Kumar P, Gupta A. Active learning query strategies for classification, regression, and clustering: A survey. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 35(4): 913–945 July 2020. DOI 10.1007/s11390-020-9487-4

Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey

Punit Kumar and Atul Gupta, *Member*, *ACM*, *IEEE*

Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh 482005, India

E-mail: {punit.kumar, atul}@iiitdmj.ac.in

Received February 16, 2019; revised January 13, 2020.

Abstract Generally, data is available abundantly in unlabeled form, and its annotation requires some cost. The labeling, as well as learning cost, can be minimized by learning with the minimum labeled data instances. Active learning (AL), learns from a few labeled data instances with the additional facility of querying the labels of instances from an expert annotator or oracle. The active learner uses an instance selection strategy for selecting those critical query instances, which reduce the generalization error as fast as possible. This process results in a refined training dataset, which helps in minimizing the overall cost. The key to the success of AL is query strategies that select the candidate query instances and help the learner in learning a valid hypothesis. This survey reviews AL query strategies for classification, regression, and clustering under the pool-based AL scenario. The query strategies under classification are further divided into: informative-based, representative-based, informative- and representative-based, and others. Also, more advanced query strategies based on reinforcement learning and deep learning, along with query strategies under the realistic environment setting, are presented. After a rigorous mathematical analysis of AL strategies, this work presents a comparative analysis of these strategies. Finally, implementation guide, applications, and challenges of AL are discussed.

Keywords active learning, active learning query strategy, active classification, active regression, active clustering, deep active learning

1 Introduction

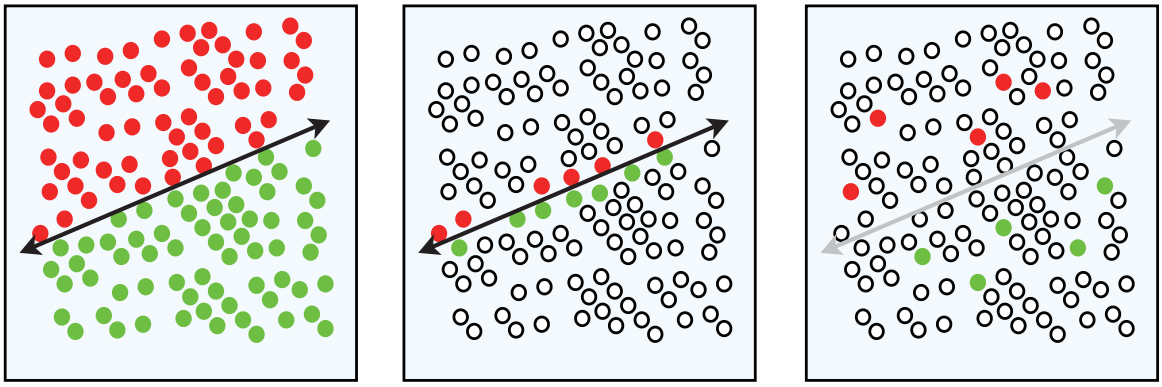
Over the past few decades, there is a need that the computer learning should be as alike as of human learning, i.e., a program should improve its result after seeing more evidence or should learn some task with experiences [1]. For this, the machine learning task starts with collecting the set of observations, which corresponds to feature vectors or unlabeled dataset. In supervised machine learning techniques, all the available unlabeled data instances are first labeled manually by the domain experts. Then hypothesis learning is started using these labeled data instances: the hypothesis function maps the feature set to its corresponding label set. Manual labeling is easy, like in spam filtering, but difficult in text classification, speech annotation, protein structure prediction, as labeling requires significant computation [2–4]. Now the issue is whether it is essential to label all the available unlabeled data instances for good generalization result or small labeled data instances are enough for learning, as shown in Fig.1.

1.1 Learning with a Little Labeled Data

The challenge of learning from a little labeled data instances is that the traditional supervised machine learning algorithms may produce the undesired results because instances may not be a good representative of the dataset. Then instances should be selected, which have a direct impact on forming the decision boundary Fig.1(c). This process significantly reduces the labeling efforts, and the research resulted in two learning techniques.

1) *Semi-Supervised Learning.* A learner starts with hypothesis learning on a small amount of unlabeled data and uses it to analyze the unlabeled instances. The learner looks for certain unlabeled instances, which can be classified correctly by the hypothesis. These instances, along with their labels, are then added to the labeled dataset. A new hypothesis is now learned from

Survey



(a) (b) (c)

Fig.1. Example of a two-class classification problem using: (a) true classification using fully labeled data, (b) lucky classification using little labeled data, and (c) target classification using little labeled data (green and red dots represent the labeled data of two classes and white circles are unlabeled data).

this new labeled dataset, and this procedure is repeated until convergence [5].

2) *Active Learning* (*AL*). A learner starts with a massive amount of unlabeled and a little amount of labeled data and learns a hypothesis function on labeled data. Then, the learner chooses the instances from the unlabeled dataset, about which the learned hypothesis function is the least confident in predicting their label, opposite to semi-supervised learning, which selects the most certain instances. This instance selection helps in reducing the misclassification error because the least certain instances contribute more toward it. The selected unlabeled instances are called query instances. Then, the learner queries the label of these instances from the domain expert/oracle. After getting the label of the query instance, the instance along with its label is added to the labeled training set. Now, the hypothesis is updated using this modified dataset, which results in a hypothesis closer to the target one. This process repeats until the preset stopping criteria are statisfied or the process runs out of the labels. AL has wide range applications: text classification [2], remote sensing image classification [3], protein structure prediction [4], etc. However, the assumption here is that there exists an expert labeler that always returns the correct label of queried instances and is available without fatigue [6–8].

This paper reviews the AL way of solving a problem by selecting the most critical instances. The critical components for the success of AL are: 1) a good query strategy for selecting the most critical data instances from the unlabeled dataset; 2) an efficient machine learning algorithm to learn the hypothesis model. This survey works on the first component of AL, i.e., how to choose the most critical instances from the unlabeled dataset. A good query strategy will help AL to achieve the goal of minimizing the output generalization error with less query instances.

1.2 Working Scenarios of AL

Many query strategies have been proposed in the literature for unlabeled instance selection, under different working scenarios. These working scenarios are divided into three main categories: 1) pool-based; 2) stream-based; 3) membership query synthesis based. A brief description of these is as follows.

1) *Pool-Based AL Scenario.* When the learner has all the unlabeled data instances available before the start of the learning process, then the learning is known as pool-based AL [9]. This large set of unlabeled instances forms a pool. Fig.2(a) shows a representation of pool-based AL process. The hypothesis function or model is learned using available labeled instances. After building the hypothesis, the learner uses it to evaluate the unlabeled instances from the pool and find whether to query its label or not. If the instance qualifies preset selection criteria, then it becomes a candidate query instance, and its label is queried from the oracle; otherwise the learner will choose another instance from the unlabeled dataset.

After getting the label of a candidate instance, this instance along with its label, is added to the labeled dataset. The hypothesis is then again learned on the updated labeled dataset. This process is repeated until some preset performance criterion is reached or the learner runs out of labels.

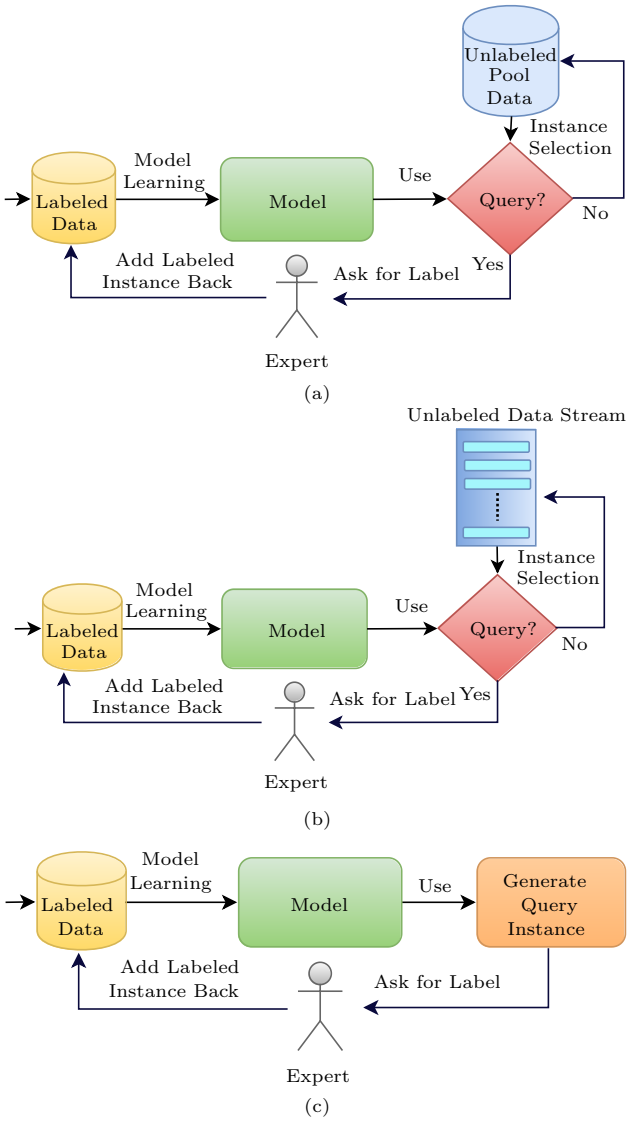


Fig.2. Representation of different AL scenarios. (a) Pool-based. (b) Stream-based. (c) Membership query synthesis based.

2) *Stream-Based AL Scenario.* The unlabeled instances float like a stream of instances, or the instances are sampled from some distribution one after another [10]. The process flow under this scenario is shown in Fig.2(b). Here also, hypothesis function is learned on labeled data. The learner then analyzes the stream of unlabeled instances using some evaluation criteria based on learned hypothesis and decides in real time whether to query the unlabeled instance or not. If an unlabeled instance qualifies the evaluation criteria, then it becomes a candidate query instance, and the learner queries its label to the oracle; otherwise this instance is discarded, and the learner looks for the next unlabeled instance from the stream. After this, the process remains the same as previous. However, setting an evaluation criterion in the stream-based scenario is more difficult, as compared with the pool-based one, because of the sequential availability of data instances [11]. If the drift or change in the distribution of streaming data with time is considered, then the problem of instance selection becomes more complicated. Then, it requires some modification in the hypothesis model with the change in the data distribution [12, 13].

3) *Membership Query Synthesis Based AL Scenario.* The learner generates query instances in the input space based on the learned hypothesis model [14]. The query instances are directly synthesized near the learned decision boundary. This synthesis led to very fast query instance generation. A representation of this AL process is shown in Fig.2(c). The starting steps of building the hypothesis model from a few labeled instances are similar to the two previous scenarios. Then the obtained model is used to generate the candidate query instance in the input space to query from oracle. Now, the process remains the same as previous [6].

This AL scenario is not so good for several problems: vision-based learning, natural language processing, and speech-based task processing, because a human annotator may not recognize the generated query instance. A combination of membership query synthesis and pool-based scenarios exists in the literature to overcome this limitation. The combined method has the advantages of fast query synthesis and also removes the intractable queries. This combination is done by synthesizing query instances near to the decision boundary and selecting its nearest available original instance as a query instance [15].

This work proceeds with query strategies under pool-based scenario. Initially, this work includes query strategies with the base assumptions: the oracle is non-noisy and always returns the correct label; the query strategies deal with the balanced datasets; the oracle charges uniformly for each label; the cost of misclassification is uniform; query strategies deal with the single label, single instance and only learn a single task at a given time instance. This work also discusses query strategies with relaxed assumption.

1.3 Contributions

There are other good survey papers like, [6] by Settles in 2010, [16] by Sun and Wang in 2010, [17] by Fu *et al.* in 2012 and [18] by Aggarwal *et al.* in 2014, available in AL literature. Settle provided a very elegant survey of AL along with description query strategies [6]. Then,

Fu *et al.* incorporated the query strategies that consider the correlation among features, labels, and their combination [17]. Aggarwal *et al.* provided another important survey with a good description of AL and explored it for some advanced scenarios [18]. However, all these are lagging in a systematic categorization of AL query strategies.

Apart from the contents covered by the existing surveys, our work provides further exploration and a systematic categorization of AL to provide an overall picture to the learner. Along with this, the query strategies under realistic environment are also discussed, where some of these assumptions are relaxed. This study also reviews the empirical and theoretical guarantees of AL query strategies and also helps the beginners in implementing the AL framework. Finally, this work provides a comparative study of these query strategies and also provides a comprehensive summary of major application areas of AL.

This section provided the motivation and background of AL. Section 2 discusses the environment settings and variables used. Section 3 presents the detail of various existing AL query strategies in the pool based scenario. Section 4 provides the detail of AL query strategies in a realistic environment. Section 5 and Section 6 review the mathematical and the empirical evaluation of AL query strategies respectively. Section 7 helps the learner in implementing the AL query strategy framework. Section 8 gives the summary of application areas of AL. Section 9 and Section 10 identify some challenges of this field and provide the conclusion, respectively.

2 Preliminary

Let D be a dataset that forms the input space and includes the two types of instances, the set of labeled instances, D^L, and the set of unlabeled instances, D^U. Let n be the total number of instances in D and n_l, n_u be the number of labeled and unlabeled instances in D^L and D^U, respectively. The i-th instance of D^L is denoted by $(\boldsymbol{x}^{(i)}, y^{(i)})$, where each $\boldsymbol{x}^{(i)} \in X$ is a d-dimensional feature vector, i.e., $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, ..., x_d^{(i)})$ in feature space X and $y^{(i)} \in Y$ is its corresponding regression or classification value from label space Y[①]. Similarly, the i-th instance of D^U is denoted by $(\boldsymbol{x}^{(i)}, ?)$, where $\boldsymbol{x}^{(i)}$ is again a d-dimensional feature vector in X and "?" denotes that the label is unknown. For the given labeled dataset D^L, the learner tries to learn a hypothesis function, $h \in H$, that maps each d-dimension feature vector $\boldsymbol{x}^{(i)} \in X$ to the corresponding label $y^{(i)} \in Y$, i.e., $h(X, Y) : X \to Y$, where H is the hypothesis space. Let $E_{\text{in}}(h)$ and $E_{\text{out}}(h)$ be the in-sample and the out-of-sample error of hypothesis function h respectively. The in-sample error corresponds to the misclassification error on the training data by hypothesis h. The out-of-sample error corresponds to the misclassification error on all the input space by hypothesis h. The out-of-sample error is also known as the generalization error. All the notations used in the manuscript are described in Table 1 to make the reading more convenient.

3 Pool-Based AL Query Strategies with Base Assumptions

AL is a vast area of machine learning, which includes many techniques of instance selection. Putting all of them at one platform is difficult. This work categorizes the existing query strategies and also gives their comprehensive detail. Based on the domains of machine learning, the query strategies under the pool-based scenario can be classified into three categories: query strategies for classification, regression, and clustering, as shown in Fig.3.

3.1 Pool-Based Query Strategies for Classification

This is an extensively investigated area, and many strategies for querying the unlabeled data instances have been proposed in the literature under this area. These query strategies are further divided into three categories by the nature of instances queried: informative-based, representative-based, informative- and representative-based, and others query strategies, as shown in Fig.3. The remaining section describes strategies under this categorization.

3.1.1 Informative-Based Query Strategies

The strategies under this scheme are based on the informativeness of the input instances. The more the hypothesis function is uncertain in classifying the input data instance, $\boldsymbol{x}^{(i)} \in D^U$, the more the informativeness of that instance will be, and the more its chances to become a candidate query instance. The informative-based query strategies use uncertainty as an evaluation criterion for selecting the candidate query instance.

[①]Some parts of the manuscript abuse the notations and use $x \in X$ for input instance and $y \in Y$ for the class label.

Table 1. Different Symbols Used and Their Description

Symbol	Description	Symbol	Description
$\underline{a}, \underline{b}, \underline{c}, \underline{d}, \underline{e}, f, g$	Hyperplances	p_c^i	Probability of belonging to class c
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{u}, \boldsymbol{v}$	Arbitrary vectors	$\boldsymbol{P}$	Probability vector
B	Ball of the given radius	Pr	Probability distribution
c	A class	$\hat{p}$	Probability estimate
CC	Collective classifier	q_k	Bounds for the number of queries
CO	Content classifier	r	Radius of the ball
C, C', C'', C_j	Different clusters	SV	Support Vectors
$\mathscr{C}$	Change in the model's parameters	$S^{(i)}$	The i-th subset of the dataset
C_s	Committee size	V	Version space
$\mathbb{C}$	Combined instance selection measure	$\mathbb{V}$	Vector embedding
D, D^L, D^U	Full, labeled, and unlabeled dataset	v_l	Validation set
d	Feature's dimension	ν	Set of vertices
$d_{i,j}$	Distance between the i-th and the j-th nodes	$\boldsymbol{w}$	Weight vector
d_c	Cut-off distance	$x^i \in X$	The i-th instance that belongs to the instance set
$\hat{d}$	VC-dimension	$\hat{x}$	Candidate instance
$\mathfrak{d}$	Probabilistic difference	$y^i \in Y$	The i-th label that belongs to the label set
e	Set of edges	$\hat{y}_j^{(i)}$	The j-th popular label for the i-th instance
$E_{\text{in}}, E_{\text{out}}$	In and out of sample error respectively	z	Number of clusters
E	Expectation	?	Used for unknown labels
f	Function mapping	$+, -$	Labels for binary classes
G	Graph	λ	Regularization parameter
g	Gradient computation time	Λ, ϕ	Lipschitz parameter
$H, h, \bar{h}, h^{(S)}, h^*, h_i$	Different hypotheses	λ_k^D	Maximum data diameter of a cell
$\mathbb{H}$	Hash function	γ	Kernel's parameter
J, J', J^*	Cost function	π, σ	Equation's parameters
K	Kernel function	α, β	Continuity and noise parameters
k	Clusters' level	ρ_i	Density of the i-th instance
m	Sample complexity	Ω	Learning rate
m_p	Number of model's parameters	ω	Minimum distance between nodes
m_n	Number of models	$\hat{\theta}$	Disagreement coefficient
m_i^+, m_i^-	Size of margins	Θ	Model parameters
M_3, M_2	Distribution of unlabeled and labeled data	$\bigtriangledown$	Gradient
$\boldsymbol{M}_1$	Similarity matrix	τ	Fraction of Pr used in ψ-splitting
n_n	Number of labeled instances in the neighborhood	ψ	Splitting index for hypothesis
n_l, n_u, n	Number of labeled, unlabeled, and full data	ϑ	The angle between $x^{(i)}$ and w
$\mathbb{N}$	Natural numbers	$\hat{\eta}$	Regression function
$\mathcal{N}$	Gaussian distribution	ϵ	Generalization error
$\tilde{O}$	Time complexity	δ	Cut-off probability
$P, p1, p2$	Probabilities	ξ	Clusters' center

Based on this, many query strategies have been proposed in the literature that greedily query those unlabeled instances, which are near to the decision boundary. The description of these strategies is presented as follows.

