Learning to Sample: an Active Learning Framework

Jingyu Shao, Qing Wang and Fangbing Liu
Research School of Computer Science
Australian National University
Acton, ACT, Australia
{jingyu.shao, qing.wang, fangbing.liu}@anu.edu.au

***Abstract*—Meta-learning algorithms for active learning are emerging as a promising paradigm for learning the "best" active learning strategy. However, current learning-based active learning approaches still require sufficient training data so as to generalize meta-learning models for active learning. This is contrary to the nature of active learning which typically starts with a small number of labeled samples. The unavailability of large amounts of labeled samples for training meta-learning models would inevitably lead to poor performance (e.g., instabilities and overfitting). In our paper, we tackle these issues by proposing a novel learning-based active learning framework, called *Learning To Sample* (LTS). This framework has two key components: a sampling model and a boosting model, which can mutually learn from each other in iterations to improve the performance of each other. Within this framework, the sampling model incorporates uncertainty sampling and diversity sampling into a unified process for optimization, enabling us to actively select the most representative and informative samples based on an optimized integration of uncertainty and diversity. To evaluate the effectiveness of the LTS framework, we have conducted extensive experiments on three different classification tasks: image classification, salary level prediction, and entity resolution. The experimental results show that our LTS framework significantly outperforms all the baselines when the label budget is limited, especially for datasets with highly imbalanced classes. In addition to this, our LTS framework can effectively tackle the cold start problem occurring in many existing active learning approaches.**

***Index Terms*—active learning, meta-learning, uncertainty sampling, diversity sampling, boosting**

I. Introduction

Sampling is a fundamental technique for acquiring training data in machine learning applications. However, obtaining large amounts of manually labeled samples is often expensive or simply infeasible in practice. To alleviate this issue, active learning has been extensively studied in the past decades [30], which aims to select fewer labeled samples to train a machine learning model as effectively as possible, achieving similar or greater accuracy. At its core, active learning seeks for the most representative or informative samples to be labeled for training by leveraging observations from previously labeled samples [7], [8], [27].

To date, various active learning techniques have been developed from different perspectives [30], such as uncertainty sampling [34], [36], query-by-committee [31], error or variance minimization [15], [29], and expected model change [3]. They all attempted to address a key challenge in active learning: given a dataset, how to decide which samples in the dataset are more representative or informative than the others for training a machine learning model? However, as evidenced by the experiments presented in these works, there is no one-fit-all solution for active learning. Due to the variety of datasets and machine learning models, different active learning techniques may perform best in different circumstances, depending on the dataset at hand and the machine learning model being chosen.

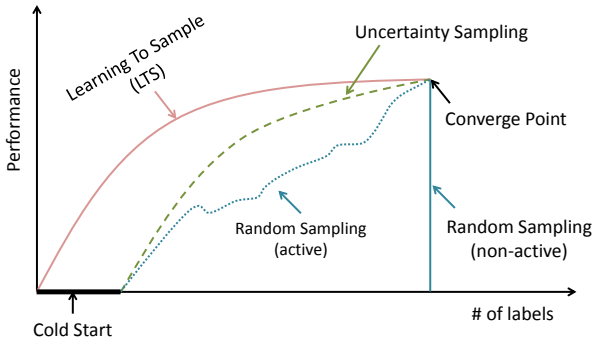

Fig. 1: An illustration of Learning To Sample (LTS) in relation to uncertainty sampling and random sampling, where random sampling (active) indicates that random samples are gradually selected during the iterations of active learning, and random sampling (non-active) indicates that all samples are randomly selected in a one-off manner (i.e., no active learning).

Recently, several learning-based active learning approaches have been proposed to address such limitations [17], [23]. Instead of using pre-defined strategies for active learning, these works considered to learn the "best" active learning strategy based on the estimated model performance of a meta-learning model. For example, Hsu and Lin [17] developed an approach to learn from the performance of a set of active learning strategies adaptively so as to decide a desired active learning strategy. Konyushkova et al. [23] proposed a learning based approach using the Monte Carlo method to predict the reduction of generalization error by each unlabeled instance. Nevertheless, these learning-based active learning approaches still require sufficient training data so as to generalize a meta-learning model. On the contrary, active learning typically starts with a small number of labeled samples (i.e., seed

samples) and gradually adds more labeled samples through an iterative learning process. Thus, a meta-learning model can only be trained on a small number of labeled samples at the beginning, which leads to poor performance (e.g., instabilities and overfitting).

In this paper, we aim to propose a learning-based active learning framework to enable a unified sampling process for selecting representative and information samples from different perspectives. Different from the previous active learning approaches, we ground our work based on the following observations: (1) Although uncertainty sampling is one of the widely used active learning techniques [25], uncertainty sampling alone tends to select samples that are similar to each other, i.e., samples being selected from a sample space often have similar features [36]. (2) Diversity sampling targets to select samples of different kinds (e.g., samples with different features), which is complementary to uncertainty sampling. Thus, the obstacle of uncertainty sampling can be circumvented by combining uncertainty sampling and diversity sampling into a unified sampling process. (3) To find the "best" way to integrate these two sampling strategies, meta-learning is a powerful tool, which can optimize this integration process by learning hints from the chosen machine learn models and datasets.

Based on the above observations, we design a novel learning-based active learning framework, called *Learning To Sample* (LTS). In a nutshell, the LTS framework consists of two key components: a sampling model G and a boosting model F, which are learned iteratively, and their results can mutually strength each other in iterations. As illustrated in Fig. 1, the goal of this LTS framework is to help machine learning models achieve better performance with less training data by providing a learning-based active learning process. The design of the LTS framework incorporates the uncertainty and diversity aspects of sampling into a unified process, which can also circumvent the cold start problem [7], [23].

Contributions In summary, the contributions in this work are as follows:

- We propose a novel active learning framework, namely *Learning To Sample* (LTS), in which a boosting model F and a sampling model G can dynamically learn from each other in iterations for improving the performance of each other.
- Our sampling model incorporates uncertainty and diversity of samples into a unified process for optimization. This allows us to actively select samples based on the joint impacts of probabilities of being mis-classified by a boosting model and the distribution of samples in a sample space.
- The experimental results show that our active learning approach significantly outperforms all the baselines when the label budget is limited, especially for those datasets with highly imbalanced classes. It also shows that our approach can effectively tackle the cold start problem.

It is worth noting that, technically, the boosting model F can be replaced by any classification model and the regressors in the sampling model G can be replaced by any regression model. Thus, the LTS framework is indeed not restricted to specific machine learning models used for classification and regression.

