t_j) > 0$. For $\Theta$,

$$\Theta = I_{\text{diag}}(T \mu^T T \mu) - K_{11}$$

$$= K_{11}K_{11} + \sum_{j \neq 1} ||K(t_1, t_j)||^2 - K_{11}$$

$$= \sum_{j \neq 1} ||K(t_1, t_j)||^2 + \Theta > 0$$

(42)

where $\Theta = K_{11}^2 - K_{11} > 0$ in the given suppose. Then, $J(T, K_{11}) > k\mathbb{R}$ is set up. Meanwhile,

$$\mu^T T \mu = \sum_{i,j} K(t_i, t_j) < k^2 K(t_i, t_i) = k^2 \mathbb{R}$$

and $\Theta = K_{11}K_{11} + \sum_{j \neq 1} ||K(t_1, t_j)||^2 - K_{11}$ (44)

Therefore, $J(r, v, \nu) < k\mathbb{R}^2 + (k^2 - 1)\mathbb{R}$ is set up. $\square$

References

[1] D. Cohn, L. Atlas, and R. Ladner, “Improving generalization with active learning,” Machine learning, vol. 15, no. 2, pp. 201-221, 1994.
[2] S. Hao, J. Lu, P. Zhao, et al., “Second-order Online Active Learning and Its Applications,” IEEE Transactions on Knowledge and Data Engineering, 2017, DOI 10.1109/TKDE.2017.2778097.
[3] S. Xiong, J. Azimi, and X. Fern, “Active learning of constraints for semi-supervised clustering,” IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 1, pp. 43-54, 2014.
[4] D. Lewis, and W. Gale, “A sequential algorithm for training text classifiers,” Proc. 17st Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 3-12, 1994.
[5] S. Tong, and D. Koller, “Support vector machine active learning with applications to text classification,” Journal of Machine Learning Research[JMLR], pp. 45-66, 2001.
[6] N. Roy, and A. McCallum, “Diverse ensMEBles for active learning,” Proc. 18th Int’l Conf. Machine Learning (ICML), pp. 441-448, 2001.
[7] P. Melville, and R. Mooney, “Diverse ensMEBles for active learning,” Proc. 21st Int’l Conf. Machine Learning (ICML), pp. 584-591, 2004.
[8] H. Seung, M. Opper, and H. Sompolinsky, “Query by committee,” Proc. 5th Int’l Conf. learning theory (COLT), pp. 287-294, 1992.
[9] C. Tosh, and S. Dasgupta, “Diameter-Based Active Learning,” Proc. 34th Int’l Conf. Machine Learning (ICML), pp. 3444-3452, 2017.