1) *Uncertainty Sampling.* Uncertainty sampling is one of the most commonly used frameworks for AL, which was proposed by Lewis and Gale in 1995 [19]. The learner selects an input instance, about the label of which, the hypothesis function is the least certain. The least certain instances lie near the decision boundary, while the far-away instances are most confident. The most certain instances are treated as redundant because they do not contribute much to the formation of the decision boundary [20].

In case of the two-class classification problem, using the probabilistic model, the learner chooses an unlabeled instance to query, whose posterior probability is near about 0.5 [9]. In the case of a multi-class classification problem, the query instance is selected using (1) [21].

$$\hat{\boldsymbol{x}} = \arg \max_{\boldsymbol{x}^{(i)} \in D^U} (1 - P(\hat{y}_1^{(i)} / \boldsymbol{x}^{(i)})), \tag{1}$$

where $P(\hat{y}_1^{(i)} / \boldsymbol{x}^{(i)})$ is the probability of the first popular class label $\hat{y}_1^{(i)}$ — that has the highest posterior probability — for instance, $\boldsymbol{x}^{(i)}$, and $\hat{\boldsymbol{x}}$ represents the obtained candidate query instance under this strategy. This method selects the unlabeled instance by considering the information of best-predicted class label while ignoring the information for the other labels. As

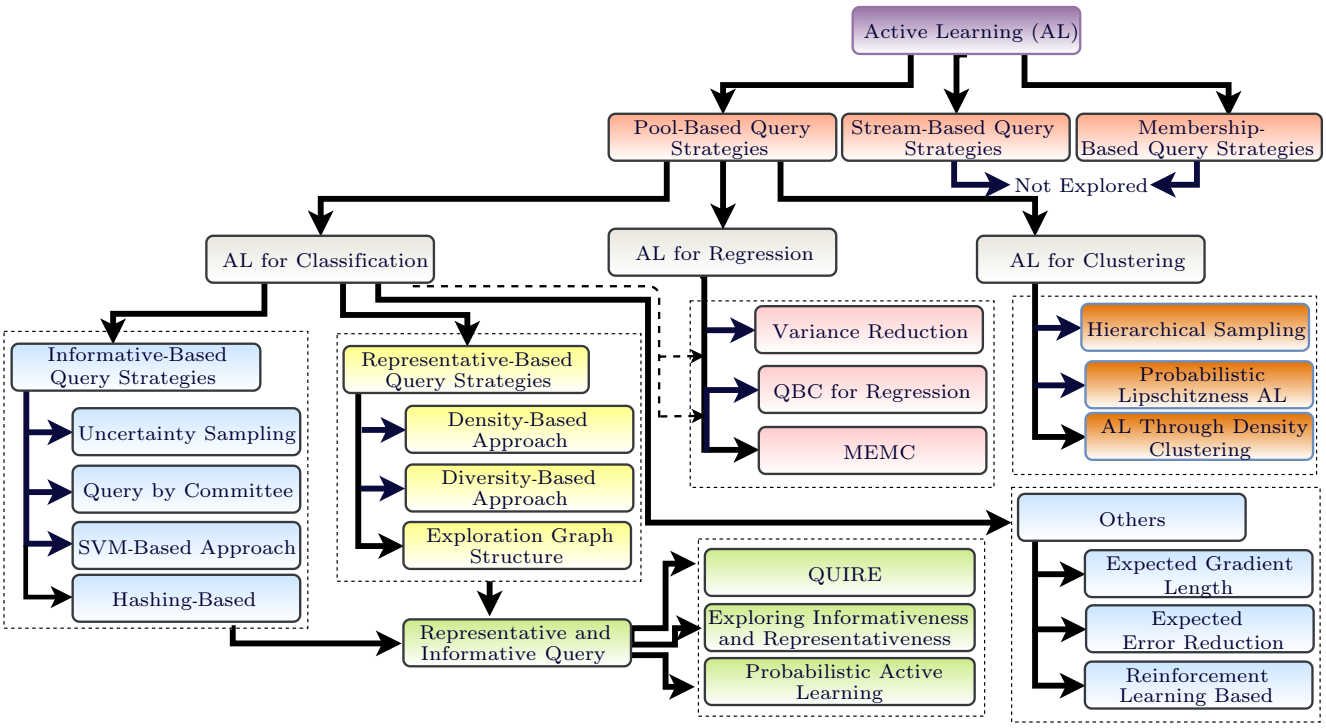


Fig.3. Hierarchical categorization of AL query strategies. Query strategies are connected by the dotted line, used in both classification and regression.

this model is not utilizing the information of the labels, this may lead to a wrong instance selection. An improvement over the existing model has been proposed in [21] to rectify this issue. The improved model selects two most popular class labels, $\hat{y}_1^{(i)}$ and $\hat{y}_2^{(i)}$, corresponding to each unlabeled instance, $\boldsymbol{x}^{(i)}$, which have the highest posterior probabilities. Out of these instances, the instance that has a minimum difference between the two most confident posterior probabilities as per (2), is chosen as the candidate query instance.

$$\hat{\boldsymbol{x}} = \arg \min_{\boldsymbol{x}^{(i)} \in D^U} (P(\hat{y}_1^{(i)}/\boldsymbol{x}^{(i)}) - P(\hat{y}_2^{(i)}/\boldsymbol{x}^{(i)})). \quad (2)$$

The problem with this strategy is that instance selection becomes complicated as the number of class labels increases. Another typical measure of uncertainty is the entropy [22], which measures the purity of the sample and is given by (3) [22].

$$\hat{\boldsymbol{x}} = \arg \max_{\boldsymbol{x}^{(i)} \in D^U} -\sum_j P(y^{(j)}/\boldsymbol{x}^{(i)}) \log P(y^{(j)}/\boldsymbol{x}^{(i)}), \quad (3)$$

where $P(y^{(j)}/\boldsymbol{x}^{(i)})$ is the conditional probability of class label $y^{(j)}$ for given unlabeled instance $\boldsymbol{x}^{(i)}$ and j ranges over all possible class labels [23]. This strategy reduces the maximum expected risk without updating the model for each unlabeled instance. Therefore, this type of measures is fast and quite easy to use.

2) *Query by Committee* (*QBC*). Seung *et al.* proposed QBC in 1992 [24]. A representation of the QBC process is shown in Fig.4. QBC is a disagreement-based learning, which maintains a committee of the hypotheses. These hypotheses are trained using a different subset of samples drawn from the labeled dataset. The condition for QBC is that the set of hypotheses that are learned must be consistent with the given labeled data instances. The unlabeled instance with the maximum disagreement of committee members in label prediction is qualified as a candidate query instance. The learner then queries the label of that identified instance.

For accurate working of this method, each committee member should represent the different region of the version space. The version space is the space formed by the set of all hypotheses, which are consistent with the training set [1]. Abe and Mamitsuka proposed boosting and bagging methods, also known as the ensemble method, for making a committee of the hypotheses [25]. The logic behind the QBC framework is to minimize the version space by querying the unlabeled instance in the controversial region of the input space, with the minimum number of queries. Melville and Mooney pro-

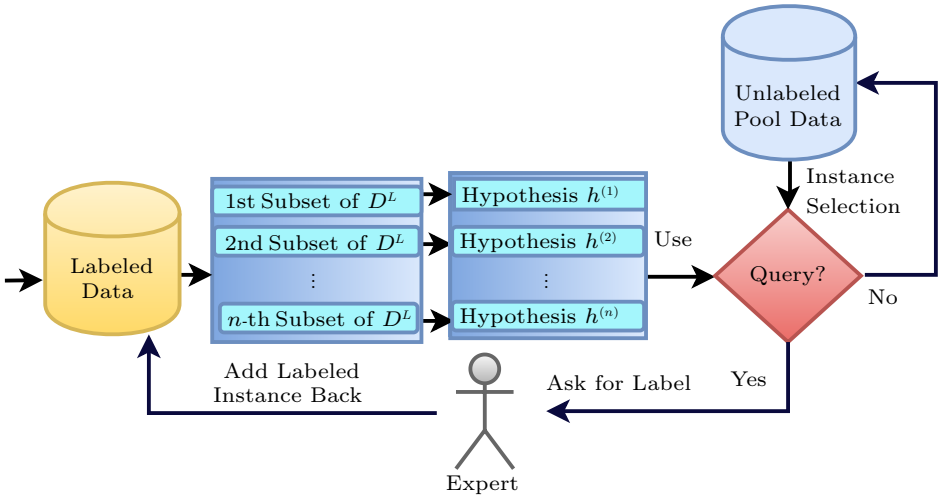


Fig.4. Representation of QBC query strategy.

posed a modification in QBC that the learner should not only form the committee of those members, which covers the whole version space, but also prefer the disagreement among the committee members [26], because forming a large committee that has members representing nearly the same hypothesis, is not so useful. Muslea *et al.* used only the committee of two members in QBC, which represent the whole version space with maximum disagreement [27].

3) *Support Vector Machines (SVM) Based Approach.* SVM is the supervised machine learning classifier that learns the linear decision boundary, having the maximum distance from the nearest training data instances [28]. It is a well-studied model for supervised learning and also performs very well in AL, due to its inherent property of separating the training data instances with the maximum margin. In this query strategy, the learner selects those unlabeled instances to query, which are closer to the decision boundary [29].

Tong and Koller used the concept of version space with SVM, for selecting the unlabeled instances to query [30]. As SVM is a linear classifier, then the version space exists if the training data instances are linearly separable. For the non-linear training dataset, the data instances are mapped to a higher dimension using some kernel function to make them linearly separable in a new space. This kernel trick satisfies the linear separability assumption of training data, and hence the version space can be formed using SVM on non-linear datasets also.

After this, learning aims to reduce the version space as fast as possible. For this, Tong *et al.* used the duality between the feature space and parameter space [30]. Fig.5(a) represents a feature space, in which circles/rectangles represent instances and separating boundary by line/plane. Colored circles/rectangles represent the labeled data, and the maximum margin obtains the decision boundary from the labeled data. Fig.5(b) represents the dual form of feature space, where colored hyperplanes represent the labeled instances, and the dotted black hyperplanes represent the unlabeled instances. Here, the weight vector, $\boldsymbol{w}_i$, represents the decision boundary learned on the available labeled data instances and the area between colored hyperplanes forms the version space.

The description of schemes for selecting the candidate query instance by the maximum reduction of version space, up to half in each iteration, is as follows.

1) *Simple Margin.* SVM learns a weight vector, $\boldsymbol{w}_i$, from the input feature vectors, $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(i)}\}$, and their corresponding labels, $\{y^{(1)}, y^{(2)}, ..., y^{(i)}\}$. This weight vector forms the center of the largest hypersphere that can fit the version space with the margin equal to the radius of the hypersphere, as shown in Fig.5(b). Now, the learner chooses the input instance from the unlabeled data, i.e., dotted hyperplane, in such a way that the chosen instance is closest to the vector $\boldsymbol{w}_i$. For this, the learner finds the distance of each unlabeled input instance and chooses the instance with the smallest distance. For example, the hyperplane, "$\underline{d}$" in Fig.5(b) represents the candidate query instance because it divides the version space maximally. This scheme works with the assumptions that the version space is regular and $\boldsymbol{w}_i$ forms the center of version space, which is not always true as represented in Fig.5(c).

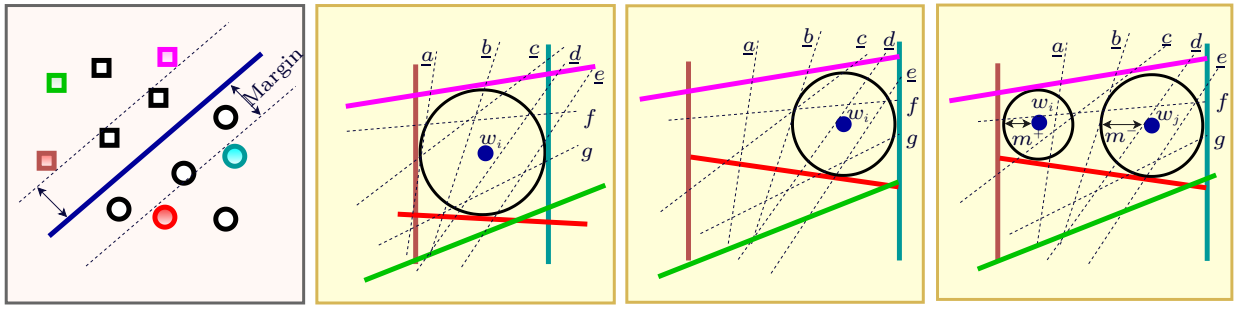


Fig.5. Version space reduction with SVM-based query strategy. (a) Representation of boundary learned on label data using SVM classifier (labeled data is represented by colored circles/rectangles and unlabeled data by black circles/rectangles). (b) Instance selection using simple margin method in dual representation of (a) to reduce the version space (colored hyperplanes represent the corresponding labeled instances, and dotted hyperplanes represent the corresponding unlabeled instances). (c) Example of limitation of simple margin. (d) Instance selection using the max-min method to reduce the version space.

2) *MaxMin Margin.* This approach overcomes the difficulty of simple margin method by working with an elongated version space. In this, SVM runs twice for each unlabeled instance, in the case of the two-class classification problem. In the first round, it adds the unlabeled instance in the training set by assigning "+" label (say for class 1). In the second round, the same unlabeled instance is added by assigning "–" label (say for class 2). Then, the learner looks for the size of the margin, which is equal to the radius of the fitted hypersphere. Let m_i^+ and m_i^- be the size of the margin obtained after assigning "+" and "–" labels to the unlabeled instance $\boldsymbol{x}^{(i)}$ respectively, as shown in Fig.5(d). Then, the learner chooses the instance $\boldsymbol{x}^{(i)} \in D^U$, which has the maximum value of $\min(m_i^+, m_i^-)$ as a candidate query instance.

3) *Ratio Margin.* This scheme is very similar to the MaxMin margin method, which also takes care of elongated version space. For each unlabeled input instance, the learner calculates the margins m_i^+ and m_i^- by running SVM twice, as in the MaxMin approach. The instance $\boldsymbol{x}^{(i)} \in D^U$, having the maximum value of $\min\left(\frac{m_i^-}{m_i^+}, \frac{m_i^+}{m_i^-}\right)$, is selected as a candidate query instance.

Along with this, there are some other methods for maximally narrowing the version space by choosing the input instance closer to the separating hyperplane [32, 33]

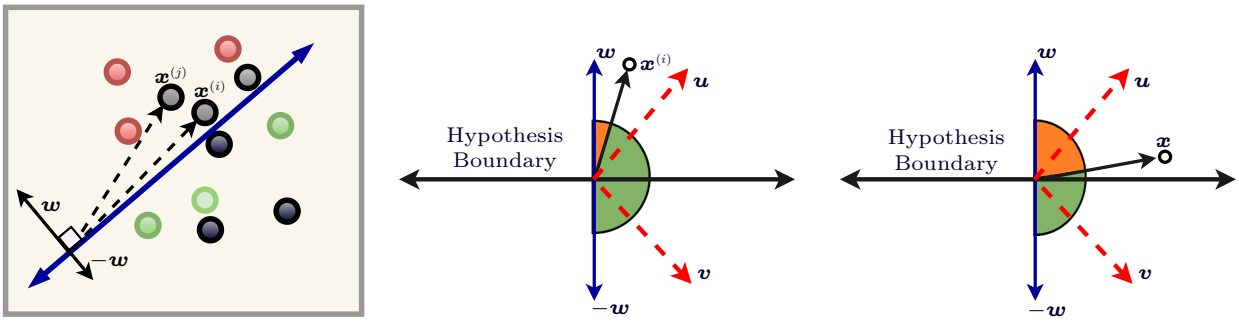
4) *Hashing-Based Approach.* While working with a massive set of unlabeled input instances, it is very computationally expensive to evaluate each instance, at each iteration of instance selection. As a linear search of candidate query instance is problematic with the massive pool, the hashing-based approach can be used to resolve this issue.

Most commonly used approach is locality sensitive hashing (LSH), introduced by Indyk and Motwani for nearest neighbor search [34]. LSH uses a randomized hash function mapping similar keys to the same bucket. This scheme reduces the search time by allowing to search only in the subset of the dataset after hashing the unlabeled instances. Here, the learner works with the assumption of having the maximum probability of collision for those input instances, which are close or similar to each other. In other words, the similarity of the instances is inversely proportional to the distance. Gionis *et al.* used several hash functions for mapping the data points [35]. Jain *et al.* proposed two hashing algorithms for the nearest neighbor to a query hyperplane (NNQH) approach, to retrieve the unlabeled instances which are closer to the hyperplane [36]. These two formulations for NNQH are given as follows.