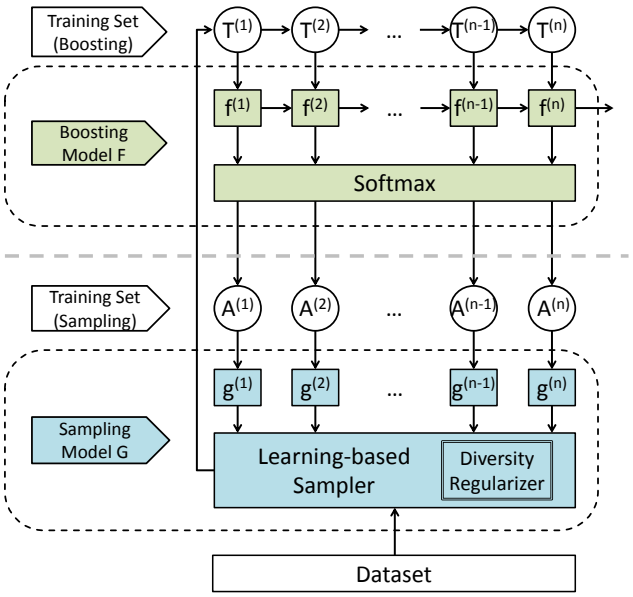

Fig. 2: The overall framework of Learning To Sample (LTS)

II. Learning To Sample Framework

In this section, we present our learning based active learning framework, called *Learning To Sample* (LTS).

As illustrated in Figure 2, the LTS framework has two key components: a boosting model F (highlighted in green) and a sampling model G (highlighted in blue). Accordingly, there are two learning processes that are closely coupled: (1) learning the boosting model F, and (2) learning the sampling model G. Specifically, a boosting model F aims to create a strong learner based on a set of weak learners. Thus, the boosting model F is trained iteratively on a sequence of incrementally built training sets in order to add new functions for improving its model performance. Samples in these training sets are actively selected by the sampling model G which is dynamically learned from the performance of the boosting model F during its iterative training process. In the following, we discuss the boosting model and the sampling model in detail.

A. Boosting Model

Let $X \subseteq \mathbb{R}^d$ be a dataset with $|X|$ instances and ζ be a budget on the total number of instances from X that can be labeled by a human oracle. A training set $T = \{(x_i, y_i)\}_{i=1}^{|T|}$, where $x_i \in X$ and $y_i \in \mathbb{R}$, consists of a set of instances from X and their labels from $\mathbb{R}$. This training set T is incrementally built as the boosting model interacts with the sampling model, i.e., a sequence of training subsets $\langle T^{(1)}, \ldots, T^{(n)} \rangle$ such that $T^{(1)} \subseteq T^{(2)} \subseteq \cdots \subseteq T^{(n)}$, $T^{(n)} = T$, and $|T^{(n)}| \leq \zeta$, where

$T^{(t)}$ for $t \in [1, n]$ is a training subset being used for training the boosting model at the t-th iteration.

A boosting model F trains a sequence of functions $\langle f^{(1)}, \ldots, f^{(n)} \rangle$ in an additive manner, where $f^{(t)}$ for $t \in [1, n]$ is a function being added into F at the t-th iteration. More specifically, the individual results of the first t-1 functions are combined to predict the label of an instance at the $(t$-1)-th iteration such that:

$$\hat{y}_i^{(t-1)} = \sum_{k=1}^{t-1} f^{(k)}(x_i). \tag{1}$$

Then, the t-th function $f^{(t)}$ is trained on the actively selected training subset $T^{(t)}$ by minimizing the following objective function:

$$\sum_{(x_i, y_i) \in T^{(t)}} \ell_1(\hat{y}_i^{(t-1)} + f^{(t)}(x_i), y_i) + \Omega_1(f^{(t)}) \tag{2}$$

where ℓ_1 is a differentiable loss function and $\Omega_1(f^{(t)})$ is the penalty for the complexity of $f^{(t)}$.

After the t-th function $f^{(t)}$ is learned, the boosting model F sends its feedback to the sampling model G via a softmax layer. This allows the sampling model G to leverage hints from the prediction results of $\langle f^{(1)}, \ldots, f^{(t)} \rangle$ and actively select the most informative instances as new samples for the next iteration, leading to $T^{(t+1)}$. We use the *Softmax* function [33] to obtain probabilities of being mis-classified for training samples. Specifically, in the t-th iteration, the softmax layer takes $\mathbf{l}^{(t)} = \langle \ell(\hat{y}_1^{(t)}, y_1), \ldots, \ell(\hat{y}_q^{(t)}, y_q) \rangle$ as input, where $q = |T^{(t)}|$ and each $\ell(\hat{y}_j^{(t)}, y_j)$ in $\mathbf{l}^{(t)}$ refers to the loss of a training sample x_j from $T^{(t)}$, then generates $\mathbf{z}^{(t)} = \langle z_1^{(t)}, \ldots, z_q^{(t)} \rangle$, i.e.,

$$z_i^{(t)} = \mathit{Softmax}(l_i^{(t)}), \tag{3}$$

where $\mathit{Softmax}(l_i^{(t)}) = e^{l_i^{(t)}} / \sum_{j=1}^{q} e^{l_j^{(t)}}$ and $l_i^{(t)} = \ell(\hat{y}_i^{(t)}, y_i)$.

B. Sampling Model

Let $X_L^{(t)} = \{x_i \in X | (x_i, y_i) \in T^{(t)}\}$ be the set of labeled instances and $X_U^{(t)} = X - X_L^{(t)}$ be the set of unlabeled instances in the t-th iteration. A sampling model G aims to select a set $\Delta^{(t)}$ of the most informative samples from unlabeled instances at the t-th iteration such that $X_L^{(t+1)} = X_L^{(t)} \cup \Delta^{(t)}$ and $X_U^{(t+1)} = X_U^{(t)} - \Delta^{(t)}$. Consequently, $T^{(t+1)} = T^{(t)} \cup \{(x_i, y_i) | x_i \in \Delta^{(t)}\}$ is generated and sent to the boosting model F for training the function f^{t+1}.

The question arising here is: how to actively select a set $\Delta^{(t)}$ of the most informative samples at the t-th iteration? In the LTS framework, two kinds of samples are primarily targeted: (1) samples that are likely to be mis-classified by the boosting model; (2) samples that have diverse features in the sample space. They relate to the uncertainty and diversity aspects of sampling, respectively. Hence, at the t-th iteration, the sampling model G learns to select a set $\Delta^{(t)}$ of most informative samples by maximizing the following objective:

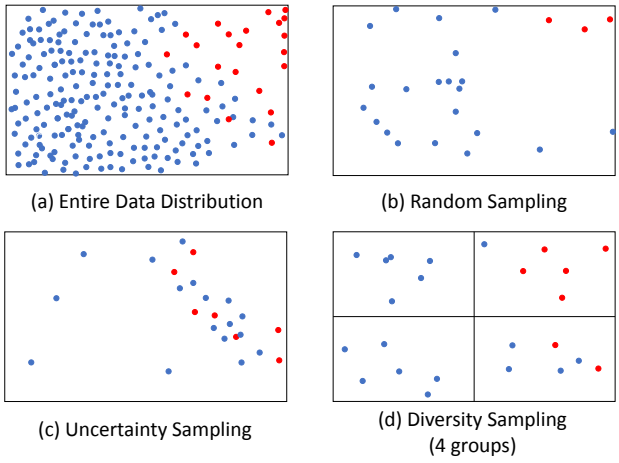

Fig. 3: Comparison of different sampling strategies, where 24 samples are selected in each of (b), (c) and (d).

$$\begin{aligned} \textbf{maximize} \quad & \sum_{i=1}^{k} v_i g^{(t)}(x_i) + \alpha \times \Gamma(\mathbf{v}) \\ \textbf{subject to} \quad & ||\mathbf{v}||_1 = |\Delta^{(t)}| \end{aligned} \tag{4}$$

where $k = |X_U^{(t)}|$, $\mathbf{v} = (v_1, ..., v_k)^T \in \{0, 1\}^k$, and each v_i is associated with an instance $x_i \in X_U^{(t)}$. When $v_i = 1$, it indicates that x_i is selected as a sample, and conversely, $v_i = 0$ indicates that x_i is not selected. The term $g^{(t)}(x_i)$ indicates the uncertainty score of an instance x_i which is predicated by a regressor $g^{(t)}$, and the regularization term $\Gamma(\mathbf{v})$ controls the distribution of selected instances in order to ensure their diversity in the sample space. α is a parameter used for balancing the impacts of uncertainty and diversity on samples, i.e., $\alpha > 1$ indicates that diverse samples are preferred, while $\alpha < 1$ indicates that samples with high probabilities of being mis-classified are preferred. Further details for our sampling model will be discussed in the next section.

III. Sampling Strategies

In the following, we discuss how the sampling model G handles the uncertainty and diversity aspects of samples. We first present an uncertainty sampling strategy by training a regressor $g^{(t)}$ in each iteration, then describe how the regularization term $\Gamma(\mathbf{v})$ is used to deal with diversity sampling.

Figure 3 illustrates our sampling strategies, i.e. uncertainty sampling and diversity sampling, in comparison with random sampling. Figure 3.(a) describes a real data distribution with two classes (red and blue). Figure 3.(b) shows that random sampling can only select very few samples from the minority class (red). Figure 3.(c) shows using uncertainty sampling leads to samples that are similar. Figure 3.(d) shows that diversity sampling can evenly select samples from different groups in the sample space.

A. Uncertainty Sampling

In the LTS framework, we predict the uncertainty of instances by learning from the performance of the boosting model, i.e. the training loss. We dynamically construct a training dataset to train a regressor for predicting the uncertainty in each iteration.

Formally, a training set $A^{(t)}$ for the sampling model G is constructed at the t-th iteration such that $A^{(t)} = \{(x_i, z_i^{(t)})|(x_i, y_i) \in T^{(t)}, z_i^{(t)} \in [0,1]\}$, where $\mathbf{z}^{(t)} = \langle z_1^{(t)}, \ldots, z_q^{(t)} \rangle$ is generated by the softmax layer of the boosting model F and $q = |T^{(t)}|$ as shown in Eq. 3. Thus, each training set $A^{(t)}$ contains the same set of instances as in $T^{(t)}$, but the labels of these instances in $A^{(t)}$ are different from the labels in $T^{(t)}$. Furthermore, each label $z_i^{(t)}$ represents the probability of being mis-classified of an instance x_i after the first t iterations. We then predict the uncertainty score $g^{(t)}(x_i)$ of an unlabeled instance $x_i \in X_U^{(t)}$ in Eq. 4 by solving a regression problem, i.e., training $g^{(t)}$ to minimize the following objective in the t-th iteration:

$$\sum_{(x_i, z_i^{(t)}) \in A^{(t)}} w_i^{(t)} \ell_2(g^{(t)}(x_i), z_i^{(t)}) + \Omega_2(g^{(t)}) \tag{5}$$

where ℓ_2 is also a differentiable loss function, $\Omega_2(g^{(t)})$ is the penalty for the complexity of $g^{(t)}$, and $w_i^{(t)}$ is a weighted value for x_i and is dynamically adjusted during the iterations. The intuition behind $w_i^{(t)}$ is to give higher weighted values to samples that are uncertain in more iterations, rather than samples that are uncertain in fewer iterations. For example, if a sample is mis-classified by the boosting model for a number of times, it will be assigned a higher weighted value than another sample which is mis-classified only once. We will present a method of assigning dynamic weighted values in Section IV.

B. Diversity Sampling

In the LTS framework, we deal with the diversity of samples by partitioning the sample space into a number of different groups such that instances in the same group are more similar than the instances in different groups. Then we use the regularization term $\Gamma(\mathbf{v})$ in Eq. 4 to regulate the sampling model, i.e., selecting samples from each group evenly.

Suppose that unlabeled instances in $X^{(t)}$ are partitioned into a set of groups $\{X_1^{(t)}, \ldots, X_b^{(t)}\}$ alike in certain features. Then we define the regularization term $\Gamma(\mathbf{v})$ over $\{X_1^{(t)}, \ldots, X_b^{(t)}\}$ using a $l_{2,1}$-norm function as:

$$\Gamma(\mathbf{v}) = ||\mathbf{v}||_{2,1} = \sum_{j=1}^{b} ||\mathbf{v}_j||_2 \tag{6}$$

where b is the total number of groups associated with $X_U^{(t)}$, $\mathbf{v}$ is partitioned into $\{\mathbf{v}_1, \ldots, \mathbf{v}_b\}$ where $\Sigma_{j=1}^{b}|\mathbf{v}_j| = |\mathbf{v}|$, $\mathbf{v}_j \in \{0,1\}^m$, $m = |X_j^{(t)}|$ and $j \in [1, b]$. That is, $||\mathbf{v}_j||_2$ is the l_2-norm of $\mathbf{v}_j$ that is a binary vector whose elements correspond to instances in group $X_j^{(t)}$.