The first one is to measure the locality sensitiveness among the data instances by the angle between the data points, $\boldsymbol{x}^{(i)}$, and a normal to the separating hyperplane, $\boldsymbol{w}$, as shown in Fig.6(a). Consider a separating hyperplane passing through the origin and a unit vector, $\boldsymbol{w}$, normal to this hyperplane. The unlabeled input instances, which are closer to the hyperplane and also nearly perpendicular to vector $\boldsymbol{w}$, are of interest. Let $\vartheta_{\boldsymbol{x}^{(i)}\boldsymbol{w}}$ be the angle between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$, as a measure of distance $ds(\boldsymbol{x}^{(i)}, \boldsymbol{w})$ for NNQH. Now, consider the two-bit hash function by Jain *et al.* for NNQH [36, 37]. A mathematical formulation of the two-bit hash function for input feature vector $\boldsymbol{x}^{(i)}$ is given by (4).

$$\begin{aligned}\mathbb{H}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) &= [\mathbb{H}_{\boldsymbol{u}}(\boldsymbol{x}^{(i)}), \mathbb{H}_{\boldsymbol{v}}(\boldsymbol{x}^{(i)})] \\ &= [\mathrm{sign}(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{x}^{(i)}), \mathrm{sign}(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{x}^{(i)})], \quad (4)\end{aligned}$$

where $\mathbb{H}_{\boldsymbol{u}}(\boldsymbol{x}^{(i)})$ will be 1, if $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{x}^{(i)} > 0$; otherwise 0,



(a) (b) (c)

Fig.6. Hashing using angle-based similarity. (a) Angle-based locality measurement. (b) Case where the instance x is closer to weight vector $\boldsymbol{w}$. (c) Case where the instance is closer to the decision boundary.

and $\boldsymbol{v}$ and $\boldsymbol{u}$ are sampled from d-dimensional Gaussian distribution. The vector $\boldsymbol{v}$ is used to determine the angle between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$, and the vector $\boldsymbol{u}$ is used to determine the angle between $\boldsymbol{x}^{(i)}$ and $-\boldsymbol{w}$. If the angle between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$ is small, then the angle between $\boldsymbol{x}^{(i)}$ and $-\boldsymbol{w}$ is likely to be very large, as shown in Fig.6(b), and hence the bits of this hash function are likely to be different. When $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$ are approximately perpendicular, as shown in Fig.6(c), then the bits produced are likely to be the same. This helps the learner to map unlabeled data instances into two categories, i.e., near and far from separating hyperplane. To retrieve the candidate instance, the search space is reduced by hashing the hyperplane vector $\boldsymbol{w}$ using the hash function as:

$$\mathbb{H}_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) = [sign(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}), -sign(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{w})].$$

The second approach for NNQH is embedded hyperplane hashing. Here, the learner embeds the data points and a separating boundary into the Euclidean space in such a way that the distance between them in the embedded space is preserved, as shown in Fig.7. The formulation for embedding a d-dimensional vector, say $\boldsymbol{a}$, into a d'-dimensional space is presented in [38], which is given by (5).

$$\mathbb{V}(\boldsymbol{a}) = (a_1^2, a_1a_2, ..., a_1a_d, a_2^2, a_2a_3, ..., a_d^2) \in \mathbb{R}^{d'}, \quad (5)$$

where $d' = d(d+1)/2$ and a_i denotes the i-th element of $\boldsymbol{a}$. Consider two unit vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The embeddings of these vectors in the Euclidean space are given by $\mathbb{V}(\boldsymbol{a})$ and $\mathbb{V}(\boldsymbol{b})$ respectively. The Euclidean distance between $\mathbb{V}(\boldsymbol{a})$ and $\mathbb{V}(\boldsymbol{b})$ is given by $|\mathbb{V}(\boldsymbol{a}) - \mathbb{V}(\boldsymbol{b})| = 2 + 2(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b})^2$ [39]. Now, minimizing the distance between $\mathbb{V}(\boldsymbol{a})$ and $\mathbb{V}(\boldsymbol{b})$ is simply minimizing $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$. For the categorization of data in this embedded hyperplane, an embedding hyperplane hash (EH-Hash) function, $\mathbb{H}_{\boldsymbol{u}}(\mathbb{V}(\boldsymbol{a})) = \text{sign}(\boldsymbol{u}^{\mathrm{T}}\mathbb{V}(\boldsymbol{a}))$, is defined based on the Euclidean distance, where $\boldsymbol{u}$ is a d'-dimensional vector sampled from Gaussian distribution, $\boldsymbol{u} \sim \mathcal{N}(0,1)$ [40]. Similarly, the hyperplane vector $\boldsymbol{w}$ is also hashed using, $\mathbb{H}_{\boldsymbol{u}}(\mathbb{V}(\boldsymbol{w})) = \text{sign}(-\boldsymbol{u}^{\mathrm{T}}\mathbb{V}(\boldsymbol{w}))$, to find out the target set of instances, from which the candidate instance can be identified easily. Any instance selection strategy can use this as preprocessing for better performance.

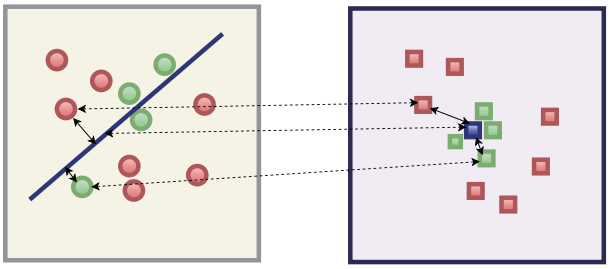


Fig.7. Embedded hyperplane hashing.

• *Challenges of Informative Query Strategies.* The above-discussed query strategies are based on the theme of selecting candidate query instances that are near to the decision boundary. Various query strategies under this category are quite intuitive, easy to implement and make the area of AL easily understandable. The drawback of this scheme is that it does not consider the relationships among the unlabeled instances and treats each input instance as independent and identically distributed (i.i.d) [17]. The informativeness of each instance is obtained independently, and the instance with high informativeness is selected as a candidate query instance. This process results in the selection of multiple instances of the similar type and hence generates the bad set of candidate instances as shown

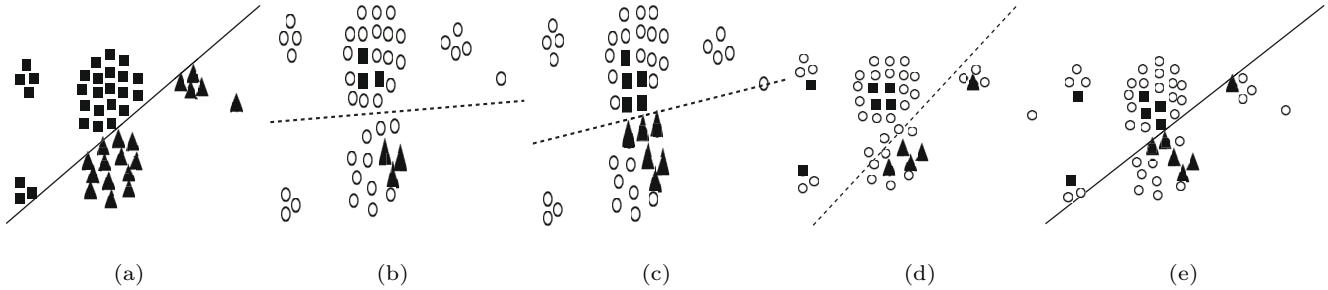


Fig.8. Representation for different classifications. (a) True classification. (b) Classification on initially labeled dataset. (c) Classification after querying informative input instances. (d) Classification after querying representative input instances. (e) Classification after querying both informative and representative input instances.

pictorially in Fig.8(c). All the instances selected are from a small region of input space and other regions of input space are not involved in building the hypothesis; this results in the problem of sampling bias. Another problem with this category is that it is prone to query outliers [41]: as shown in Fig.8(c), a single instance that lies on decision boundary seems to be most informative. However, querying this instance does not have a significant impact on the hypothesis model and querying this may lead to wastage of resources.

3.1.2 *Representative Based AL*

This scheme tries to fully utilize the information structure of unlabeled data for obtaining the candidate query instances. Also, it tries to remove the difficulties faced by the informative query strategies. The details of the existing query strategies under this scheme are given further in this subsection.

1) *Density-Based Approach.* In this strategy, the learner tries to select the candidate query instance from high-density input space, which serves as the representative instance. As the instance selection is made from high-density space, this automatically removes the problem of outliers faced by different query strategies under the informative scheme. Wu *et al.* used distance as a measure of representativeness [42]. The distance of the selected unlabeled input instance is calculated from all the other remaining instances, as a measure of density and the process is repeated for every unlabeled instance. The instance with the minimum value of the distance is selected as the candidate query instance.

Another method proposed in the literature is the clustering-based measure of representativeness [43]. In this method, the clustering of input space is done, and the cluster centers are identified. The nearest neighbor instance corresponding to each cluster center forms the set of representative instances. However, the performance of this method depends on the refinement of clusters returned by the clustering method.

Another way to measure the representativeness is similarity-based technique, which uses the correlation between feature vectors. Fu *et al.* discussed some popular ways of finding similarities, such as cosine similarity, KL divergence, and Gaussian similarity [17]. However, this type of correlations is not suitable for all types of problem scenarios. As in text mining, different words have a similar meaning, and some words behave differently in different scenarios. Therefore, semantic-based similarity schemes are useful in this context.

2) *Diversity-Based Approach.* In a parallel environment, multiple annotators are available to speed up the learning task. Then the learners need to select multiple unlabeled instances to query, which is known as batch mode learning. However, the selection of multiple instances at a time using the same informative criteria may lead to redundancy in the candidate set. Then, the diversity-based approach is used to overcome this problem. First of all, this approach was investigated by Brinker for batch mode learning [44]. Then, Hoi *et al.* used this batch mode AL for text categorization [45] and medical image classification [46].

Wu *et al.* utilized diversity on all the labeled instances, instead of only on the batches [42]. Initially, let $\boldsymbol{x}^{(i)} \in D^U$ be selected as the candidate query instance by the criterion of minimum distance from separating hyperplane. This instance is added to D^L after getting the label from the expert. Now, to select another unlabeled candidate instance, the diversity measure with D^L is also calculated along with minimum distance measure. To calculate the diversity measure, the angles between the feature vector of the selected instance and the feature vectors of remaining instances of the labeled data set are determined. This strategy selects candidate instances that are diverse to make a

useful training dataset. The formulation of diversity is by (6).

$$Diversity(\boldsymbol{x}) = 1 - \max_{\boldsymbol{x}^{(i)} \in D^L} \frac{|K(\boldsymbol{x}^{(i)}, \boldsymbol{x})|}{\sqrt{K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}), K(\boldsymbol{x}, \boldsymbol{x})}}, \quad (6)$$

where K is the kernel function used on the inner product of the data instances. The problem with this method is that the diversity measure alone can produce a misleading result, thereby researchers used this measure with other measures like density or informativeness for better performance [47].

3) *Exploration of Graph Structure.* In this query strategy, the active learner uses the graph structure for selecting the candidate query instances. The nodes (ν) of the graph G represent the training instance, and the weight on the edges (E) between the nodes represents the correlation among the nodes of graph $G = (\nu, e)$. Now, the aim is to use this correlation in AL for labeling the unlabeled instances, i.e., given some labeled data and correlation information, the inference about the labels of their neighbors can be obtained. Therefore, it reduces the labeling task and increases prediction accuracy.

Typical machine learning classification algorithms treat each input instance independently, without considering the underlying structure and their correlation information. This method uses collective classification [48, 50] for predicting the class of unlabeled instances, which considers three types of correlation into account, which are given as follows:

1) correlation between the label of node ν and its attributes;

2) correlation between the label of node ν and the labels of neighbors including their observed attributes;

3) correlation between the label of node ν and attributes of unobserved neighbors.

Here, the information about the node and its neighbors has been taken into account, instead of treating each instance independently. For example, the documents that cite each other generally have the same topics, and social networks also have the same scenario. Bilgic *et al.* proposed a collective classification based approach under AL for networked instance. In this approach, the learner utilizes the information of graph structure along with local and collective classification information [51]. Their proposed approach uses iterative classification algorithm with two types of classifiers, collective and local, for the evaluation of unlabeled instances [49]. The collective classifier (CC) is a probabilistic classifier given by, $P(y/x, aagr(N_i))$, where $aagr(N_i)$ is the aggregate of label information of the neighbors of the i-th instance. The other classifier that refers only to the features of the local nodes and learns $P(y/x)$ is called content classifier (CO) or local classifier.

The first step of this process is to create the clusters, say C_j, of nodes ν, where j varies upto the number of clusters. For every unlabeled node in the cluster, say $\nu_i \in C_j$, predictions of most likely labels by CC and CO, and the popular label of already observed nodes are required. Then the local disagreement, LD for node ν_i, is the entropy of the predicted class distribution. Then, the disagreement score is calculated for each cluster, which is given by the sum of all these local disagreements (Dis), and further given by (7).

$$Dis(CC, CO, C_j, D^L) = \sum_{\nu_i \in C_j} LD(CC, CO, \nu_i, D^L). \quad (7)$$

After computing the disagreement score for each cluster, the learner selects z clusters with the highest disagreement score. From these z clusters, the instance with the highest local disagreement score is selected as the candidate query instance, and its label is queried. The learner repeats this process until the budget is exhausted or performance requirements are satisfied.

• *Challenges of Representative Based AL.* The query strategies under this scheme use structure learning methods, which try to find out representative instances of the input dataset. An example of the decision boundary learned after querying the representative unlabeled instances is shown in Fig.8(d). This scheme removes the problem of sampling bias by selecting the instances from different regions of the input space, and the redundant instance selection problem is also eliminated. It also tackles the problem of querying the outliers, which exists in an informative case. However, this scheme may require more query instances in order to achieve the target decision boundary. This categorization may result in slow convergence. Also, it is hard to find all the representative instances of the input space.

3.1.3 Informative and Representative Based AL

From the discussion presented earlier about the informative and representative query strategies, it can be inferred that there is a trade-off between these two measures. If the informativeness of a chosen instance increases, then there may be a chance in the reduction of its representativeness.

None of the schemes discussed can make a perfect selection of unlabeled instances to form a good candidate query set. The use of the combination of these two schemes has been proposed in the literature. Wang and Ye in [52] used informative and representative criteria for effectively selecting the instances to minimize the empirical risk. Settles and Craves in [41] also proposed an information density based scheme, which combines these two measures: for informativeness, QBC or uncertainty sampling approach is used; and for representativeness, similarity-based techniques are used. Nguyen and Smeulders proposed a dynamic usage of these two measures for the selection of query instances [53].

The strategies discussed above provide an ad-hoc solution to the problem. Details of some more existing sound strategies that combine these two schemes for instance selection are given further in this section.

1) *Query Informative and Representative Examples* (*QUIRE*). This method was proposed by Hung *et al.* in 2014, which is based on the min-max framework of learning [54, 55]. QUIRE connects informative and representative components. This strategy uses the prediction accuracy based on the labeled data instances as a measure of informativeness and the prediction accuracy based on the unlabeled data instance as the measure of representativeness.

Hoi *et al.* initially used the min-max framework of learning for SVM-based batch mode AL in semi-supervised mode [55]. (8) gives the evaluation function of SVM for maximum margin setting with regularization.

$$J(l,f)=\sum_{i=1}^{n_l} l(y^{(i)},f(\boldsymbol{x}^{(i)}))+\frac{\lambda}{2}||f||_H^2, \tag{8}$$

where l refers to the loss function, f the hyperplane function and λ the regularization parameter. Using the evaluation function given in (8), the hypothesis model $\hat{f}$ is learned on the labeled input instances, which is given by (9).

$$\hat{f}=\arg\min_{f\in H} J(l,f). \tag{9}$$

The model utilizes the margin-based approach, therefore the chosen instance lies nearer to the boundary for small values of $\hat{f}(\boldsymbol{x}^{(i)})$. The formulation for selecting an instance near the boundary is given by (10).

$$\hat{\boldsymbol{x}}=\arg\min_{\boldsymbol{x}^{(i)}\in D^U}\max_{y=\{+,-\}}|\hat{f}(\boldsymbol{x}^{(i)})|. \tag{10}$$

This equation corresponds to the min-max view of AL and $\hat{\boldsymbol{x}}$ is guaranteed to be the most informative unlabeled instance. For query instances to be both informative and representative, the objective function should include both labeled and unlabeled input instances for learning. However, this is like a hypothetical situation as the labels for unlabeled instances are not known. With the help of manifold assumption presented in [56], the learner can have the solution, which minimizes the evaluation function. Therefore, let x_s be the test data from unlabeled instances, and the remaining unlabeled instances are $D^{U'}=D^U-x_s$. Also, let $y_s\in\{+,-\}$ be the label for x_s and for all the remaining unlabeled instances, label y_u will be from $\{+,-\}^{n_u-1}$. The modified evaluation function is given by (11).

$$\begin{aligned}&J'(l,f)\\&=\min_{y_u\in\{+,-\}^{n_u-1}}\max_{y_s\in\{+,-\}}\min_{f\in H}\sum_{i=1}^{n} l(y^{(i)},f(\boldsymbol{x}^{(i)}))+\\&\frac{\lambda}{2}||f||_H^2.\end{aligned} \tag{11}$$

The instance $x_s\in D^U$, with the smallest value of evaluation function (11) is selected as a candidate instance. Although this strategy provides a good set of candidate query instances, it requires a semi-supervised manifold assumption for unlabeled data instances.

2) *Exploring Representativeness and Informativeness for AL* (*ERIAL*). Du *et al.* proposed this approach in 2017 [57]. It uses "two-sample discrepancy test" [58] for representativeness and "modified best vs second best" (Bvs2B) [59] to construct uncertainty measure for informativeness of each data instance. The two-sample discrepancy test is used to determine the discrepancy between two samples drawn independently from two multivariate distributions of the same domain [60]. In AL, for each unlabeled instance, distribution discrepancy is obtained between the labeled and the unlabeled datasets. This test requires the estimate, say M_2, for the distribution of labeled data and the estimate, say M_3, for the distribution of the unlabeled data. The unlabeled instance for which the difference between M_2 and M_3 is small will decrease the distribution discrepancy of the labeled and the unlabeled data. Along with this, a similarity matrix $\boldsymbol{M}_1$ of $n_u\times n_u$ is used, where matrix entry $(i,\ j)$ represents the similarity between the i-th and the j-th instances.