It is known that the $l_{2,1}$-norm favors on selecting samples with diversity [20]. When the value of the $l_{2,1}$-norm is small, non-zero entries of $\mathbf{v}$ are concentrated in a small number of groups, i.e. the distribution of samples is limited to a small number of groups and accordingly the diversity of samples is low. On the contrary, when maximizing the $l_{2,1}$-norm in Eq. 4, there is a counter-effect on the distribution of samples, i.e. non-zero entries of $\mathbf{v}$ are widely distributed w.r.t. as many groups as possible and thus the diversity of samples is high.

Example 3.1: Consider Figure 3(d) in which the sample space is partitioned into four groups and a number of 24 samples will be selected. If we select 6 samples from each group, $||\mathbf{v_j}||_2 = \sqrt{6}$, we have $\Gamma(\mathbf{v}) = \sum_{j=1}^{4} ||\mathbf{v}_j||_2 = \sqrt{6} \times 4 = 9.8$. If we select 24 samples from only one group, $||\mathbf{v_j}||_2 = \sqrt{24}$, then $\Gamma(\mathbf{v}) = \sum_{j=1}^{1} ||\mathbf{v}_j||_2 = \sqrt{24} = 4.9$.

IV. Algorithm Description

In this section, we propose an algorithm for the LTS framework and discuss several important aspects of this algorithm which may influence the effectiveness of sampling.

A high-level description of the algorithm is presented in Algorithm 1. This algorithm takes a k-grouped dataset, a label budget and the number of iterations as input. The first step is to initialize the training set T^0 and select a set of seed samples from k groups using our diversity sampling strategy (Lines 1-2). Then the algorithm iterates to train a boosting model by actively selecting samples (Lines 4-9). For each t-th iteration, we first update the training set $T^{(t)}$ by adding newly selected samples $\Delta^{(t-1)}$ into the previous training set $T^{(t-1)}$ (Line 4). Then an additive function $f^{(t)}$ is trained for the boosting model F (Line 5). After that, a new training set $A^{(t)}$ is generated for the sampling model G based on the output of the current F (Line 6), and a regressor is trained for uncertainty prediction (Line 7). We then update the groups $\{X_1^{(t)}, \ldots, X_k^{(t)}$ by excluding the previous selected samples in $\Delta^{(t-1)}$, and select a new set of samples $\Delta^(t)$ based on Eq. 4 Eq. 4 (Lines 8 - 9). The algorithm finally yields a trained boosting model as output.

In the following, we first focus on discussing three important aspects of the algorithm: (i) How to decide dynamic weighted values for samples? (ii) How to partition a sample space into different groups? (iii) How to distribute a given label budget across iterations? Then, we will discuss how the cold start problem can be alleviated by our algorithm.

A. How to decide dynamic weighted values for samples?

During the training process of the boosting model, some samples in the training set may have high training losses in a number of iterations. Such samples are often informative for predicting uncertainty. Thus, a dynamic weighted value $w_i^{(t)}$ is assigned to each sample x_i to indicate its importance, as shown in Eq. 5. By extending the work by Freund and Schapire [12], we develop the following method of assigning dynamic weighted values in the LTS framework. In each iteration, dynamic weighted values of samples are updated in two steps:

Algorithm 1: Learning To Sample (LTS)

Input: X with k groups, i.e. $\sum_{i=1}^{k} X_i^{(0)} = X$; label budget ζ; Balancing parameter α; Number of iterations n;

Output: A boosting model F

1 Initialize $T^{(0)} = \emptyset$

2 Select a set of seed samples $\Delta^{(0)}$ from k groups to maximize $\Gamma(\mathbf{v})$, where $|\Delta^{(0)}| = \frac{\zeta}{n}$

3 **for** *t = 1, ..., n* **do**

4 Update $T^{(t)} = T^{(t-1)} + \Delta^{(t-1)}$

5 Train an additive function $f^{(t)}$ by minimizing the objective in Eq. 2 using $T^{(t)}$

6 Generate a training set $A^{(t)}$

7 Train a regression function $g^{(t)}$ by minimizing the objective in Eq. 5 using $A^{(t)}$

8 Update $X_i^{(t)} = \{x \in X_i^{(t-1)} | x \notin \Delta^{(t-1)}\}$, where $i = 1, \ldots, k$

9 Select a set of samples $\Delta^{(t)}$ from $\sum_{i=1}^{k} X_i^{(t)}$ by maximizing the objective in Eq. 4, with $|\Delta^{(t)}| = \frac{\zeta}{n}$

(1) **Initialization:** For each new sample x_i at the t-th iteration, i.e. a sample in $\Delta^{(t-1)}$, we have:

$$w_i^{(t-1)} = \frac{1}{|\Delta^{(t-1)}|}. \quad (7)$$

(2) **Adjustment:** Then, the weighted value for each sample x_i in $A^{(t)}$ is re-calculated as:

$$w_i^{(t)} = w_i^{(t-1)} \times \frac{e^{-\frac{1}{2} ln(\frac{1-\epsilon^{(t-1)}}{\epsilon^{(t-1)}}) g^{(t-1)}(x_i) z_i^{(t-1)}}}{Z_t}, \quad (8)$$

where $\epsilon^{(t-1)} = \frac{\sum_i z_i^{(t-1)}}{|T^{(t-1)}|}$ and Z_t is a normalization factor ensuring that the sum of all weighted values of samples in $A^{(t)}$ equals to 1.

In our algorithm, a regressor $g^{(t)}$ is iteratively trained by minimizing the objective in Eq. 5, in which dynamic weighted values are updated using the above method in each iteration.

B. How to partition a sample space into groups?

A key challenge of diversity sampling is: how to partition a sample space into groups such that instances in the same group are more similar than instances in different groups? In many real-world applications, samples that have same features are likely to be more similar than samples that have different features. Thus, we consider to partition a sample space based on available features of samples. This can also avoid common issues of sampling based on a data distribution, such as selecting too many similar samples from high density areas. In doing so, diversity sampling in our algorithm can select samples that are complementary to ones being selected by uncertainty sampling.

Formally, given a sample space with d features, a label budget ζ and a number n of iterations, we partition the sample space into k groups where $k = \lceil \sqrt[d]{\frac{\zeta}{n}} \rceil^d$ and $\lceil\ \rceil$ indicates the ceiling function. For example, if we have $\zeta = 600$, $n = 20$ and $d = 4$, then $k = \lceil \sqrt[4]{\frac{600}{20}} \rceil^4 = \lceil 2.34 \rceil^4 = 81$, i.e., 81 groups. Each of such groups corresponds to an area in the sample space and samples from the same area have some common features.

C. How to distribute label budget across iterations?

Under a given label budget ζ, when more samples are selected at the beginning of the training process, it implies that less samples can be used in the later iterations to leverage hints from observed samples for improving performance. For example, when $|\Delta^{(1)}| = \zeta$, i.e., all samples are used in the first iteration, the training process in the LTS framework would be the same as in the traditional training process. On the other hand, if allocating more samples to the later iterations, the boosting model F would have higher variance in the early iterations, but a better chance to "bias" samples for active learning in the later iterations.

In our algorithm, we distribute a label budget equally over all iterations, i.e., $|\Delta^{(t)}| = \zeta/n$ for any $t \in [1, n]$ (Line 2 of Algorithm 1). An alternative is to distribute samples in an exponentially decreasing manner over iterations, i.e., $|\Delta^{(t)}| = \zeta/2^t$. As will be discussed in our experiments later, the former approach outperforms the latter one in almost all cases.

D. Discussion

As reported in the previous works [7], [23], the *cold start* problem often occurs in active learning because only a small amount of labeled samples is available in early iterations. Essentially, this is due to the inability of making reliable predictions by a machine learning model if training data is not sufficient. When a dataset has highly imbalanced classes (i.e., the number of instances from a majority class is much more than the number of instances from a minority class), the cold start problem can be further aggravated. Treating samples of all classes equally often leads to selecting samples that are likely to be similar or highly correlated, and thus are not representative [20], [36].

In the LTS framework, the uncertainty of samples is measured using a regressor that is dynamically trained on samples labeled with their losses from the boosting model. If we select samples by only taking the uncertainty of samples into consideration, the cold start problem would also occur in our work. Since one of the reasons underlying the cold start problem is that training data is too small to be representative,

TABLE I: Characteristics of datasets

Classification Tasks	Datasets	# Attributes	# Instances ($\|X\|$)	# Classes	Types of Labels	Class Imbalance Ratio
Image classification	Mnist	28×28	60,000	10	10 digits (i.e. 0-9)	N/A
Salary level prediction	Adult	14	48,842	2	{above 50k, not above 50k}	1 : 3
Entity resolution	Cora	12	837,865	2	{match, non-match}	1 : 49
	DBLP-Scholar	4	168,112,008	2	{match, non-match}	1 : 71,233
	DBLP-ACM	4	6,001,104	2	{match, non-match}	1 : 2,698
	NCVoter	18	10M	2	{match, non-match}	1:420

we thus partition a sample space into a number of groups based on similarity of features and introduce the regularization term $\Gamma(\mathbf{v})$ to ensure that more representative samples are selected from such a k-grouped sample space. Our experiments show that this approach works effectively for addressing the cold start problem (the experimental results will be discussed later in Section V).

V. Experiments

We have conducted experiments to empirically verify our LTS approach, aiming to answer the following questions:

(1) Given a limited label budget, how does our LTS approach perform in comparison with other sampling methods?
(2) How effectively can our LTS approach deal with the cold start problem and the class imbalance problem?
(3) How does the balancing parameter α affect the performance of our LTS approach?
(4) How do two sampling distribution methods perform, i.e. equal distribution vs exponentially decreasing distribution?
(5) How does our LTS approach perform in reducing label budgets while still achieving the same level of quality for classification as other sampling methods?

A. Experimental Setup

We evaluate our LTS framework on three different classification tasks: image classification, salary level prediction, and entity resolution [32]. The first is a multi-class classification task, while the other two are binary classification tasks.

Datasets. Six datasets are used in our experiments: (1) *Mnist*[1] dataset contains 28×28 images, and each image corresponds to a handwritten digit. The task is to classify the images into ten categories, i.e. from 0 to 9. (2) *Adult*[2] dataset contains adults' personal information. The task is to predict if a person's salary income is more than 50k. (3) *Cora*[3] dataset contains bibliographic records of machine learning publications. (4) *DBLP-Scholar*[3] dataset contains bibliographic records from the DBLP and Google Scholar websites. (5) *DBLP-ACM* [24] dataset contains bibliographic records from the DBLP and ACM websites. (6) *North Carolina Voter Registration (NCVoter)*[4] dataset contains real-world voter registration information of people from North Carolina in the USA. The datasets (3)-(6) are used for entity resolution, which aims to detect if two records from one or two datasets refer to the same entity (i.e. to classify two records as being a match or a non-match).

Table I summarizes the characteristics of the above six datasets. We can see that the datasets for entity resolution are highly imbalanced, i.e., the number of instances from the majority class (non-match) is much more than the number of instances from the minority class (match) in these datasets.

Baseline methods. We use the following baseline methods: (1) *CART* [1], short for Classification And Regression Tree, is a decision tree approach. (2) *XG* [4], short for eXtreme Gradient Boosting, is a widely used and state-of-the-art boosting approach for decision trees. (3) *XG+RS*, refers to applying XG on training sets built using the random sampling strategy. (4) *XG+US*, refers to applying XG on training sets built only using the uncertainty sampling strategy, i.e., $\alpha = 0$ in our LTS framework. (5) *XG+DS*, refers to applying XG on training sets built only using the diversity sampling strategy, i.e., $\alpha \to \infty$ in our LTS approach. For clarity, our LTS approach is denoted as *XG+LTS*. To evaluate how the exponentially decreasing distribution of samples may affect performance, we denote a variant of *XG+LTS* as *XG+LTS(E)* which only differs from XG+LTS in distributing samples in an exponentially decreasing manner. By default, we set $\alpha = 1$ for XG+LTS and *XG+LTS(E)*, unless otherwise stated. For XG, the maximum depth of each tree is 5, and other parameters are set as default as used in [4].

Measures. We use *accuracy* to evaluate the classification results over the first two datasets, i.e. Mnist and Adult. As the datasets of entity resolution tasks are highly imbalanced, we use *precision*, *recall* and *f-measure* as measures for entity resolution instead of accuracy. Basically, *recall* is the fraction of true positives among the total number of true matches, *precision* is the fraction of true positives over all positives, and *f-measure* (FM) is the harmonic mean of recall and precision, i.e. $FM = \frac{2*Recall*Precision}{Recall+Precision}$.

Label budgets. In our experiments, for each dataset X, we specify a label budget in terms of a certain percentage of the size of the dataset ($|X|$). For example, when using 1% as the label budget for the dataset NCVoter, i.e. 1% of $|X|$, we have 100,000 samples because NCVoter contains 10M instances in total. We also set $n = 20$ (i.e., 20 iterations), and distribute a label budget as follows:

- For the methods CART and XG, a label budget is used in the first iteration to randomly select all samples within the given label budget for training.