Let $\boldsymbol{P}^i$ be the posterior probabilities of the i-th instance belonging to different classes and p_c^i is the probability of the i-th instance belonging to class "c". This results in $\boldsymbol{P}^i=\left(p_1^i,p_2^i,...,p_{|c|}^i\right)$, where $|c|$ is the number of classes. Then the formulation for $\boldsymbol{M}_1$, M_2, and M_3

on the labeled and the unlabeled data respectively, is given by (12), (13), and (14) respectively.

$$\boldsymbol{M}_1(i,j) = \frac{1}{2}\exp\left(-\gamma \left\|\boldsymbol{P}^i - \boldsymbol{P}^j\right\|_2^2\right), \tag{12}$$

$$M_2(i) = \frac{n_l+1}{n}\sum_{j=1}^{n_l}\exp\left(-\gamma \left\|\boldsymbol{P}^i - \boldsymbol{P}^j\right\|_2^2\right), \tag{13}$$

$$M_3(i) = \frac{n_u-1}{n}\sum_{j=1}^{n_u}\exp\left(-\gamma \left\|\boldsymbol{P}^i - \boldsymbol{P}^j\right\|_2^2\right), \tag{14}$$

where γ is the kernel parameter, $\sum_{k=1}^{|c|} p_k^i = 1$ and $\sum_{k=1}^{|c|} p_k^j = 1$. The instance selection can be made diversely by including $\boldsymbol{M}_1$, with distribution discrepancy and the final formulation for the representative part is given by (15).

$$\min_{\sigma} \sigma^{\mathrm{T}}\boldsymbol{M}_1\sigma + \sigma^{\mathrm{T}}(M_2 - M_3)$$
$$\text{s.t. } \sigma^{\mathrm{T}}1^{n_u}, \sigma_i \in [0,1]. \tag{15}$$

Then, Bvs2B finds the data instances for which the classifier is most uncertain about its label. It finds the probabilities of the input instance for each class and compares the probabilities of the two most probable classes (16). A smaller value of the difference, $\mathfrak{d}^{(i)}$, indicates that the classifier is most uncertain about the label of this instance and hence, it becomes the candidate query instance according to the informative measure.

$$\mathfrak{d}^{(i)} = p_1^{(i)} - p_2^{(i)}, \tag{16}$$

where $p_1^{(i)}$ and $p_2^{(i)}$ are the first and the second highest class belongingness posterior probabilities of the i-th data instance respectively. The selected instance is close to the decision boundary. However, with the modified Bvs2B, the instance should not only be close to the separating boundary, but it should also be close to the support vectors of SVM, that forms the decision boundary. A similarity measure is used to compute the distance between every unlabeled instance $\boldsymbol{x}^{(i)}$ and support vector $\boldsymbol{SV}_j$, as given by (17).

$$f(\boldsymbol{x}^{(i)}, \boldsymbol{SV}_j) = \exp(||p^i - p_j||^2), \tag{17}$$

where p^i and p_j are the posterior probabilities of unlabeled instance $\boldsymbol{x}^{(i)}$ and support vector $\boldsymbol{SV}_j$, respectively. The obtained distance measure is then used to find the closest support vector $\boldsymbol{SV}_j$. This measure is combined with the calculated uncertainty measure to obtain the modified Bvs2B measure by (18):

$$\mathbb{C}(\boldsymbol{x}^{(i)}) = \mathfrak{d}^{(i)} \times f(\boldsymbol{x}^{(i)}, \boldsymbol{SV}_j). \tag{18}$$

Now, we combine this modified Bvs2B measure with the representative measure (12), (13), (14) to form an optimization problem given by (19).

$$\min_{\sigma} \sigma^{\mathrm{T}}\boldsymbol{M}_1\sigma + \sigma^{\mathrm{T}}(M_2 - M_3) + \pi\mathbb{C}$$
$$\text{s.t. } \sigma^{\mathrm{T}}1^{n_u}, \sigma_i \in [0,1]. \tag{19}$$

This objective function (19) is solved w.r.t. σ using quadratic programming; the largest value of σ is set to 1 and others to 0. The input instance corresponding to $\sigma = 1$, becomes the candidate query instance. The learner repeats this process until the terminating condition gets satisfied, or the learner runs out of labels.

Wang *et al.* in [61] also provided the instance selection strategy by combining the uncertainty and diversity criteria, for the multi-class setting. Here, the maximum margin was used as criteria for uncertainty and maximum mean discrepancy for diversity.

3) *Probabilistic AL (PAL)*. This query strategy is introduced by Krempl *et al.* in 2014 [62]. This scheme combines the idea of uncertainty sampling and error reduction with smoothness assumption [63]. According to this assumption, if two instances are closer to each other in the feature space, then their labels should also be close. An instance that affects the performance of classification the most is of interest.

In PAL, a probabilistic estimate is used for investigating the neighborhood information of the candidate instance as well as the overall gain in classification performance. To obtain the neighborhood information, label statistics, ls, is used, which computes the neighborhood statistics of input instance x. The statistics ls_x for particular instance x is a tuple of two attributes, i.e., $ls_x(n_n, \hat{p})$. The first attribute n_n is the total number of labeled instances in the neighborhood of x. The second attribute $\hat{p}$ is the posterior estimate, for the total number of the positive labeled neighbor set, as shown in Fig.9.

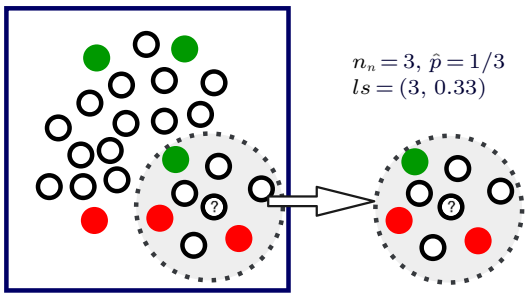


Fig.9. Representation for neighborhood label statistics for PAL query strategy (green and red circles are the instances belonging to positive and negative classes, respectively).

To calculate the expected gain in performance, it requires the knowledge of label y of input instance x, the posterior estimate of positive class within its neighborhood and ls. As the label is not known, then it is modeled as a random variable, and expectation E_y is computed over both positive and negative labels for the two-class classification problem. Here, the posterior is also not known: hence posterior is again modeled as random variable and expectation, E_p, over this random variable is again computed to determine the probabilistic gain, which is given by (20).

$$\text{pgain}(ls_x) = E_p\left[E_y\left[perf(D^L \cup (x,y))\right]\right], \quad (20)$$

where $perf(D^L \cup (x,y))$ is the performance of classification on the validation set as in EER, after incorporating instance x and label y in the training set. This gain is weighted by neighborhood density ρ_x, which is the ratio of the number of labeled and unlabeled instances in the neighborhood of x to the total number of labeled and unlabeled instances in the pool. The instance with the highest probabilistic gain is then selected as the candidate query instance, which is given by (21):

$$\hat{x} = \arg\max_{x \in D^U}(\rho_x \times \text{pgain}(ls_x)). \quad (21)$$

Krempl *et al.* further improved PAL to non-myopic PAL [64] called optimized probabilistic AL (OPAL). OPAL also considers the remaining label budget before querying the label of the unlabeled instance. The advantage of PAL is good classification performance, and its time complexity is comparable to that of uncertainty sampling.

• *Challenges of Informative and Representative Based AL Query Strategies.* This scheme provides a good set of candidate query instances. However, the problem is to combine these two mechanisms effectively, so that the trade-off between these two can be optimized.

3.1.4 Others

This subsection consists of query strategies that are hard to categorize as an informative, representative, or their combination. The query strategies under this categorization are given as follows.

1) *Expected Gradient Length* (*EGL*). Settles *et al.* introduced EGL for classification [65]. This query strategy focuses on those unlabeled input instances, which cause the greatest change in the parameters of the current hypothesis model. This technique of query is also referred to as "expected model change" and applicable wherever the gradient-based learning is used.

EGL starts by learning the hypothesis on the labeled data. The learner then selects an unlabeled input instance from the pool of data. The hypothesis is then again learned by adding the selected instance along with a chosen label into the labeled dataset. The learner looks for the change in the current parameter values of the model due to the added instance. The instance causing the maximum change in the parameter values is selected as a candidate query instance. This newly labeled instance is added to the training set, and this process is repeated until some preset criterion is achieved.

For binary classification, let $\bigtriangledown J(\Theta)$ be the gradient of the cost function with respect to the model parameters Θ. The cost function can be the squared difference between the actual and the predicted result. Let $\bigtriangledown J(\Theta)^+$ be the gradient obtained after mixing the selected unlabeled instance, $\boldsymbol{x}^{(i)} \in D^U$, when its label is chosen as "+". Similarly, let $\bigtriangledown J(\Theta)^-$ be the gradient obtained after mixing the selected unlabeled instance, $\boldsymbol{x}^{(i)} \in D^U$, when its label is chosen as "–". As the label of the instance is not known, the expectation is computed over these two gradients to form the EGL, as given by (22).

$$\begin{aligned} &EGL(\boldsymbol{x}^{(i)}) \\ &= p_i \times \bigtriangledown(J(\Theta)^+) + (1-p_i) \times \bigtriangledown(J(\Theta)^-), \end{aligned} \quad (22)$$

where p_i is the probability of obtaining the "+" label for the unlabeled instance $\boldsymbol{x}^{(i)}$. The learner chooses that unlabeled instance for the query, which has the maximum value for $EGL(\boldsymbol{x}^{(i)})$. This method is quite accurate, but it requires to train the model for every unlabeled input instance to identify the candidate query instance. Therefore, it is very computationally expensive for those datasets, having large feature vectors and a large number of class labels. This method is also affected greatly by outliers.

2) *Expected Error Reduction* (*EER*). Roy and McLallum in 2001, proposed this query strategy for text classification [66]. The learner looks for those instances to query, which minimizes the generalization error the most. A validation set, v_l, is required to evaluate the performance of the learned hypothesis. v_l is obtained after removing one instance randomly from the set of unlabeled instances and then added to the training set to check its validity for the candidate instance.

The initial hypothesis is learned on the available labeled data. Then the learner chooses an instance, x,

from the unlabeled data along with a label ($y \in Y$) and adds them to the training set. Then, the hypothesis model is retrained on the updated training set. The performance of the updated hypothesis is evaluated on the validation set using (23), and this process is repeated over all the possibilities of class labels. The resulted expected loss for each possible label y is weighted by the posterior probability obtained from the current classifier to get the average expected error.

$$\hat{E}_{\hat{P}_{D^L}} = \frac{1}{|v_l|} \sum_{x \in v_l} \sum_{y \in Y} \hat{P}_{D^{L*}}(y/x) \log \hat{P}_{D^{L*}}(y/x). \quad (23)$$

Here, the posterior $\hat{P}_{D^{L*}}(y/x)$ is learned on $D^{L*} = D^L + (x, y)$, i.e., after adding the instance x with label y in the training set, with the log loss as an estimate of error. Now, the remaining instances are evaluated based on expected error criteria to identify the candidate query instance. The candidate query instance is obtained corresponding to the minimum expected error. This approach has also been used for other applications like security and nonparametric model training [67]. Major drawbacks of this strategy are: it is very computationally expensive, the expectation depends on the good posterior estimate, and also it needs a validation set for checking the performance.

3) *Reinforcement Learning Based Approach.* Reinforcement learning (RL) based approach replaces the handcrafted query strategy criteria with machine learned criteria. Fang *et al.* used the policy learned by the RL agent to identify the query instances [68]. This study also shows the correspondence between AL and RL tasks and results in the formulation of AL as the decision process. They presented this study for name-entity relationship (NER) task of natural language text processing. A cross-lingual embedding is used to learn the RL agent on high resource languages and then utilize the learned agent on low resource languages. This work was done for the stream-based scenario.

A solution for the pool-based setting is provided by Liu *et al.* in the form of imitation learning [69]. It requires an expert that can take the perfect action at a particular AL situation, and then a policy network is trained to imitate the taken action [70]. Pang *et al.* [70] used deep RL not only to learn the proper criteria for instance selection but also to achieve the generalization, i.e., the agent can be generalized across the different datasets. They designed the AL task as a meta-learning problem and used the deep neural network (DNN) to generate the embedding of datasets of the different domains in the common space. This enables the learned policy to be generalized across different datasets of different feature space dimensions. Bachman *et al.* also presented a meta-learning approach for building the AL query strategy [71]. The interaction of the RL agent with many related tasks forms the query strategy, which is further utilized for the instance selection on other related tasks that lack prior data.

3.2 Pool-Based Query Strategies for Regression

The work done in this area of AL is comparatively less than the previous ones. The query strategies for regression are given further in this subsection.

3.2.1 Variance Reduction

The first statistical analysis of AL for regression was done by Cohn *et al.* in 1994, for studying the dynamics of robot arm problem [72]. The variance reduction chooses instances that minimize the output variance, along with the reduction in the generalization error. Geman *et al.* in 1992 gave an analysis of total expected generalization error [73], which is given by (24).

$$E_{\text{out}}(h^{(S^{(i)})}) = E_x[(h^{(S^{(i)})}(x) - h^*(x))^2], \quad (24)$$

where $h^{(S^{(i)})}$ is the hypothesis learned on the sample $S^{(i)} \in \{S^{(1)}, S^{(2)}, ..., S^{(k)}\}$ and all these $S^{(i)}$s are sampled from the input labeled data. Let h^* be the true hypothesis and E_{out} be the out of sample error, which measures the performance of learned hypothesis on unseen data instances. Let $\left\{h^{(S^{(1)})}, h^{(S^{(2)})}, ..., h^{(S^{(k)})}\right\}$ be the set of hypotheses learned on samples $\left\{S^{(1)}, S^{(2)}, ..., S^{(k)}\right\}$, respectively. Then the average approximation of learned hypothesis is given by (25) [73].

$$\bar{h} \approx \frac{1}{k} \sum_{K=1}^{k} h^{(S^{(K)})}. \quad (25)$$

Now let S be the total number of samples in the dataset, then the expectation on the generalization error for all samples is given as [73]:

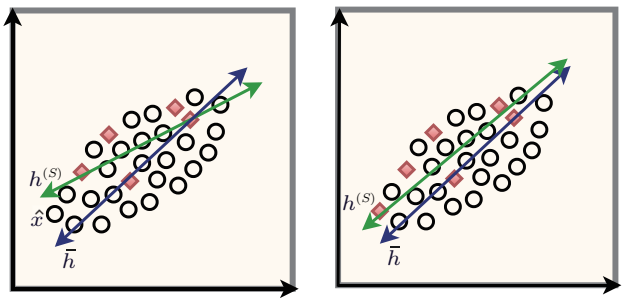
$$E_S[E_{\text{out}}(h^{(S)})] = E_S\left[E_x[((h^{(S)}(x) - h^*(x))^2]\right]. \quad (26)$$

After some manipulation, (26) can be written as a nice combination of two terms, as given by (27) and (28) [73]:

$$E_S[E_{\text{out}}(h^{(S)})] = E_x[E_S[((h^{(S)}(x) - \bar{h}(x))^2] + (\bar{h}(x) - h^*(x))^2], \quad (27)$$

$$E_D[E_{\text{out}}(h^{(S')})] = E_x[Variance(x) + bias(x)]. \quad (28)$$

The above analysis inferred that *Variance* is the only term which minimizes the error for a given model class, as the bias for a model (*bias*) is fixed. Therefore, the minimization of total expected error is to minimize the variance, as shown in Fig.10.



(a) (b)

Fig.10. AL query strategy for regression based on variance reduction. (a) Initial hypothesis learned on a small sample (green line) and average hypothesis (in blue). (b) Updation of the initial hypothesis after querying.

Fisher's information function was also utilized to minimize the error based on variance reduction [74]. The major problem with this approach is the dimension of data because this function requires a matrix of same dimension as that of data. Therefore, some dimensionality reduction methods are required for the proper utilization of this framework. Long *et al.* generalized the framework of variance reduction using Bayesian expected loss optimization to the classification under AL. This generalized method is known as expected loss optimization (ELO) [75].

3.2.2 QBC for Regression

Earlier QBC (query by committee) was utilized for the classification problem under AL, which is based on the maximum disagreement among committee members for the unlabeled instances. Later on, Freund *et al.* investigated that QBC is not only applicable to binary labels but also applicable to discrete labels [76]. Also, Krogh and Vedelsby formed a committee of a neural network for learning real-valued functions [77]. The variance in the output prediction of all committee members for the input instance $\boldsymbol{x}^{(i)}$, is called the ambiguity of that instance. This measure provides the estimate for generalization error of the committee.

The learner chooses the unlabeled instance for the query which has maximum ambiguity. This approach minimizes the variance over input space by querying the unlabeled instance with maximum variance over the committee members [78].

3.2.3 Maximizing Expected Model Change (MEMC)

This learning framework for regression under AL was proposed by Cai *et al.* in 2013 [79] and also known as expected model change maximization (EMCM). The learner chooses the unlabeled instance, which causes the maximum change in the current model parameter. This method is inspired by the stochastic gradient [80], where the model is updated after learning on each input instance instead of updating the model on the whole dataset. The model change is estimated by the difference between current model parameters and the parameters obtained after learning the model on the updated training set. A similar idea of maximum model change was studied previously by Settles *et al.* in the classification for AL [65]. The formulation for EMCM is given by (29).

$$\mathscr{C}(\boldsymbol{x}^{+}) = \Omega \frac{\partial \ell_{\boldsymbol{x}^{+}}(\theta)}{\partial \theta}^{[65]}, \tag{29}$$

where $\mathscr{C}(\boldsymbol{x}^{+})$ is the change in model parameters with learning rate Ω and $\ell_{\boldsymbol{x}^{+}}$ is the loss function obtained after incorporating a new instance $\boldsymbol{x}^{(i)}$ in the labeled data. The instance contributing maximum change in the model parameters is selected as candidate query instance, $\hat{x}$, as given by (30).

$$\hat{x} = \arg \max_{\boldsymbol{x}^{(i)} \in D^{U}} ||\mathscr{C}(\boldsymbol{x}^{(i)})||^{[65]}. \tag{30}$$

Cai *et al.* further used this EMCM model along with SVM for AL [81]. However, this model directly minimizes the target cost function to obtain the candidate query instance but at the cost of high computational effort.

3.3 Pool-Based Query Strategies for Clustering

The AL philosophy of intelligently querying the data instances can be merged with conventional clustering algorithms, to improve the clustering quality. This process of learning takes the unlabeled instances as input and returns the fully labeled data instances [82]. The strategies under this framework are the followings.

3.3.1 Hierarchical Sampling

The first approach for clustering under AL was presented by Dasgupta and Hsu in 2008 [83]. The labeling strategy works with the assumption that there exist nice clusters with pure labels, and the instances of data clusters belong to two classes only. The rules followed by their algorithm for querying the unlabeled instance are as follows.

Rule 1. The learner first specifies the chosen cluster, C, and then selects input instances informally from C. The labels of selected instances are then queried to oracle.

Rule 2. Let cluster C be further divided into subsets, say C', such that all subsets $C'' \in C'$ are the subsets of set C also.

Rule 3. By this splitting, a nested structure of clusters so formed, allows the reuse of queried labels.

Fig.11 shows the learning framework for hierarchical sampling. The learner starts with a first cluster of the whole dataset, containing all unlabeled instances. This single cluster node serves as the root node of the hierarchical cluster tree. The learner then randomly chooses some unlabeled input instances from the cluster and queries their labels from the oracle. Then, based on the obtained label statistics, the learner decides whether to split the node or not. If the percentage of positive and negative instances in a node is higher than some threshold, then the learner splits the node to obtain a more refined cluster. This process is repeated further for every child node to grow the tree downwards until the learner gets pure clusters, i.e., a cluster having all instances of the same type or some stopping criterion is reached. In case of stopping criteria, if a slightly impure cluster is obtained, then the learner assigns the majority label to all the instances [82].

3.3.2 *Probabilistic Lipschitzness Based Clustering (PLAL)*

The learning procedure of this technique is similar to the hierarchical sampling. The assumption here is that the unlabeled data must satisfy "probabilistic lipschitzness" (PL) notation of clustering. According to this assumption, the closer instances are likely to have same labels, i.e., the denser area should be labeled homogeneously and the class boundary should go through low-density region [84]. The PL assumption is better than the existence of a hierarchical cluster tree assumption.