[1]Available from: *http://yann.lecun.com/exdb/mnist/*
[2]Available from: *https://archive.ics.uci.edu/ml/datasets/adult*
[3]Available from: *http://secondstring.sourceforge.net*
[4]Available from: *http://alt.ncsbe.gov/data/*

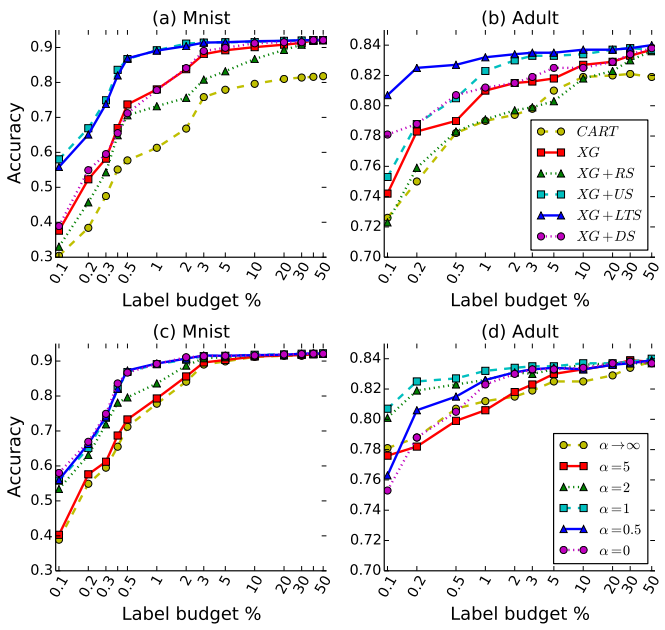

Fig. 4: Comparison of accuracy results for image classification and salary level prediction tasks under different label budgets

- For the methods XG+RS, XG+US, XG+DS and XG+LTS, a given label budget is evenly divided over 20 iterations. For example, given a label budget 1% for NCVoter, 5,000 samples are used in each iteration for 20 iterations.
- For the method XG+LTS(E), a given label budget is divided over 20 iterations in an exponentially decreasing manner.

B. Results and Discussion

We discuss our experimental results to answer the aforementioned questions at the beginning of this section.

1) Performance under different label budgets: Figure 4 presents the performance (accuracy) of our approach and the baseline methods on the first two datasets: Mnist and Adult. The f-measure results of entity resolution are presented in Table II. Generally, for all the datasets, all the methods converge, except CART, when the label budget is sufficient, e.g. 50% of the total instances are labeled for training in Mnist and Adult and 5% in Cora. XG+LTS outperforms all the baselines over all the datasets. The balancing parameter α for the best performance varies, depending on label budgets and datasets. For example, when the label budget is 5%, XG+LTS with $\alpha = 1$ performs best in Cora and XG+LTS with $\alpha = 0.5$ performs best in DBLP-ACM. When the label budget is relatively small, e.g. less than 1%, XG+DS achieves a better performance than XG+US in all datasets except for Mnist. When the label budget is larger, e.g. in the range 1% to 10%, XG+US performs better than XG+DS. In all cases, CART has the worst performance among all the methods, which is followed by XG+RS.

For the dataset Mnist, both XG+US and XG+LTS obtain better results than the others. The reason why XG+DS does not perform well is due to the large feature space of Mnist. There are in total 784 features in this dataset. Thus, the number of groups is much larger than the number of samples being selected in each iteration, which leads to suboptimal performance. For the dataset Adult, XG+DS performs better than XG+US when the label budget is limited, e.g. less than 0.2%. However, XG+US achieves better performance when the label budget increases, e.g. more than 1%. For the other datasets, the baselines CART, XG, XG+RS and XG+US have no result when the label budget is small, e.g. 0.01% in Cora and NCVoter, 0.1% in DBLP-ACM and DBLP-Scholar. However, both XG+LTS and XG+DS achieve good performance, even when the label budget is small.

From Figure 4 and Table II, we draw the following conclusions: (1) Both uncertainty sampling and diversity sampling contribute to the improvement of the performance. (2) When the label budget is limited, diversity sampling can select informative samples more effectively. However, when the label budget is sufficient, diversity samples are less informative than uncertainty samples.

2) Cold start problem and class imbalance problem: As shown in Figure 4 and Table II, when the label budget is small, i.e. 0.01% and less in Cora, 0.5% and less in NCVoter and DBLP-ACM, and 0.1% and less in DBLP-Scholar, the methods CART, XG, XG+RS and XG+US have the cold start problem (i.e, the FM values are zero). Compared with these methods, XG+LTS only has the cold start problem in the case that the label budget is 0.1% in DBLP-ACM. More interestingly, XG+DS does not have the code start problem in all settings of our experiments over all datasets. Since XG+DS is a special case of XG+LTS, this indicates that, when the label budget is small, we can handle the cold start problem by choosing a high value for the parameter α.

The four datasets used for entity resolution are highly imbalanced. We can see from Table II that XG+DS outperforms all the other methods when the label budget is small, while all the baselines have no result. When a dataset is highly imbalanced, samples from the majority class are likely to be selected and samples from the minority class are often ignored, which aggravates the cold start problem.

3) Performance under different values of balancing parameter α: Figure 4 and Table II show that we have conducted experiments on different values of α (i.e. $\alpha \in \{0, 0.5, 1, 2, 5, \infty\}$) over all six datasets. When the value of α increases, the XG+LTS approach biases more on the diversity. When the label budget increases, the XG+LTS approach achieves better performance with a smaller value of α. When the budget is low, e.g. less than 0.1% in Cora dataset, a larger α has a better performance. It indicates that diversity sampling contributes more when the label budget is smaller. On the other hand, when the budget is relatively high, e.g. larger than 5% in DBLP-ACM and DBLP-Scholar, a smaller α can achieve better performance, and the f-measure results from

TABLE II: Comparison of f-measure results for entity resolution tasks under different label budgets

Dataset	Label Budget ζ (% of $\|X\|$)	CART	XG	XG+RS	XG + US	XG+LTS				XG + DS	XG + LTS(E)
					$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$	$\alpha = 5$	$\alpha \to \infty$	$\alpha = 1$
Cora	0.01	0	0	0	0	0.637	0.857	0.861	0.867	**0.878**	0.862
	0.05	0.741	0.763	0.750	0.827	0.851	0.864	0.870	0.883	**0.885**	0.867
	0.1	0.788	0.796	0.787	0.823	0.863	0.862	0.873	**0.887**	0.886	0.870
	0.5	0.848	0.835	0.835	0.873	0.893	**0.900**	0.895	0.895	0.893	0.890
	1	0.868	0.878	0.880	0.870	0.896	0.902	**0.904**	0.898	0.894	0.896
	5	0.878	0.897	0.892	0.907	0.912	**0.915**	0.913	0.902	0.898	0.904
NCVoter	0.01	0	0	0	0	0.403	0.324	0.403	0.752	**0.875**	0.571
	0.05	0	0	0	0	0.903	0.954	0.989	**0.993**	0.991	0.934
	0.1	0	0	0	0	0.989	**0.994**	0.993	0.993	0.993	0.993
	0.5	0	0	0	0	0.993	**0.994**	0.993	0.993	0.991	**0.994**
	1	0.334	0.379	0.398	0	0.993	0.993	0.993	0.992	**0.994**	0.993
	5	0.993	0.993	0.994	0.993	0.993	**0.997**	0.993	0.994	0.993	0.994
DBLP-ACM	0.1	0	0	0	0	0	0	0	0	**0.397**	0
	0.5	0	0	0	0	0.382	0.702	**0.720**	0.651	0.632	0.679
	1	0.348	0.347	0.279	0	0.813	**0.878**	0.778	0.730	0.721	0.793
	2	0.599	0.767	0.680	0.403	0.851	**0.884**	0.867	0.789	0.783	0.854
	5	0.870	0.850	0.803	0.874	**0.935**	0.931	0.889	0.837	0.833	0.891
	10	0.903	0.911	0.890	0.926	**0.983**	0.981	0.937	0.893	0.899	0.933
DBLP-Scholar	0.1	0	0	0	0	0.586	0.723	0.733	**0.741**	0.731	0.727
	0.5	0.378	0.54	0.498	0.555	0.764	0.773	**0.794**	0.790	0.780	0.781
	1	0.562	0.669	0.659	0.738	0.793	0.804	**0.808**	0.793	0.792	0.794
	2	0.772	0.806	0.771	0.807	0.810	**0.815**	0.813	0.799	0.801	0.811
	5	0.773	0.822	0.803	0.836	**0.838**	0.836	0.831	0.821	0.818	0.828
	10	0.808	0.835	0.830	**0.865**	0.859	0.851	0.844	0.837	0.829	0.853

high α is much smaller, e.g. in DBLP-ACM, the performance of $\alpha = 5$ is about 10% less than that of $\alpha = 0.5$. It indicates that uncertainty sampling contributes more when the label budget is relatively large. The f-measure results in NCVoter are not distinguishable under various values of α when the label budget is greater than 1%, since all the f-measure results are similar, i.e. larger than 0.99.

4) Performance under different sampling distribution methods: Now we discuss the experimental results of the LTS approach when using two different sampling distribution methods, i.e. XG+LTS and XG+LTS(E). The experimental results are presented in Figure 5. We can see that XG+LTS obtains better f-measure results in almost all cases, except for two settings where the label budgets are very small: 0.01% in Cora and 0.1% in DBLP-Scholar. This is due to that diversity sampling contributes more in these cases. Therefore, in our LTS approach, we choose equal sampling distribution rather than exponentially decreasing sampling distribution.

5) Comparison of label budgets under the same performance: Table III presents our experimental results on the four datasets for entity resolution. We set the desired FM value as 0.9 for each dataset, except for the dataset DBLP-Scholar. This is because the dataset DBLP-Scholar is noisy and a classification result with the FM value 0.9 can hardly be achieved. Therefore, we set the desired FM value 0.8 for this dataset. Then we record the amount of label budgets required by each method in order to achieve the desired F-measure values. From Table III, we can see that, our XG+LTS method ($\alpha = 1$) requires the smallest number of samples for each of these datasets, in comparison with the other baseline methods. Especially, for the dataset NCVoter, our XG+LTS approach requires a significantly smaller number of samples for achieving the same performance, in comparison with the baseline methods CART, XG, XG+RS and XG+US. Although XG+DS requires a comparable label budget as our XG+LTS method for the dataset NCVoter, it requires at least a double amount of label budgets for the other three datasets.

TABLE III: Comparison of label budgets w.r.t. classification results with desired FM values, where XG+LTS has $\alpha = 1$.

Dataset	Cora	DBLP-ACM	DBLP-Scholar	NCVoter
CART	5%	10%	10%	3%
XG	4%	8%	2%	2%
XG + RS	5%	12%	5%	2%
XG + US	2%	7%	2%	7%
XG + DS	3%	10%	2%	**0.03%**
XG + LTS	**0.5%**	**4%**	**0.9%**	**0.03%**
FM values	0.9	0.9	0.8	0.9

VI. Related Work

A. Active Learning

The goal of active learning is to enable a machine learning based model, to achieve better performance with relatively fewer but representative training samples, especially when the labels are expensive and very hard to obtain. These samples may be selected from an unlabeled dataset by posing queries and then asking labels from an oracle [30]. Despite a large number of studies on developing active learning approaches, it is still difficult for a specific task to determine its best-suited

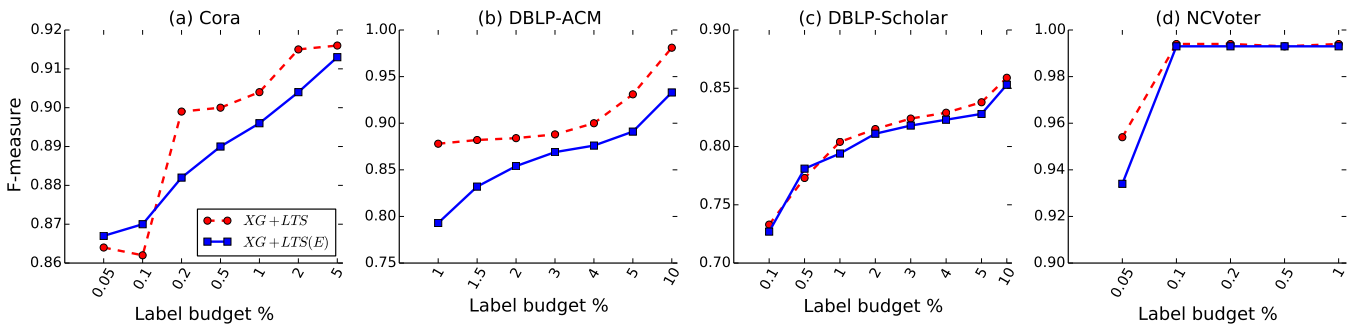

Fig. 5: Comparison of f-measure results for the LTS approach under two different sampling distributions

one. Thus, meta-learning algorithms have attracted much attention in recent years, driven by the desire to automate the selection process of active learning approaches. For example, Hsu and Lin [17] proposed a learning based active learning approach, which allowed a model to adaptively learn from a number of sampling strategies.