PL is a measure of the extent, to which, similar instances tend to have similar labels, which is used for grouping the unlabeled data instances. An earlier version of PL was introduced by Steinwart and Scovel in 2007 [85]. Urner *et al.* in 2011 suggested the use of PL for AL [86]. PL implies that the data can be divided into homogeneous clusters, separated by low-density regions.

The standard Lipschitzness is defined as a function $f : \mathbb{R}^n \to \mathbb{R}^m$, which will be locally Lipschitz at a point $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, if there exists,

- a neighborhood $B(\boldsymbol{x}^{(0)}, r)$ with $r > 0$ and
- $\Lambda > 0$,

such that inequality in (31) follows.

$$|f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(j)})| \leqslant \Lambda |\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}|, \tag{31}$$

for all $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)} \in B(\boldsymbol{x}^{(0)}, r)$, where $B(\boldsymbol{x}^{(0)}, r)$ is the open ball around $\boldsymbol{x}^{(0)}$ of radius r and f is the labeling function. This condition works well for data, where the instances of different classes are well separated. However, if this is not the case, then this condition is violated, and it requires some relaxation on this condition. The relaxed formulation is known as PL. A function, $f : X \to \{0, 1\}$, is said to be PL w.r.t. a probability distribution Pr over X, if inequality (32) holds.

$$Pr_{\boldsymbol{x}^{(i)}}(\exists \boldsymbol{x}^{(j)} | f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(j)})| \geqslant \frac{1}{\Lambda} |\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}|) \leqslant \phi(\Lambda), \forall \Lambda > 0, \tag{32}$$

where $\phi : \mathbb{R}^+ \to [0, 1]$ is a monotonically increasing function.

The clustering procedure takes unlabeled data and spatial tree as input. The spatial tree is a binary tree

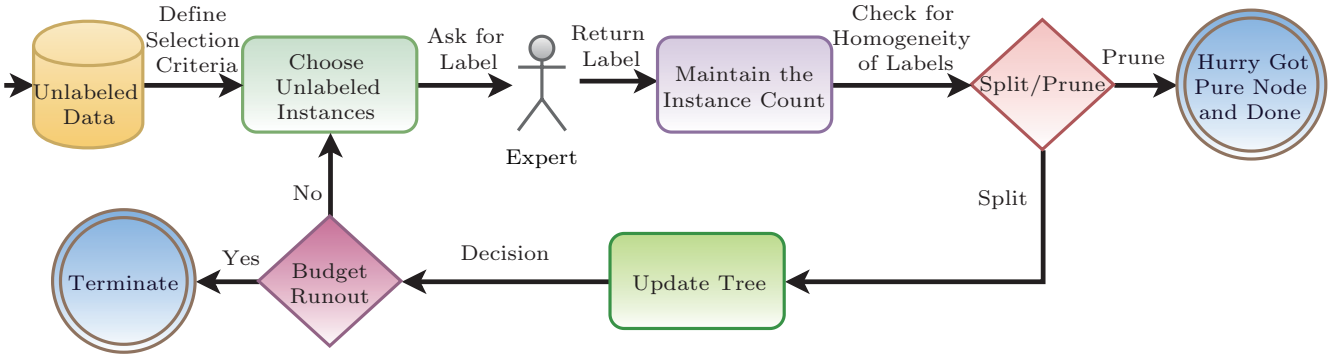


Fig.11. Framework of AL for active clustering based on hierarchical sampling.

that iteratively divides the input space among the nodes of the tree, and each node of the tree is called a cell [87].

Initially, the whole input space is considered as the root node of the spatial tree, which is an active node as it contains whole unlabeled data. The learner queries sufficient number of labels from the active node and checks for homogeneity or heterogeneity of nodes. A homogeneous node obtains the same label for all queried instances, and remaining instances also get the returned label. The homogeneous node is also called an inactive node and opposite to it, the heterogeneous node is added to active nodes. The process of splitting and querying is repeated for active nodes in the next iteration. This procedure ends with the label of all unlabeled instances [88].

3.3.3 AL Through Density Clustering (ALEC)

Wang *et al.* proposed ALEC in 2017 [89]. This method starts with the unlabeled dataset and exploits its structure for selecting the most representative instances to query. ALEC answers the questions: how to initialize the dataset, how to do clustering, how to select the critical instances for the query, which instance to classify and what to do with the remaining instance if any. The main steps of ALEC are as follows.

Step 1. *Building the Master Tree.* The master tree represents the relationship between the nodes and also provides the information about the growth of clusters in the tree. Building this master tree requires local density ρ_i and minimal distance ω_i functions [90]. The local density at the i-th input instance is equal to the number of the data points, whose distance from the i-th instance is less than or equal to some constant d_c and is given by (33).

$$\rho_i = \sum_j \chi(d_{ij} - d_c), \tag{33}$$

where $\chi(x) = 1$ if $x < 0$; otherwise $\chi(x) = 0$. The parameter d_{ij} is the distance between the two data points i, j and d_c is the cutoff distance.

The minimal distance ω_i is measured as the smallest distance between the data point i, and the data point j having a local density higher than i, and is given by (34).

$$\omega_i = \min_{j:\rho_j > \rho_i} (d_{ij}). \tag{34}$$

These parameters are used to determine the master or parent of a node or data point to build the tree. The master of a node is its nearest neighbor having a higher density, or a node x_j will be the master of node x_i, if and only if, $\rho_j > \rho_i$ and $d_{ij} < d_{il}$, for any other node l. In this manner, the master of every node is determined, and a master tree is constructed. The node with a high density does not have any master and becomes the root node of the master tree.

Step 2. *Clustering.* The clusters are formed from the master tree. For efficient clustering, Wang *et al.* used the CFSFDP algorithm [89], which was proposed by Rodriguez and Laio in 2014 [90]. CFSFDP stands for clustering by fast search and finds out density peaks. The data point with a high value of distance (ω_i) and density (ρ_i) is chosen as the cluster center. Here, parameter ξ_i is a combined term for (ω_i) and (ρ_i), which is used to determine the cluster center and is given by (35).

$$\xi_i = \omega_i \times \rho_i. \tag{35}$$

The data point having the highest value of ξ_i becomes the cluster center. Initially, two clusters are formed by selecting their centers with the help of parameter ξ_i.

Step 3. *Selection of Instance.* After forming the clusters, the query instances are selected from the respective clusters according to the value of ξ_i. In the ALEC algorithm, $\sqrt{n_u}$ top instances with a large value of ξ_i are selected for the query from the total n_u unlabeled instances of the cluster.

Step 4. *Classification.* If the labels of queried instances are homogeneous, then this label is assigned to all other remaining instances of that cluster; otherwise splitting of the cluster is required. This process of querying and splitting is repeated to grow the cluster tree until the process runs out labels or some stopping criteria are used. If there still exist some more impure or unlabeled instances, then standard voting strategies are used to assign the label to these instances.

4 AL in Realistic Environment

AL plays a vital role in many areas of classification, regression, and clustering, by reducing the labeling and learning cost. However, various AL query strategies work under certain assumptions, which may not always be feasible in reality. Therefore, to motivate the researchers to work further in this area, some of the challenges that AL faces in practice, are discussed here along with some solution proposed in the literature.

1) *Noisy Oracle.* The AL setup is based on identifying the candidate query instance and querying the label of that instance to oracle. Here the assumption is that the oracle is always faithful. However, the oracle

might not always be perfect in practice, and this may lead to the failure of the whole AL setup.

Yan *et al.* proposed a solution by combining the multiple imperfect oracles [91]. However, this scheme introduces an overhead of querying the instances to multiple oracles and also the problem of identifying the most precise result. Fang *et al.* proposed another solution to this problem, known as self-taught AL [92]. In this, the oracle learns complementary knowledge from the other oracles for achieving a better result. Shu *et al.* also proposed a solution by describing a technique called "learning with self-healing" [93]. In this, the learned model is applied to the training dataset formed by querying, and the highly uncertain instance is again queried to improve the quality of the dataset. A recent solution to this problem is provided without crowd-sourcing. This strategy works in two steps. First of all, an input instance is identified that enormously influences the learned model. An instance (say x) is said to be highly influential if the model (say h) is trained using that instance along with its label (say y), and significantly disagrees with h on labeling other instances. The second step is to identify those input instances, which are highly influenced due to the change of model in the previous step. An instance is said to be highly influenced, if the committee of the learned model agrees on the common label for x but disagrees with $h(x)$. The first step is used for selecting the influential instance, and the second step is used for eliminating the noisy or influenced instances [94].

2) *Class Imbalance.* Initially, the model is learned from the sampled data by assuming that the instances are drawn equally from each of the class. However, this is not always the case. Instances may belong to one class only, and this may be because of the class skewness. Ertekin *et al.* tried to solve this issue by under-sampling the bigger class and oversampling of the smaller class [95, 96].

3) *Changing Model Class.* Suppose the learner initially learns a low complexity model from a small number of labeled instances. However, after observing the inappropriate error performance, there may be a need for switching to models of a higher degree. In this case, the candidate query instances of the previous model may not be the candidate for the newer model. This problem leads to relearning the model by selecting the new set of candidate query instances. The problem of changing model class arises because the instances which are informative for one class model may not be informative for another class model [97, 98].

4) *Cost-Sensitive AL (CSAL).* Another assumption considered by most of the AL strategies is that the cost of labeling all the instances is equal. However, this may not be the case always, as the labeling cost may vary from instance to instance. Also, misclassification cost may change for different instances. Hence, minimizing the number of query instances may not correspond to minimizing the cost required [99]. The two types of cost minimization involved in CSAL are as follows.

- *By Minimizing the Labeling Cost.* The cost of labeling may not be uniform for all the instances. As, in the name entity relationship task of natural language processing (NLP), Tomanek *et al.* [100] used the annotation time of an instance as a cost measure. This study transformed the utility of the instance selection, and its labeling cost into a rank. A higher utility claims a higher rank, and a lower cost claims a higher rank. Then the linear combination or ratio of these two was used as a measure of the total cost for instance selection [100]. Settles *et al.* investigated variable labeling cost and provided an empirical study for annotation cost in real setting [99]. This study tells that it is tough to predict the cost of labeling as the time required in labeling depends upon the proficiency of the annotator, which changes from one person to another.

For hyper-spectral data, labeling the spatially distributed data is also cost-sensitive. In this case, labeling cost involves traveling to a particular location and then performs some tests. Liu *et al.* used random or uncertainty sampling to find the candidate query instance, and then traveling salesman solution was used to follow the chosen point [101]. Persello *et al.* investigated that the annotation of an instance in spatially distributed data also depends on the previously labeled instances. Therefore, this problem was addressed as a Markov decision process, and the next candidate instance is identified based on the long-term cumulative reward [102].

- *By Minimizing the Misclassification Cost.* In the case of intrusion detection system, the cost of mislabeling is much higher than any other normal system. Margineantu proposed a solution to minimize the combined cost of instance selection and misclassification decisions [103]. The solution assumes that associated misclassification loss with the decision is represented by a $c \times c$ matrix, where c is the number of classes. This cost matrix is combined with the instance selection method to choose the target candidate instances.

Krishnamurthy *et al.* [104] investigated the problem of cost-sensitive for multi-class classification. In this

setting, CSAL queries for the cost of a subset of labels for input instance as it is expensive to query the cost for all the labels. Here, the cost refers to the prediction cost, not the labeling cost. This study assumes that the learner has access to a set of the regression function, which is used to predict the range of possible cost that a label may take. The learner queries the cost for that label, for which the instance has the largest cost range, and its estimated minimum cost is smaller than the smallest maximum cost [104].

5) *Multi-Instance AL.* Settles *et al.* formulated the problem of multi-instance AL [65]. In this problem, rather than assigning the label to the individual instance, the label is assigned to a group of instances. This group of instances is considered as a bag, which is assigned positive label if at least one of its instances is positive, and negative label if all of its instances are negative. This method applies to the text, images, and video classification, where the instances are formed by the small portion of these objects [105].

In recent work, Wang *et al.* utilized the diversity and the informative measure for instance selection [106]. For diversity, clustering-based methods and the fuzzy rough set were utilized.

6) *Multi-Label AL.* Another assumption in AL classification problem is that each instance belongs only to one class. However, in practice, there may be some cases where input instances can belong to more than one class at a time, for example an image may contain multiple labels. This scenario leads to the assignment of various labels to a single instance, and hence the difficulty of selecting candidate query instance in AL. In literature, this type of problems is solved in two ways.

a) *Algorithm Adaptation.* Wu *et al.* proposed a solution by extending the traditional AL method to multi-label AL [107]. Yang *et al.* proposed a method for predicting the multiple labels of a single instance by summing the expected losses for all the labels [108]. Similarly, classifier prediction is used for inconsistency and rank aggregation is used to find the score across all the labels [109]. Hence, this process involves two steps: i) an evaluating function is used to score the instance-label pair, and ii) then an aggregating function is used to aggregate the score [110].

b) *Problem Transformation.* In this approach, the multi-label learning problem is transformed into a single label problem, and then a single label learning approach can be used for instance selection. Compressed sensing [111] was used to transform the label vector into lower dimension space using random projection matrix. Then the sample vector and the projected vector were used for learning, i.e., the hypothesis is learned from the sample vector to the projected vector. For the test sample, first, the label for the test sample is predicted, and then reconstruction is used for mapping back the projected vector to the label set [112].

Along with this, label reduction is made, when the number of labels is large. This is done by combining the labels after finding the correlation between them and the reduced labels are used for learning [113]. A genetic algorithm based solution is used to optimize the multi-objective function, which includes informativeness, representativeness, and diversity. This multi-objective function is used for the selection of instances in batch mode learning [114].

7) *Multi-Task AL.* Generally, all AL strategies are involved in learning only a single task at a time. However, in practice, the learner may be engaged in learning multiple tasks using a single instance. In this case, a single instance is labeled simultaneously for all the subtasks. Reichart *et al.* worked out multi-task AL for the case of two tasks [115]. Zhang also tried to solve this problem by evaluating the score of all tasks for each instance, and the candidate query instance is selected based on the combined effect of the calculated score [116].

The major challenge in this area is to identify the input instances that are most beneficial for jointly learning the task. For this, identifying the relatedness of the tasks is an essential job, as the related tasks may have closeness in the parameters and priors. Hierarchical relation among the tasks helps in selecting the related tasks for better results [117].

8) *Transfer Learning with AL.* Transfer learning works, when one task (source task) has sufficient training data, and another task (target task) has limited data. These two tasks should be similar but not identical. For this, one solution is provided by Gavves *et al.* [118] by combining the AL and zero shot learning [94]. Zero shot learning uses the relation between the source and the target for predicting the label distribution for unlabeled target data without using the target annotated data. The obtained results act as a prior for the AL algorithm. Now, this prior information and SVM-based query strategy framework is used for instance selection. Besides this, Wang *et al.* proposed a solution, which allows the transfer if there are smooth changes between the conditional and the marginal distribution of the source and the target data [119].

9) *AL in Batches.* Generally, each AL task selects one query instance at a time and asks its label. However, some time instance selection process is slow, and also multiple annotators may be available. Then in these cases, to speed up the annotation process and to fully utilize the resources, multiple instances are selected for annotation. Informative and representative criteria are used to select the instances in batch. This process starts by ranking the informative instances and selects high-ranked instances for clustering. Then, these clustering centroids are used as batch [47]. Gou *et al.* formulated the instance selection problem as an optimization problem, which combines classifier discriminating performance and uncertainty to form the objective function [120]. Chakraborty *et al.* provided another solution, which makes batch size adaptive. In this approach, the criterion for batch size is combined with instance selection criteria for batch instance selection [121].

10) *AL in a Distributed Environment.* Generally all work in AL is done for the centralized environment, where all the data is available on one node, and the whole processing is done on the same node. However, this is not always the case in a realistic scenario, as in distributed applications, data is spread over geographical locations connected by the network. Shen *et al.* in [122] provided a solution for this in two steps: a) distributed sample selection strategy helps the nodes to cooperatively select the data based on informativeness, diversity and representativeness; b) distributed classification algorithm helps each node to train their local classifiers in the global sense.

11) *AL with Deep Learning.* Deep learning (DL) was introduced by Hinton *et al.* [123], which replaced the manual feature extraction with machine-learned features. Wang *et al.* combined the convolutional neural network (CNN) — an architecture of DL — and AL for image classification [124]. However, the direct integration of CNN and AL has some difficulties:

a) CNN requires a large amount of labeled data for training, but a small amount of labeled data is available in AL scenario.;

b) In CNN, feature learning and classifier training both are jointly optimized, thereby AL may face the divergence problem.

The first issue has been addressed by the complementary instance selection technique. In this technique, CNN is fed with unlabeled instances, and then according to the output of CNN classifier, the parameters of CNN are tuned with two kinds of instances. The instances with least prediction confidence, i.e., the most uncertain instances, come under the first kind of instances, and the instances with high prediction confidence, i.e., most certain instances, come under the second kind. Also, the instances with the least confidence are the minority instances because they are few, and the instances with high confidence are the majority instances. Minority instances are queried from the expert and majority instances are pseudo-labeled according to the output of CNN. Now, these queried and pseudo-labeled instances are used to fine-tune the CNN. Therefore, this solution resolves the problem of small labeled data. The second problem is resolved by incremental learning technique, i.e., selecting unlabeled instances in an easy-to-hard manner for pseudo-labeling. Initially the learned model is quite vague, thereby the pseudo label is assigned only to highly certain instances by putting a high threshold to the confidence level. With an increase in the performance of the classifier, this threshold is decreased accordingly.

Rahhal *et al.* used deep active learning (DAL) for electrocardiogram (ECG) signals classification [125]. Here also, DL is used for feature extraction, and then AL is used to find out the most informed/uncertain instances for the query. Also, Zhou *et al.* used DL and AL combination for sentiment analysis [126]. The combination of DL and AL outperforms the state-of-the-art methods with less label complexity.

5 Analysis of AL Strategies

This section analyzes the effectiveness of AL query strategies in reducing the sample complexity. Analysis of AL query strategies for classification and clustering is done in subsequent subsections.

5.1 Analysis of AL Strategies for Classification

Analysis of AL query strategies is done for parametric and nonparametric classification settings. The parametric setting starts with the assumption of nature of hypothesis learning, while in nonparametric setting relaxing this assumption.