Among various active learning approaches, uncertainty sampling is one of the widely used techniques, which was first proposed by Lewis and Gale [25]. Normally, uncertainty sampling approaches select samples by measuring their uncertainty, such as probabilistic confidence [6], fisher information [30], entropy [16] and so on. This technique is usually associated with a probabilistic learning model in order to infer labels with the highest probability [22], [28]. A common issue of uncertainty sampling approaches, although computationally efficient and simple to use, is that they do not consider the diversity of data, for example, data with imbalanced class distribution [10]. Furthermore, most of existing uncertainty sampling techniques have the limitation that a sample can be an uncertain sample to one class but a certain sample to another class [18].

Diversity sampling is also a useful technique in active learning [2], [35], which aims to select representative samples according to the data distribution. In practice, although uncertain samples are often similar to each other [36], diversity sampling requires samples to be dissimilar in certain features. Thus, samples from different groups or classes are more preferred. In our work, we adopt the $l_{2,1}$ norm [20] for diversity sampling.

B. Learning based Active Learning

Two kinds of learning based active learning approaches have been proposed in the literature: One learns to select active learning strategies for a given dataset; The other builds a machine learning model to rank samples for selection.

Hsu and Lin [17] proposed *Active Learning by Learning* (ALBL) which relates active learning with multi-armed bandit learner. This approach aims to learn from the performance of a set of active learning strategies so as to decide which is the best. Chu and Lin extended this work by transferring the experience on active learning strategies from one dataset to different datasets [5].

The key idea of a recent work called *Learning Active Learning* (LAL) [23] is to train a regressor which can predict the generalization error reduction of each unlabelled instance and greedily select one with highest error reduction for labelling. This regressor can be trained as follows: First, given two training sets differing in only one sample, a pair of classifiers is trained, and the corresponding error reduction value of the sample is obtained. Second, the parameters from different pairs of classifiers and the corresponding error reduction values are collected using the Monte Carlo method to train the regressor. Compared with LAL, our LTS framework captures uncertainty of samples in a learning process w.r.t. a sampling model G. More specifically, our LTS framework first predicts samples' probabilities of being mis-classified by a machine learning model F, and based on that, a sampling model G is then trained.

There are several other approaches named with "learning to sample". For example, Li et al. [26] proposed a generative adversarial network (GAN) based sampling approach which learns to generate synthesized samples by learning likelihood ratios. This approach can also learn to draw samples from an un-normalized distribution via a reference distribution or using Markov Chain Monte Carlo (MCMC). Jamshidi et al. [19] proposed a transfer learning based approach, which learns the changing of each environment repeatedly for sample selection in configurable software systems. Dovrat et al. [9] proposed an approach to simplify 3D point clouds by matching them to a fixed size of samples via a learned deep network. However, all these approaches do not specifically focus on developing active learning techniques.

C. Boosting Techniques

A number of *boosting* techniques have been proposed which use a set of weak learners (e.g. decision tree and SVM) to create a single strong learner [21]. Freund developed the first boosting algorithm [11]. Later on, the first adaptive boosting approach, called *AdaBoost*, was proposed [13], in which the parameters of a model can be self-adjusted based on the actual performance in each iteration, including weights for samples and weights for additive learners. Compared with *AdaBoost*, which favors on dealing with classification tasks,

Gradient Boosting [14] approaches were proposed to solve both classification and regression problems by reducing the loss of a model in a gradient descent way. The state-of-the-art gradient boosting approach is *XGBoost* [4]. With the use of the sparsity-aware algorithm and the weighted quantile sketch for approximate learning, *XGBoost* can deliver accuracy results efficiently.

VII. Conclusion

In this paper, we have proposed a novel learning based active learning framework called learning to sample. This framework is composed of a sampling model G and a boosting model F. The boosting model is constructed based on a dynamic training set with an increasing number of samples in each iteration. These additional samples are selected iteratively by the sampling model which can learn from the performance of the boosting model through a unified process for two sampling strategies: uncertainty sampling(US) and diversity sampling(DS). The experimental results show that our approach outperforms all the baselines, particularly when the number of samples is relatively small. In addition to this, our framework can handle the cold start problem and the class imbalance problem.

Acknowledgment

This work was partially funded by the Australian Research Council (ARC) under Discovery Project DP160101934.

References

[1] Leo Breiman. *Classification and regression trees*. Wadsworth International Group, 1984.

[2] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

[3] Wenbin Cai, Yexun Zhang, Ya Zhang, Siyuan Zhou, Wenquan Wang, Zhuoxiang Chen, and Chris Ding. Active learning for classification with maximum model change. *ACM Transactions on Information Systems (TOIS)*, 2017.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd international conference on Knowledge Discovery and Data mining (SIGKDD)*, 2016.

[5] Hong-Min Chu and Hsuan-Tien Lin. Can active learning experience be transferred? In *Proceedings of the 16th International Conference on Data Mining (ICDM)*, 2016.

[6] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the AAAI conference on artificial intelligence*, 2005.

[7] Yue Deng, KaWai Chen, Yilin Shen, and Hongxia Jin. Adversarial active learning for sequences labeling and generation. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018.

[8] Pinar Donmez and Jaime G Carbonell. Paired-sampling in density-sensitive active learning. 2008.

[9] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the international Conference on Information and Knowledge Management (CIKM)*, 2007.

[11] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 1995.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 1997.

[13] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1996.

[14] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 2000.

[15] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

[16] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2008.

[17] Wei-Ning Hsu and Hsuan-Tien Lin. Active learning by learning. In *Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence*, 2015.

[18] Prateek Jain and Ashish Kapoor. Active learning for large multi-class problems. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[19] Pooyan Jamshidi, Miguel Velez, Christian Kästner, and Norbert Siegmund. Learning to sample: Exploiting similarities across environments to learn performance models for configurable systems. In *Proceedings of the 26th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2018.

[20] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2014.

[21] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 1994.

[22] Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. Mmr-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, 2006.

[23] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning active learning from data. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2017.

[24] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. *VLDB Endowment*, 2010.

[25] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the conference on Information Retrieval (SIGIR)*, 1994.

[26] Chunyuan Li, Jianqiao Li, Guoyin Wang, and Lawrence Carin. Learning to sample with adversarially learned likelihood-ratio. 2018.

[27] Lucas Maystre and Matthias Grossglauser. Just sort it! a simple and effective approach to active preference learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[28] Buyue Qian, Xiang Wang, Nan Cao, Hongfei Li, and Yu-Gang Jiang. A relative similarity based method for interactive patient risk prediction. *Data Mining and Knowledge Discovery*, 2015.

[29] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[30] Burr Settles. Active learning literature survey. 2010.

[31] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[32] Jingyu Shao, Qing Wang, and Yu Lin. Skyblocking for entity resolution. *Information Systems*, 2019.

[33] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[34] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, pages 45–66.

[35] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, 2007.

[36] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 2015.