5.1.1 *Analysis of AL Strategies for Classification in Parametric Setting*

A mathematical study of sample complexity shows the number of input instances needed to train the classifier with some generalization error. Probabilistic approximate correct (PAC) was proposed by Valiant in

1984, to analyze the supervised learning [127]. Considering the case of perfect classification, it allows learning the hypothesis having a low generalization error ϵ, with a high probability of $(1-\delta)$. The formulation for PAC is given by (36) [127].

$$P((E_{\text{out}}(h_t) - E_{\text{in}}(h_t)) \leqslant \epsilon) \geqslant 1-\delta, \tag{36}$$

where E_{out} and E_{in} are the out-of-sample and in-sample error of the hypothesis $h_t \in H$, learned with the input sample of size t, respectively. In the setting of finite hypothesis space $|H|$ and for the case of perfect classification, the sample complexity, m, of supervised learning is given by (37) [127].

$$m \geqslant \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta)). \tag{37}$$

[1] gives more details of the PAC framework, and Vapnik presented a modified version of PAC in 1998 [31], as given by (38).

$$m \geqslant O\left(\frac{1}{\epsilon}\left(\hat{d}\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right)\right), \tag{38}$$

where $\hat{d}$ is the VC dimension that measures the complexity of the hypothesis space. This bound provides the flexibility in calculating m for different settings of ϵ and δ. (38) can be further approximated by (39) [31].

$$m \approx \tilde{O}\left(\frac{\hat{d}}{\epsilon}\right). \tag{39}$$

The above formulation applies to supervised learning. Two types of analyses for AL sample complexity bound are available: the first is based on disagreement coefficient, and the second is based on splitting index.

1) *Disagreement-Based Analysis.* For sample complexity bound of AL, Hanneke introduced disagreement coefficient [128] and then used disagreement region formed by "query by disagreement" (QBD) algorithm for analysis [7]. This algorithm is similar to QBC but is applicable in the streaming scenario. An example to illustrate the disagreement region is shown in Fig.12(a) [82]. Here the difference between two hypotheses, i.e., disagreement region, is shown by the region shaded in yellow and is given by (40) [128].

$$dis(h,h') = P(h(x) \neq h'(x)), \tag{40}$$

where $dis(h,h')$ gives the number of instances on which hypotheses h and h' disagree. Let $B(h^*,r)$ be the ball of radius r, representing the set of hypotheses h, which are within a difference of r from the true hypothesis h^*. (41) [128] gives the formulation for such a ball, and Fig.12(b) presents this ball with the true hypothesis in red.

$$B(h^*,r) = \{h \in H | dis(h,h') \leqslant r\}. \tag{41}$$

The region of disagreement for the version space $V \in H$ is the set of instances x, on which hypothesis $h \in V$ disagrees, and given by (42).

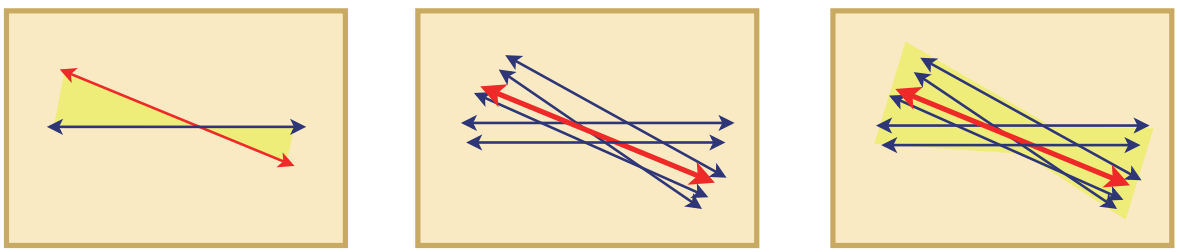
$$dis(V) = \left\{x \in D^U | \exists h,h' \in V : h(x) \neq h'(x)\right\}. \tag{42}$$

The yellow shaded region in Fig.12(c) shows the disagreement region for the true hypothesis. Hanneke introduced disagreement coefficient $\hat{\theta}$, using disagreement region, which is given by (43).

$$\hat{\theta} = \sup_{r>0} \frac{P[dis(B(h^*,r)B)]}{r}\ [128]. \tag{43}$$

This coefficient is used to characterize the sample complexity of AL and gives the information of variation of disagreement region with r. Now, this coefficient is used to determine the sample bound, with the assumptions of perfect classification and finite hypothesis space H, having VC dimension $\hat{d}$ [129], given by (44).

$$m_{QBD} \leqslant O\left(\hat{\theta}\left(\hat{d}\log\hat{\theta} + \log\frac{\log\frac{1}{\epsilon}}{\delta}\right)\log\frac{1}{\epsilon}\right). \tag{44}$$



(a) (b) (c)

Fig. 12. Illustration of region of disagreement. (a) Disagreement region (in yellow) between true hypothesis (in red) and learned hypothesis (in blue). (b) Set of hypotheses that are within disagreement region of radius r from the true hypothesis. (c) Region of disagreement (in yellow) for case (b).

A further approximation of (44) is given by (45) [129].

$$m_{QBD} \approx \tilde{O}\left(\hat{\theta}\hat{d}\log\frac{1}{\epsilon}\right). \tag{45}$$

The sample complexity bound in (45) shows an exponential improvement of AL over supervised learning. The number of instances required for training in AL scenarios is of the order of $\log\frac{1}{\epsilon}$ rather than of the order of $\frac{1}{\epsilon}$, required in the case of supervised learning. The further details of sample complexity bound can be found in [130]. The analysis presented above shows that AL is better than supervised learning.

2) *Splitting Index Based Analysis.* Along with the disagreement coefficient proposed by Hanneke [130] for analysis of AL query strategies, Dasgupta also presented the splitting index based AL query analysis with the binary class assumption [131]. Splitting index ψ captures the geometry of hypothesis space H and estimates the number of queries the active learner has to make to reduce the error of hypotheses below ϵ. The error between the hypotheses (say $h, h' \in H$) is measured as the distance (DT) and given by (46).

$$DT(h, h') = Pr_{x\sim D}(h(x) \neq h'(x)), \tag{46}$$

where D represents distribution over X. Now, the diameter ($diam$) of version space $V \subset H$ is given by (47) [131].

$$diam(V) = \max_{h,h'\in V} DT(h, h'). \tag{47}$$

$diam(V)$ represents the worst case error of hypotheses in V. The distance between h and h' forms an edge between them and the edge set is represented as, $e = \{\{h_1, h_2\}, \{h_1, h_3\}, ..., \{h_{n-1}, h_n\}\} \subset \binom{H}{2}$. Now, for instance x, the index ψ splits e, if labeling x splits the ψ fraction edges of e, which is given by (48).

$$\max\{|e^+|, |e^-|\} \leqslant (1 - \psi)|e|, \tag{48}$$

where $|e^+| = e \cap \binom{H_x^+}{2}$, represents the edge set of hypotheses that favor x with the "+" label, which is similar to $|e^-|$ also. Fig.13 demonstrates the ψ splitting of the hypotheses edges. To get the hypothesis with the error below ϵ, edges having a length higher than ϵ should be eliminated, as given by (49).

$$e_\epsilon = \{\{h, h'\} \in e : DT(h, h') \geqslant \epsilon\}. \tag{49}$$

Now, subset V is (ψ, ϵ, τ)-splittable with edges e, if (50) holds.

$$Pr_{x\sim D}(x, \psi - split, e_\epsilon) \geqslant \tau. \tag{50}$$

This represents that τ fraction of Pr is required for ψ-splitting of V and $\tau \leqslant \epsilon$. With this formulation, Dasgupta came up with the following results [131]:

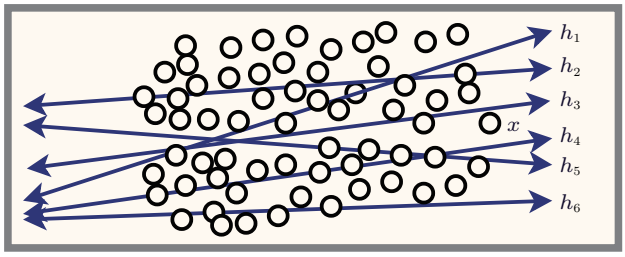


Fig.13. Demonstration for the ψ splitting with the assumptions that the blue lines $\{h_1, h_2, h_3, h_4, h_5, h_6\}$ represent hypotheses in H and the circles represent unlabeled instances $x \in D^U$. The edge set e is obtained as $e = \binom{H}{2}$. Also, the hypotheses $\{h_1, h_2, h_3\}$ map x to "+" and $\{h_4, h_5, h_6\}$ map x to "–", and then $H_x^+ = \{h_1, h_2, h_3\}$ and $H_x^- = \{h_4, h_5, h_6\}$.

1) any AL algorithm with $1/\tau$ unlabeled data must require to query at least $1/\psi$ labels;

2) H with VC-dimension $\hat{d}$, requires $\tilde{O}(\hat{d}/(\psi\tau)\log^2(1/\epsilon))$ unlabeled data and queries $\tilde{O}(\hat{d}/\psi\log^2(1/\epsilon))$ labels.

The resulted sample complexity is less than or equal to the sample complexity of passive learning $\left(\frac{\hat{d}}{\epsilon}\right)$, because ψ is always at least equal to ϵ.

Later on, Tosh and Dasgupta changed the worst case notation of the diameter to the average case by averaging the pairwise distance between the hypotheses of version space, instead of taking maximum [132]. This results in the new splitting index $\psi' = \frac{\psi}{\log(1/\epsilon)}$ and parallelly translates sample complexity.

5.1.2 *Analysis of AL Strategies for Classification in the Nonparametric Setting*

In parametric classification, an assumption is made about the hypothesis to be learned; generally, it is a linear separator. However, in the nonparametric classification setting, this assumption is relaxed, and the classifier proceeds with hierarchical space splitting with zoom-in into the regions that are not classified correctly. Nonparametric learning algorithms use regression function — say, $\hat{\eta} = E[Y/X = x]$ and $Y = \{0, 1\}$ — and plug-in into the Bayes classifier [133] as:

$$\hat{h}(x) = \begin{cases} 1, \text{ if } \hat{\eta}(x) > 1/2, \\ 0, \text{ otherwise.} \end{cases}$$

This nonparametric learning works with smoothness assumption, i.e., the regression function should satisfy Hölder's continuity assumption with parameter α and

also satisfy Tsybakov's noise condition with parameter β [133]. With $\alpha \leqslant 1$ and $\alpha\beta \leqslant d$, the nonparametric AL convergence rate is $m^{-\alpha(\beta+1)/(2\alpha+d-\alpha\beta)}$, in contrast to nonparametric passive learning rate, which is $m^{-\alpha(\beta+1)/(2\alpha+d)}$ [134]. Locatelli *et al.* further refined these results by considering interaction between the parameters α, β, d, and marginal distribution over X, i.e., (P_X) [135]. With $\alpha \geqslant 1$, $\beta \geqslant 0$, and uniform P_X, the convergence result of AL is changed to $m^{-\alpha(\beta+1)/(2\alpha+[d-\beta]_+)}$. In contrast to this, the convergence for passive learning remains the same. Also, for unrestricted P_X, AL convergence rate changes as same as passive learning convergence rate and the convergence rate of passive learning changes to $m^{-\alpha(\beta+1)/(2\alpha+d+\alpha\beta)}$.

5.2 Analysis of AL Strategies for Clustering

Earlier in Subsection 3.3.2, a discussion about PLAL is presented, which is clustering-based AL technique. Let $X = [0,1]^d$ be the input domain and the labeling function $h : \mathrm{X} \to \{0,1\}$ is ϕ-Lipschitz with some function ϕ. Let q_k be the bound on the number of queries made for each cell in hierarchical clustering at the k-th level. Also, let λ_k^D be the maximum data diameter of a cell at level k, which is formulated as: $\lambda_k^D = \max\{diam(C,D)\}$, where C denotes the cells or clusters at level k and D denotes the set of input instance. Data diameter of a cell is defined as: $diam(C,D) = \max\limits_{x,x' \in C \cap D} \| x - x' \|$. For $\lambda_i \in [0, \sqrt{d}]$ and $\lambda_k^D \leqslant \lambda_k$, the expected number of queries required in PLAL is bounded by $\min\limits_{k \in \mathbb{N}}(q_k 2^k + \phi(\lambda_k).n_u)$, where 2^k represents the upper bound on the number of cells at level k, and $\phi(\lambda_k).n_u$ represents the expected number of points that lies in heterogeneous clusters at level k.

6 Empirical Evaluation of AL Query Strategies

Several studies demonstrated the empirical evaluation of AL query strategies. A few details of these studies are presented in Table 2. The aim of most of the studies is to identify the best performing query strategy.

Schein *et al.* studied the performance of AL query strategies with the logistic regression classifier in the binary and multi-class setting [136]. This work compares the query by committee, uncertainty sampling, variance reduction, and their variations. The experiment was done with 10 datasets, out of which seven were obtained from the real-world domain, and three were artificially generated. With accuracy as a performance measure, the margin sampling: an uncertainty sampling variant, and QBB-MM [137] are the winning strategies. Yang *et al.* [138] extended this study with logistic regression for more comprehensive analysis. In that study, the comparison was made on three artificial sets and 44 real-world datasets, with six AL query strategies and their variations. From this extensive study, the conclusion was made that the uncertainty sampling variant with entropy is the most promising method and it does outperform the min-max view approach (QUIRE), variance reduction methods, and the maximum model change algorithm in their experiments.

Settles and Craven [41] did another study to find out the best query strategy on the sequence labeling task. Their work compared the uncertainty sampling, QBC along with their variations and expected gradient length and information density. The experiment was done with eight natural language processing datasets along with the performance score as $F1$. On these datasets, the overall performance of information density was observed as the best [41].

Most of the previous studies focused on one classifier and one performance measure. Ramirez-Loaiza *et al.* [139] performed a study using multiple classifiers and performance measures. This work compares the random sampling, uncertainty sampling, and QBC with the five performance measures: accuracy, precision, recall, area under the curve(AUC), and $F1$ score. This study was done with 20 synthetic datasets and 10 real-world binary classification datasets. The observations of this study are as: 1) classification model selection is as much important as devising AL query strategies; 2) uncertainty sampling and QBC outperform random sampling on most of the datasets, when the performance measure is accuracy; 3) generally improved accuracy is obtained at the expense of recall; 4) with AUC, random sampling is relatively competitive with the chosen AL query strategies [139].

Reyes *et al.* performed the recent study of comparing the AL query strategies [140]. This work statistically compares the four AL query strategies on 26 UCI datasets using Friedman nonparametric statistical test. This test investigates the overall ranking for query strategies using the performance on chosen datasets, as it is challenging to rank them by looking at their performance cures. This work compares margin sampling, least confident, entropy sampling, and random sampling. According to ranking, margin sampling was awarded as the best, and random sampling as the worst.

Table 2. Description of Different Experiments for Empirical Evaluation

Serial	Experiment	Databases Used	Performance Metric	Winning Strategy
1	Schein *et al.* [136]	Datasets from UCI repository [141], LetterDB [142], OptDigits [143], TIMIT [144], WebKB [145], and three artificial datasets	Accuracy	Margin sampling: an uncertainty sampling variant, QBB-MM [137]
2	Yang *et al.* [138]	UCI datasets [141], MNIST handwritten digit dataset [146], Newsgroups dataset [147], ImageNet database [148], three binary synthetic datasets	Accuracy, area under the learning curve (ALC)	Maximum Entropy: an uncertainty sampling variant [19]
3	Settles *et al.* [41]	CoNLL-03 [149], NLPBA [150], BioCreative [151], FlySlip [152], CORA [153], eSig+Reply corpus [154]	$F1$ score	Information density [41]
4	Ramirez-Loaiza *et al.* [139]	Six AL challenge datasets [155], letterdb [142], California housing price [156], UCI KDD archive [157]	Accuracy, precision, recall, AUC, and $F1$ score	
5	Reyes *et al.* [140]	26 UCI datasets [141]	F-Measure	Margin sampling: an uncertainty sampling variant [23]

From this discussion, it is concluded that the winning strategy is different under different circumstances. A comparative summary of all the pool-based AL query strategies is presented in Table 3. This comparison helps in selecting a query strategy according to the working environment.

7 Implementation of AL

The implementation of AL query strategies requires a framework, which includes the management of the labeled and unlabeled instances, the criteria for selecting the query instances, a labeler, and the model building. Many Python toolboxes are freely available to help the beginners in learning the AL query strategy framework. All these toolboxes are built with great focus on the modularity, adaptability, and extensibility. Some of them are discussed as follows.

1) AL in Python (ALiPy) facilitates the learner to implement, evaluate, and compare most of the AL query strategies (around 20) [158]. ALiPy provides nice support to the learner for customizing the existing query strategies and also for implementing the new query strategies. 2) Libact, another Python toolbox, also provides the AL query strategies implementation under pool-based settings [159]. Along with the query strategies implementation, libact also provides the facility of on the fly best query strategy selection using a meta-algorithm: active learning by learning. 3) modAL toolbox also provides the implementation of popular AL query strategies for classification and also for regression [182].

All these implementations are done on the top of scikit-learn [183] and also with the facility to integrate Keras models in their algorithm [184]. These toolboxes are available on the github with nice documentation for installation and use.

8 Applications of AL

AL query strategies reduce the labeling and the learning cost of the machine learning task. AL has a wide range of applications in all the areas, where unlabeled data is enormous and labeled data is scarce. This work tried to cover major application areas, and their description, as given in Table 4. This table shows that AL can be used in NLP, image processing, signal processing, and many more.

9 Challenges in AL

This article has discussed AL query strategies with base assumptions and also has discussed AL for a realistic scenario. This section discusses some challenges in developing query strategies and in using active learning for real tasks. These are as follows.

- How to develop informative and representative criteria, and combine them for better results?
- When to use which active learning, is context-sensitive, thereby the selection of AL query for a problem is a tough task.
- It is also hard to find which combination of AL query strategy and performance classifier to use, as it is task-specific.
- How to decide the criteria for instance selection in batch mode AL task?

Table 3. Comparative Summary of AL Query Strategies

Approach	Time Complexity	Pros	Cons
Uncertainty sampling [19, 21–23]	$\approx O(n_u)$	– Simple, fast and easy to implement – Can be used with the probabilistic model	– Greedy in nature – Redundant instance selection
QBC [1, 24, 26, 27]	$\approx O(C_s n_u)$, C_s is the size of hypothesis committee	– Work in both regression and classification – Easy to understand	– Myopic nature of error reduction – Maintain a set of hypothesis
EGL [65]	$\approx O(n_u n_l)$	– Work fine with gradient-based learning – Quite an intuitive strategy	– Computationally expensive – Requires proper scaling of features
SVM-based [28–30]	$\approx O(n_u)$ + SVM training time	– Fast version space reduction – Inherent benefits of SVM learning	– Require learning of SVM for each of unlabeled data – Require knowledge of version space for better work
EER [66, 67]	$\approx O(n_u n_l g)$, g gradient computation	– Optimize the objective of interest	– Computationally expensive
Density-based [17, 42]	Constant, if density precomputed	– Utilize the whole dataset – Use correlation among instances	– Clusters are not clearly defined – Slow convergence
Diversity-based [8, 42, 44–47]	$\approx O(n_l)$ + informative instances	– Remove redundant instances – Act as a good preprocessor	– Alone may not work well
Hashing-based [34–38, 40]	$\approx O(n_u^{\varrho})$, $\varrho \leqslant 1$	– Faster query instance selection	– An issue in dealing with large data dimensions
Exploitation graph structure [48–51]	$\approx O(n_u)$	– The structure among instances is considered – Increased prediction performance	– Require an appropriate network structure
QUIRE [54–56]	$\approx O(n_u)$	– Include features of both informative, representative strategies	– Data needs to meet semi-supervised assumption
ERIAL [57–60]	–	– Provide natural solution using quadratic programming	– Data distribution impacts performance
PAL [62, 64]	$\approx O(n_u)$	– Fast and stable performance – Can be used with streaming data	– The problem of skewed prior
Variance reduction [72]	$\approx O(n_u m_p)$, m_p is the number of model parameters	– Applicable for both regression and classification – Directly optimize the objective of interest – A natural extension to batch queries	– Computationally expensive, – Limited to pool and synthesis scenarios – Difficult to implement
MEMC [65, 79, 80]	$\approx O(n_u d m_n)$, d = feature's dimension, m_n = number of models	– Easy to understand – Low generalization error	– A model change may not always lead to good generalization – Computationally expensive
Hierarchical sampling [82, 83]	At most eight times of the optimal z-clustering	– Exploit cluster structure – Remove sampling bias – Better theoretical guarantees	– Require pure and fine clusters – Limited to the pool-based scenario
Probabilistic Lipschitzness (PL) [84, 85]	–	– Use a realistic assumption of PL – Provide performance guarantees	– PL assumption may not hold
AL through density clustering [89, 90]	$O(dn^2)$, d = feature dimension, n = number of instances	– Efficient and scalable – Deterministic instance selection	– Require the distance between instance pairs – Cutoff distance is difficult to determine

• How to handle the noisy oracle or instance, which degrades the performance by misleading the AL process?

• In cost-sensitive AL, how to combine the instance selection criteria with the instance labeling cost. It is also hard to formulate the labeling cost in a real scenario.

• Incorporating labeling/misclassification cost in the selection of batch of instances makes the learning task more complicated.

• Identifying the relation among the tasks and using it for aggregating the score for instance selection in the multi-task setting are challenging.

• Instance selection in the multi-label scenario is also challenging, as it also requires to combine the score for individual instance label pair for instance selection.

• In a multi-instance setting, assigning asymmetric +, − labels to the bag of instance makes learning

Table 4. Applications of AL Query Strategies

Area	Task	Description
NLP	Text categorization	To find out, which text belongs to which category based on the semantic contents of document [2, 14, 19, 30, 45, 98, 108]
	POS tagging	Part of speech tagging labels each word/token of natural language sentence with the appropriate tag, where the tag may correspond to the noun, verb, adjective, etc. [10, 15]
	Name entity recognition	This task takes the sequence of words of a sentence as input and returns the named entity (organization (org), location (loc), etc.) labeled word sequence as output [11, 41, 160]
	Information extraction	Information extraction is used to find/extract some valuable information from natural language document and use this information for building the well-defined template with some attributes, like in case of job post, and put the information in template [7, 13, 161]
	Machine translation	Parsing tree is generated using a natural language sentence, which can be used for translating that sentence into another language [28, 72]
	Sentiment analysis	Determine whether the given piece of text is saying positive or negative about some task [126, 162]
	Biomedical text mining	For identification of biomedical relationships such as genes-genes interaction or protein-protein interaction [163]
Image tasks	Remote sensing image	Multi-spectral and hyper-spectral image classification [3, 101, 102]
	Medical imaging	Classification of medical images [46, 164, 165]
	Image segmentation	Extract the object of interest and avoid homogeneous regions [166, 167]
	Image classification	Multi-label image classification [168, 169]
	3D reconstruction	3D scene is reconstructed with the given set of images from multiple poses [170]
	Object localization	Deep RL and AL is used to localize the object in a scene [171]
Other tasks	Indoor localization system	Localize the object based on the WiFi signal fingerprints [172]
	Video annotation	Identify the frames that a user should annotate [8, 173]
	Drug discovery	Selection of compounds that lead to the formation of target compound [174, 175]
	Clinical annotation	Annotate the clinical free text [176, 177]
	Intrusion detection system	Identify attack and compromises [178]
	Activity recognition	Involve activity detection and recognition, but a large number of activities in daily living motivate towards AL [179]
	Protein structure prediction	Deal with protein structure formation after protein-protein interaction [4, 180]
	Logged data classification	Logged data contains the bias of the logging strategy, which is overcome by AL query strategy technique and produces a generalized hypothesis [181]

process biased.

• There is less work in AL for regression and clustering. Selection criteria can be improved for better results in a realistic scenario.

10 Conclusions

AL performs nicely with the small amount of labeled data by fully utilizing the massive amount of unlabeled data through the query mechanism. It reduces the learning and annotation cost of data. This survey explained a hierarchical categorization of AL query strategies for classification, regression, and clustering in the pool-based scenario. The query strategies are presented with the base assumptions and also for the realistic scenarios. This study also discussed more advanced query strategies with reinforcement learning and deep learning approaches. This mathematical analysis examine the sample complexity analysis of AL and supervised learning. This analysis shows a logarithmic gain of AL over supervised in terms of sample complexity. An implementation guide has also been provided, along with major challenges and summary of AL application area.

References

[1] Mitchell T. Machine Learning (1st edition). MacGraw-Hill Education, 1997.

[2] Hu R. Active learning for text classification [Ph.D. Thesis]. Dublin Institute of Technology, 2011.

[3] Tuia D, Ratle F, Pacifici F, Kanevski M F, Emery W J. Active learning methods for remote sensing image classification. *IEEE Trans. Geoscience and Remote Sensing*, 2009, 47(7-2): 2218-2232.

[4] Guo J, Chen H, Sun Z, Lin Y. A novel method for protein secondary structure prediction using dual-layer SVM and profiles. *PROTEINS: Structure, Function, and Bioinformatics*, 2004, 54(4): 738-743.

[5] Zhu X. Semi-supervised learning literature survey. Technical Report, University of Wisconsin-Madison, 2008. http://pages.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf, Nov. 2019.

[6] Settles B. Active learning literature survey. Technical Report, University of Wisconsin-Madison, 2009. http://apophenia.wdfiles.com/local–files/start/settles_active_learning.pdf, Nov. 2019.

[7] Cohn D, Atlas L, Ladner R. Improving generalization with active learning. *Machine Learning*, 1994, 15(2): 201-221.

[8] Wang M, Hua X S. Active learning in multimedia annotation and retrieval: A survey. *ACM Trans. Intelligent Systems and Technology*, 2011, 2(2): Article No. 10.

[9] Lewis D D, Catlett J. Heterogeneous uncertainty sampling for supervised learning. In *Proc. the 11th Int. Conference on Machine Learning*, July 1994, pp.148-156.

[10] Zhu X, Zhang P, Lin X, Shi Y. Active learning from data streams. In *Proc. the 7th IEEE Int. Conference on Data Mining*, October 2007, pp.757-762.

[11] Zhu X, Zhang P, Lin X, Shi Y. Active learning from stream data using optimal weight classifier ensemble. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 2010, 40(6): 1607-1621.

[12] Zliobaite I, Bifet A, Pfahringer, Holmes G. Active learning with drifting streaming data. *IEEE Trans. Neural Networks and Learning Systems*, 2014, 25(1): 27-39.

[13] Wang P, Zhang P, Guo L. Mining multi-label data streams using ensemble-based active learning. In *Proc. the 12th SIAM International Conference on Data Mining*, April 2012, pp.1131-1140.

[14] Angluin D. Queries and concept learning. *Machine Learning*, 1988, 2(4): 319-342.

[15] Wang L, Hu X, Yuan B, Lu J. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 2015, 147: 426-434.

[16] Sun L L, Wang X Z. A survey on active learning strategy. In *Proc. the Int. Conference on Machine Learning and Cybernetics*, July 2010, pp.161-166.

[17] Fu Y, Zhu X, Li B. A survey on instance selection for active learning. *Knowledge and Information Systems*, 2012, 35(2): 249-283.

[18] Aggarwal C, Kong X, Gu Q, Han J, Yu P. Active learning: A survey. In *Data Classification: Algorithms and Applications,* Aggarwal C C (ed.), CRC Press, 2014, pp.571-605.

[19] Lewis D D, Gale W A. A sequential algorithm for training text classifiers. In *Proc. the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, July 1994, pp.3-12.

[20] Atlas L, Cohn D A, Ladner R E. Training connectionist networks with queries and selective sampling. In *Proc. the 3rd Annual Conference on Neural Information Processing Systems*, November 1989, pp.566-573.

[21] Culotta A, Mccallum A. Reducing labeling effort for structured prediction tasks. In *Proc. the 20th National Conference on Artificial Intelligence*, July 2005, pp.746-751.

[22] Shannon C E. A mathematical theory of communication. *Bell System Technical Journal*, 1948, 27(3): 379-423.

[23] Scheffer T, Decomain C, Wrobel S. Active hidden Markov models for information extraction. In *Proc. the 4th International Conference on Advances in Intelligent Data Analysis*, September 2001, pp.309-318.

[24] Seung H, Opper M, Sompolinsky H. Query by committee. In *Proc. the 5th Annual Conference on Computational Learning Theory*, July 1992, pp.287-294.

[25] Abe N, Mamitsuka H. Query learning strategies using boosting and bagging. In *Proc. the 15th International Conference on Machine Learning*, July 1998, pp.1-9.

[26] Melville P, Mooney R J. Diverse ensembles for active learning. In *Proc. the 21st Int. Conference on Machine learning*, July 2004, Article No. 56.

[27] Muslea I, Minton S, Knoblock C A. Selective sampling with redundant views. In *Proc. the 17th National Conference on Artificial Intelligence*, July 2000, pp.621-626.

[28] Cortes C, Vapnik V. Support-vector networks. *Machine Learning*, 1995, 20(3): 273-297.

[29] Kremer J, Pedersen K S, Igel C. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2014, 4(4): 313-326.

[30] Tong S, Koller D. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2002, 2: 45-66.

[31] Vapnik V. An overview of statistical learning theory. *IEEE Trans. Neural Networks*, 1999, 10(5): 988-999.

[32] Schohn G, Cohn D. Less is more: Active learning with support vector machines. In *Proc. the 17th Int. Conference on Machine Learning*, June 2000, pp.839-846.

[33] Campbell C, Cristianini N, Smola A. Query learning with large margin classifiers. In *Proc. the 17th Int. Conference on Machine Learning*, June 2000, pp.111-118.

[34] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. the 30th Annual ACM Symposium on Theory of Computing*, May 1998, pp.604-613.

[35] Gionis A, Indyk P, Motwani R. Similarity search in high dimension via hashing. In *Proc. the 25th Int. Conference on Very Large Data Bases*, September 1999, pp.518-529.

[36] Jain P, Vijayanarasimhan S, Grauman K. Hashing hyperplane queries to near points with applications to large-scale active learning. In *Proc. the 24th Annual Conference on Neural Information Processing Systems*, December 2010, pp.928-936.

[37] Vijayanarasimhan S, Jain P, Grauman K. Hashing hyperplane queries to near points with applications to large-scale active learning. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014, 36(2): 276-288.

[38] Basri R, Hassner T, Zelnik-Manor L. Approximate nearest subspace search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2011, 33(2): 266-278.

[39] Basri R, Hassner T, Zelnik-Manor L. Approximate nearest subspace search with applications to pattern recognition. In *Proc. the 2017 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2007.

[40] Wang J, Shen H, Song J, Ji J. Hashing for similarity search: A survey. arXiv:1408.2927, 2014. http://arxiv.org/abs/1408.2927, Nov. 2019.

[41] Settles B, Craven M. An analysis of active learning strategies for sequence labeling tasks. In *Proc. the 2008 Conference on Empirical Methods in Natural Language Processing*, October 2008, pp.1070-1079.

[42] Wu Y, Kozintsev I, Bouguet J Y, Dulong C. Sampling strategies for active learning in personal photo retrieval. In *Proc. the IEEE International Conference on Multimedia and Expo*, July 2006, pp.529-532.

[43] Ienco D, Bifet A, Zliobaite I *et al*. Clustering based active learning for evolving data streams. In *Proc. the 16th Int. Conference on Discovery Science*, October 2013, pp.79-93.

[44] Brinker K. Incorporating diversity in active learning with support vector machines. In *Proc. the 20th Int. Conference on Machine Learning*, August 2003, pp.59-66.

[45] Hoi S C H, Jin R, Lyu M R. Large-scale text categorization by batch mode active learning. In *Proc. the 15th Int. Conference on World Wide Web*, May 2006, pp.633-642.

[46] Hoi S C H, Jin R, Zhu J, Lyu M R. Batch mode active learning and its application to medical image classification. In *Proc. the 23rd Int. Conference on Machine Learning*, June 2006, pp.417-424.

[47] Xu Z, Akella R, Zhang Y. Incorporating diversity and density in active learning for relevance feedback. In *Proc. the 29th Eur. Conf. Inf. Retrieval Research*, April 2007, pp.246-257.

[48] Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. *AI Magazine*, 2008, 29(3): 93-106.

[49] Neville J, Jensen D. Iterative classification in relational data. In *Proc. the AAAI 2000 Workshop on Learning Statistical Models from Relational Data*, July 2000, pp.42-49.

[50] Richardson M, Domingos P. Markov logic networks. *Machine Learning*, 2006, 62(1/2): 107-136.

[51] Bilgic M, Mihalkova L, Getoor L. Active learning for networked data. In *Proc. the 27th Int. Conference on Machine Learning*, June 2010, pp.79-86.

[52] Wang Z, Ye J. Querying discriminative and representative samples for batch mode active learning. In *Proc. the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2013, pp.158-166.

[53] Nguyen H T, Smeulders A. Active learning using pre-clustering. In *Proc. the 21st Int. Conference on Machine Learning*, July 2004, Article No. 19.

[54] Huang S J, Jin R, Zhou Z H. Active learning by querying informative and representative examples. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014, 36(10): 1936-1949.

[55] Hoi S C, Jin R, Zhu J, Lyu M R. Semi-supervised SVM batch mode active learning for image retrieval. In *Proc. the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, Article No. 10.

[56] Belkin M, Niyogi P, Sindhwani V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 2006, 7: 2399-2434.

[57] Du B, Wang Z, Zhang L, Zhang L, Liu W, Shen J, Tao D. Exploring representativeness and informativeness for active learning. *IEEE Trans. Cybernetics*, 2017, 47(1): 14-26.

[58] Gretton A, Borgwardt K M , Rasch M J, Schölkopf B, Smola A. A kernel two-sample test. *Journal of Machine Learning Research*, 2012, 13: 723-773.

[59] Luo W, Schwing A, Urtasun R. Latent structured active learning. In *Proc. the 27th Annual Conference on Neural Information Processing Systems*, December 2013, pp.728-736.

[60] Anderson N, Hall P, Titterington D. Two-sample test statistics for measuring discrepancies between two multivariate probability density functions using kernel-based density estimates. *Journal of Multivariate Analysis*, 1994, 50(1): 41-54.

[61] Wang Z, Fang X, Tao X *et al*. Multi-class active learning by integrating uncertainty and diversity. *IEEE Access*, 2018, 6: 22794-22803.

[62] Krempl G, Kottke D, Spiliopoulou M. Probabilistic active learning: Towards combining versatility, optimality and efficiency. In *Proc. the 17th Int. Conference on Discovery Science*, October 2014, pp.168-179.

[63] Chapelle O, Schölkopf B, Zien A. Semi-Supervised Learning. The MIT Press, 2010.

[64] Krempl G, Kottke D, Lemaire V. Optimised probabilistic active learning (OPAL) — For fast, non-myopic, cost-sensitive active classification. *Machine Learning*, 2015, 100(2-3): 449-476.

[65] Settles B, Craven M, Ray S. Multiple-instance active learning. In *Proc. the 21st Annual Conference on Neural Information Processing Systems*, December 2007, pp.1289-1296.

[66] Roy N, McCallum A. Toward optimal active learning through sampling estimation of error reduction. In *Proc. the 18th Int. Conference on Machine Learning*, June 2001, pp441-448.

[67] Moskovitch R, Nissim N, Stopel D *et al*. Improving the detection of unknown computer worms activity using active learning. In *Proc. the 30th German Conference on AI*, September 2007, pp.489-493.

[68] Fang M, Li Y, Cohn T. Learning how to active learn: A deep reinforcement learning approach. In *Proc. the Conference on Empirical Methods in Natural Language Processing,* September 2017, pp.595-605.

[69] Liu M, Buntine W, Haffari G. Learning how to actively learn: A deep imitation learning approach. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.1874-1883.

[70] Pang K, Dong M, Wu Y *et al*. Meta-learning transferable active learning policies by deep reinforcement learning. arXiv:1806.04798, 2008. https://arxiv.org/abs/1806.04798, Nov. 2019.

[71] Bachman P, Sordoni A, Trischler A. Learning algorithms for active learning. In *Proc. the 34th Int. Conference on Machine Learning*, August 2017, pp.301-310.

[72] Cohn D, Ghahramani Z, Jordan M. Active learning with statistical models. In *Proc. the 1994 Annual Conference on Neural Information Processing Systems*, December 1994, pp.705-712.

[73] Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. *Neural Computation*, 1992, 4(1): 1-58.

[74] Schervish M. Theory of Statistics (1st edition). Springer, 1995.

[75] Long B, Chapelle O, Zhang Y, Chang Y, Zheng Y, Tseng B. Active learning for ranking through expected loss optimization. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(5): 1180-1191.

[76] Freund Y, Seung H, Shamir E, Tishby N. Selective sampling using the query by committee algorithm. *Machine Learning*, 1997, 28(23): 133-168.

[77] Krogh A, Vedelsby J. Neural network ensembles, cross validation, and active learning. In *Proc. the 8th Annual Conference on Neural Information Processing Systems*, November 1995, pp.231-238.

[78] Burbidge R, Rowland J J, King R D. Active learning for regression based on query by committee. In *Proc. the 8th Int. Conference on Intelligent Data Engineering and Automated Learning*, December 2007, pp.209-218.

[79] Cai W, Zhang Y, Zhou J. Maximizing expected model change for active learning in regression. In *Proc. the 13th International Conference on Data Mining*, December 2013, pp.51-60.

[80] Bottou L. Large-scale machine learning with stochastic gradient descent. In *Proc. the 19th Int. Conference on Computational Statistics*, August 2010, pp.177-186.

[81] Cai W, Zhang Y, Zhou S Y *et al*. Active learning for support vector machines with maximum model change. In *Proc. the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases*, September 2014, pp.211-226.

[82] Dasgupta S. The two faces of active learning. In *Proc. the 20th Int. Conference on Algorithmic Learning Theory*, October 2009, Article No. 1.

[83] Dasgupta S, Hsu D. Hierarchical sampling for active learning. In *Proc. the 25th Int. Conference on Machine Learning*, June 2008, pp.208-215.

[84] Urner R, Ben-David S. Probabilistic lipschitzness: A niceness assumption for deterministic labels. In *Proc. the 27th NIPS Learning Faster from Easy Data Workshop*, December 2013.

[85] Steinwart I, Scovel C. Fast rates for support vector machines using Gaussian kernels. *The Annals of Statistics*, 2007, 35(2): 575-607.

[86] Urner R, Shalev-Shwartz S, Ben-David S. Access to unlabeled data can speed up prediction time. In *Proc. the 28th Int. Conference on Machine Learning*, June 2011, pp.641-648.

[87] Verma N, Kpotufe S, Dasgupta S. Which spatial partition trees are adaptive to intrinsic dimension? In *Proc. the 25th Conference Uncertainty Artif. Intell.*, June 2009, pp.565-574.

[88] Urner R, Wulff S, Ben-David S. PlAL: Cluster-based active learning. In *Proc. the 26th Conference on Learning Theory*, June 2013, pp.376-397.

[89] Wang M, Min F, Zhang Z H, Wu Y X. Active learning through density clustering. *Expert Systems with Applications*, 2017, 85: 305-317.

[90] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. *Science*, 2014, 344(6191): 1492-1496.

[91] Yan Y, Rosales R, Fung G, Dy J. Active learning from crowds. In *Proc. the 28th Int. Conference on Machine Learning*, June 2011, pp.1161-1168.

[92] Fang M, Zhu X, Li B, Ding W, Wu X. Self-taught active learning from crowds. In *Proc. the 12th Int. Conference on Data Mining*, December 2012, pp.858-863.

[93] Shu Z, Sheng V S, Li J. Learning from crowds with active learning and self-healing. *Neural Computing and Applications*, 2018, 30(9): 2883-2894.

[94] Lampert C H, Nickisch H, Harmeling S. Attribute-based classification for zero-shot visual object categorization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014, 36(3): 453-465.

[95] Ertekin S, Huang J, Giles C L. Active learning for class imbalance problem. In *Proc. the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 2007, pp.823-824.

[96] Attenberg J, Ertekin Ş. Class imbalance and active learning. In *Imbalanced Learning: Foundations, Algorithms, and Applications*, He H B, Ma Y Q (eds.), John Wiley & Sons, Inc., 2013, pp.101-149.

[97] Tomanek K, Morik K. Inspecting sample reusability for active learning. In *Proc. the Workshop on Active Learning and Experimental Design*, May 2010, pp.169-181.

[98] Hu R, Namee B M, Delany S J. Active learning for text classification with reusability. *Expert Systems with Applications*, 2016, 45(C): 438-449.

[99] Settles B, Craven M, Friedland L. Active learning with real annotation costs. In *Proc. the 2008 NIPS Workshop on Cost-Sensitive Learning*, December 2008.

[100] Tomanek K, Hahn U. A comparison of models for cost-sensitive active learning. In *Proc. the 23rd Int. Conference on Computational Linguistics*, August 2010, pp.1247-1255.

[101] Liu A, Jun G, Ghosh J. Active learning of hyperspectral data with spatially dependent label acquisition costs. In *Proc. the 2009 IEEE International Geoscience and Remote Sensing Symposium*, July 2009, pp.256-259.

[102] Persello C, Boularias A, Dalponte M *et al*. Cost-sensitive active learning with lookahead: Optimizing field surveys for remote sensing data classification. *IEEE Trans. Geoscience and Remote Sensing*, 2014, 52(10): 6652-6664.

[103] Margineantu D. Active cost-sensitive learning. In *Proc. the 19th International Joint Conference on Artificial Intelligence*, July 2005, pp.1622-1623.

[104] Krishnamurthy A, Agarwal A, Huang T *et al*. Active learning for cost-sensitive classification. arXiv:1703.01014, 2017. https://arxiv.org/abs/1703.01014, May 2019.

[105] Zhang D, Wang F, Shi Z *et al*. Interactive localized content based image retrieval with multiple-instance active learning. *Pattern Recognition*, 2010, 43(2): 478-484.

[106] Wang R, Wang X, Kwong S *et al*. Incorporating diversity and informativeness in multiple-instance active learning. *IEEE Trans. Fuzzy Systems*, 2017, 25(6): 1460-1475.

[107] Wu J, Sheng V S, Zhang J, Zhao P, Cui Z. Multi-label active learning for image classification. In *Proc. the 21st IEEE Int. Conference on Image Processing*, October 2014, pp.5227-5231.

[108] Yang B, Sun J T, Wang T, Chen Z. Effective multi-label active learning for text classification. In *Proc. the 15th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, June 2009, pp.917-926.

[109] Pupo O, Morell C, Ventura S. Effective active learning strategy for multi-label learning. *Neurocomputing*, 2017, 273: 494-508.

[110] Cherman E A, Papanikolaou Y, Tsoumakas G *et al*. Multi-label active learning: Key issues and a novel query strategy. *Evolving Systems*, 2017, 10(1): 63-78.

[111] Rani M, Dhok S, Deshmukh R. A systematic review of compressive sensing: Concepts, implementations and applications. *IEEE Access*, 2018, 6: 4875-4894.

[112] Som S. Learning label structure for compressed sensing based multilabel classification. In *Proc. the 2016 SAI Computing Conference*, July 2016, pp.54-60.

[113] Wu J, Ye C, Sheng V *et al*. Active learning with label correlation exploration for multi-label image classification. *IET Computer Vision*, 2017, 11(7): 577-584.

[114] Pupo O, Ventural S. Evolutionary strategy to perform batch-mode active learning on multi-label data. *ACM Trans. Intelligent Systems and Technology*, 2018, 9(4): Article No. 46.

[115] Reichart R, Tomanek K, Hahn U, Rappoport A. Multi-task active learning for linguistic annotations. In *Proc. the 46th Association for Computational Linguistics*, June 2008, pp.861-869.

[116] Zhang Y. Multi-task active learning with output constraints. In *Proc. the 24th AAAI Conference on Artificial Intelligence*, July 2010, pp.667-672.

[117] Harpale A. Multi-task active learning [Ph.D. Thesis]. School of Computer Science, Carnegie Mellon University, 2012.

[118] Gavves E, Mensink T, Tommasi T *et al*. Active transfer learning with zero-shot priors: Reusing past datasets for future tasks. In *Proc. the 2015 IEEE International Conference on Computer Vision*, December 2015, pp.2731-2739.

[119] Wang X, Huang T, Schneider J. Active transfer learning under model shift. In *Proc. the 31st Int. Conference on Machine Learning*, June 2014, pp.1305-1313.

[120] Guo Y, Schuurmans D. Discriminative batch mode active learning. In *Proc. the 21st Annual Conference on Neural Information Processing Systems*, December 2007, pp.593-600.

[121] Chakraborty S, Balasubramanian V, Panchanathan S. Adaptive batch mode active learning. *IEEE Trans. Neural Networks and Learning Systems*, 2015, 26(8): 1747-1760.

[122] Shen P, Li C, Zhang Z. Distributed active learning. *IEEE Access*, 2016, 4: 2572-2579.

[123] Hinton G E, Osindero S, The Y. A fast learning algorithm for deep belief nets. *Neural Computing*, 2006, 18(7): 1527-1554.

[124] Wang K, Zhang D, Li Y *et al*. Cost-effective active learning for deep image classification. *IEEE Trans. Circuits and Systems for Video Technology*, 2017, 27(12): 2591-2600.

[125] Rahhal M M A, Bazi Y, Alhichri H *et al*. Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, 2016, 345(C): 340-354.

[126] Zhou S, Chen Q, Wang X. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 2013, 120: 536-546.

[127] Valiant L G. A theory of the learnable. *Communications of the ACM*, 1984, 27(11): 1134-1142.

[128] Hanneke S. A bound on the label complexity of agnostic active learning. In *Proc. the 24th Int. Conference on Machine Learning*, June 2007, pp.353-360.

[129] Hanneke S. Theoretical foundations of active learning [Ph.D. Thesis]. Machine Learning Department, CMU, 2009.

[130] Hanneke S. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 2014, 7(2/3): 131-309.

[131] Dasgupta S. Coarse sample complexity bounds for active learning. In *Proc. the 19th Annual Conference on Neural Information Processing Systems*, December 2005, pp.235-242.

[132] Tosh C, Dasgupta S. Diameter-based active learning. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.3444-3452.

[133] Audibert J Y, Tsybakov A B. Fast learning rates for plug-in classifiers. *The Annals of Statistics*, 2005, 35(2): 608-633.

[134] Minsker S. Plug-in approach to active learning. *Journal of Machine Learning Research*, 2012, 13: 67-90.

[135] Locatelli A, Carpentier A, Kpotufe S. Adaptivity to noise parameters in nonparametric active learning. In *Proc. the 30th Conference on Learning Theory*, July 2017, pp.1383-1416.

[136] Schein A I, Ungar L H. Active learning for logistic regression: An evaluation. *Machine Learning*, 2007, 68(3): 235-265.

[137] Melville P, Mooney R. Diverse ensembles for active learning. In *Proc. the 21st Int. Conference on Machine Learning*, July 2004, pp.584-591.

[138] Yang Y, Loog M. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 2018, 83: 401-415.

[139] Ramirez-Loaiza M E, Sharma M, Kumar G *et al*. Active learning: An empirical study of common baselines. *Data Mining and Knowledge Discovery*, 2017, 31(2): 287-313.

[140] Pupo O, Altalhi H, Ventura S. Statistical comparisons of active learning strategies over multiple datasets. *Knowledge-Based Systems*, 2018, 145(1):274-288.

[141] Merz C, Murphy P. UCI repository of machine learning databases. http://www.ics.uci.edu/_mlearn/MLRepository.html, Nov. 2019.

[142] Frey P W, Slate D J. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 1991, 6(2): 161-182.

[143] Xu L, Krzyzak A, Suen C. Methods of combining multiple classifiers and their applications to handwritten recognition. *IEEE Trans. Systems Man and Cybernetics*, 1992, 22(3): 418-435.

[144] Garofolo J, Lamel L, Fisher W *et al*. DARPA TIMIT acoustic phonetic continuous speech corpus CD-ROM. Technical Report, 1993. https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir4930.pdf, Nov. 2019.

[145] Craven M, DiPasquo D, Freitag D *et al*. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 2000, 118(1/2): 69-113.

[146] LeCun Y, Bottou L, Bengio Y *et al*. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 1998, 86(11): 2278-2324.

[147] Lang K. NewsWeeder: Learning to filter net news. In *Proc. the 12th Int. Conference on Machine Learning*, July 1995, pp.331-339.

[148] Deng J, Dong W, Socher R *et al*. ImageNet: A large-scale hierarchical image database. In *Proc. the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2009, pp.248-255.

[149] Sang E F, de Meulder F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. the 7th Conference on Natural Language Learning*, May 2003, pp.142-147.

[150] Collier N, Kim J. Introduction to the bio-entity recognition task at JNLPBA. In *Proc. the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, August 2004, Article No. 13.

[151] Yeh A, Morgan A, Colosimo M *et al*. BioCreAtIvE task 1A: Gene mention finding evaluation. *BMC Bioinformatics*, 2005, 6(S-1): Article No. 2.

[152] Vlachos A. Evaluating and combining biomedical named entity recognition systems. In *Proc. the Workshop on Biological, Translational, and Clinical Language Processing*, June 2007, pp.199-200.

[153] Peng F, McCallum A. Information extraction from research papers using conditional random fields. *Information Processing and Management*, 2006, 42(4): 963-979

[154] de Carvalho V R, Cohen W. Learning to extract signature and reply lines from email. In *Proc. the 1st Conference on Email and Anti-Spam*, July 2004.

[155] Guyon I, Cawley G, Dror G *et al*. Results of the active learning challenge. In *Proc. the Active Learning and Experimental Design Workshop*, May 2010, pp.19-45.

[156] Pace R K, Barry R. Sparse spatial autoregressions. *Stat. Probab. Lett.*, 1997, 33(3): 291-297.

[157] Bay S D, Kibler D, Pazzani M *et al*. The UCI KDD archive of large data sets for data mining research and experimentation. *SIGKDD Explor.*, 2000, 2(2): 81-85.

[158] Tang Y P, Li G X, Huang S J. ALiPy: Active learning in Python. arXiv:1901.03802, 2019. https://arxiv.org/abs/1901.03802, Nov. 2019.

[159] Yang Y Y, Lee S C, Chung Y A *et al*. libact: Pool-based active learning in Python. arXiv:1710.00379, 2017. https://arxiv.org/abs/1710.00379, October 2019.

[160] Tran V C, Nguyen N T, Fujita H *et al*. A combination of active learning and self-learning for named entity recognition on Twitter using conditional random fields. *Knowledge-Based Systems*, 2017, 132: 179-187.

[161] Scheffer T, Decomain C, Wrobel S. Active hidden Markov models for information extraction. In *Proc. the 4th Int. Conference on Advances in Intelligent Data Analysis*, September 2001, pp.309-318.

[162] Aldogan D, Yaslan Y. A comparison study on active learning integrated ensemble approaches in sentiment analysis. *Computers and Electrical Engineering*, 2017, 57(C): 311-323.

[163] Zhang H, Huang M, Zhu X. A unified active learning framework for biomedical relation extraction. *Journal of Computer Science and Technology*, 2012, 27(6): 1302-1313.

[164] Hoi S C H, Jin R, Zhu J *et al*. Batch mode active learning and its application to medical image classification. In *Proc. the 23rd Int. Conference on Machine Learning*, June 2006, pp.417-424.

[165] Wallace B C, Small K, Brodley C *et al*. Active learning for biomedical citation screening. In *Proc. the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2010, pp.173-182.

[166] Ma A, Patel N, Li M *et al*. Confidence based active learning for whole object image segmentation. In *Proc. the 2006 Int. Workshop on Multimedia Content Representation, Classification and Security*, September 2006, pp.753-760.

[167] Pavlopoulou, Kak A, Brodley C. Application of semisupervised and active learning to interactive contour delineation. In *Proc. the ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, August 2003, pp.26-33.

[168] Boutell M R, Luo J, Shen X *et al*. Learning multi-label scene classification. *Pattern Recognition*, 2004, 37(9): 1757-1771.

[169] Zhang B, Wang Y, Chen F. Multilabel image classification via high-order label correlation driven active learning. *IEEE Trans. Image Processing*, 2014, 23(3): 1430-1441.

[170] Top A, Hamarneh G, Abugharbieh R. Active learning for interactive 3D image segmentation. In *Proc. the 14th Int. Conference on Medical Image Computing and Computer-assisted Intervention*, September 2011, pp.603-610.

[171] Caicedo J C, Lazebnik S. Active object localization with deep reinforcement learning. In *Proc. the 2015 IEEE Int. Conference on Computer Vision*, December 2015, pp.2488-2496.

[172] Kim Y, Kim S. Design of aging-resistant Wi-Fi fingerprint-based localization system with continuous active learning. In *Proc. the 20th Int. Conference on Advanced Communication Technology*, February 2018, pp.s1054-1059.

[173] Ayache S, Quénot G. Video corpus annotation using active learning. In *Proc. the 30th European Conference on Information Retrieval Research*, March 2008, pp.187-198.

[174] Reker D, Schneider G. Active-learning strategies in computer-assisted drug discovery. *Drug Discovery Today*, 2015, 20(4): 458-465.

[175] Warmuth M K, Rätsch G, Mathieson M *et al*. Active learning in the drug discovery process. In *Proc. the 15th Annual Conference on Neural Information Processing Systems*, December 2001, pp.1449-1456.

[176] Figueroa R L, Zeng-Treitler Q, Ngo L *et al*. Active learning for clinical text classification: Is it better than random sampling? *Journal of the American Medical Informatics Association*, 2012, 19(5): 809-816.

[177] Chen Y, Lasko T, Mei Q *et al*. A study of active learning methods for named entity recognition in clinical text. *Journal of Biomedical Informatics*, 2015, 58(1): 11-18.

[178] Gu Y, Zydek D. Active learning for intrusion detection. In *Proc. the 2014 National Wireless Research Collaboration Symposium*, May 2014, pp.117-122.

[179] Hossain H M S, Roy N, Khan M. Active learning enabled activity recognition. In *Proc. the 2016 IEEE Int. Conference on Pervasive Computing and Communications*, March 2016, Article No. 26.

[180] Reker D, Schneider P, Schneider G. Multi-objective active machine learning rapidly improves structure-activity models and reveals new protein-protein interaction inhibitors. *Chemical Science*, 2016, 7(6): 3919-3927.

[181] Yan S, Chaudhuri K, Javidi T. Active learning with logged data. arXiv:1802.09069, 2018. https://arxiv.org/abs/1802.09069, Nov. 2019.

[182] Danka T, Horvath P. modAL: A modular active learning framework for Python. arXiv:1805.00979, 2018. https://arxiv.org/abs/1805.00979, Nov. 2019.

[183] Pedregosa F, Varoquaux G, Gramfort A *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011, 12: 2825 -2830.

[184] Atienza R. Advanced Deep Learning with Keras: Apply Deep Learning Techniques, Autoencoders, GANs, Variational Autoencoders, Deep Reinforcement Learning, Policy Gradients, and More. Packt Publishing, 2018.

Punit Kumar received his B.Tech. degree in computer science and engineering from Kurukshetra University, Kurukshetra, Haryana, in 2011, and his M.Tech. degree in computer science and engineering from Deenbandhu Chhotu Ram University of Science and Technology, Murthal, Haryana, in 2014. He is currently pursuing his Ph.D. degree in computer science at Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh.

Atul Gupta received his Ph.D. degree in computer science from the Department of Computer Science and Engineering, at IIT Kanpur, India. He is currently an associate professor with the Indian Institute of Information Technology, Design and Manufacturing (IIITDM), Jabalpur, Madhya Pradesh. He has served as head of Computer Science and Engineering Department during 2011–2012 and 2015–2016 at IIITDM Jabalpur, Madhya Pradesh. He is also looking after as Convener, Digital Initiatives and Convener Computer Center. He is a professional member of ACM and IEEE.