Hierarchical Active Learning with Overlapping Regions

Zhipeng Luo
ZHL78@pitt.edu
Department of Computer Science
University of Pittsburgh
Pittsburgh, Pennsylvania

Milos Hauskrecht
milos@pitt.edu
Department of Computer Science
University of Pittsburgh
Pittsburgh, Pennsylvania

ABSTRACT

Learning of classification models from real-world data often requires substantial human effort devoted to *instance* annotation. As this process can be very time-consuming and costly, finding effective ways to reduce the annotation cost becomes critical for building such models. To address this problem we explore a new type of human feedback – *region*-based feedback. Briefly, a region is defined as a hypercubic subspace of the input data space and represents a *subpopulation* of data instances; the region's label is a human assessment of the class *proportion* of the data subpopulation. By using *learning from label proportions* algorithms one can learn instance-based classifiers from such labeled regions. In general, the key challenge is that there can be infinite many regions one can define and query in a given data space. To minimize the number and complexity of region-based queries, we propose and develop a *hierarchical active learning* solution that aims at incrementally building a *concise* hierarchy of regions. Furthermore, to avoid building a possibly class-irrelevant region hierarchy, we further propose to grow multiple different hierarchies in parallel and expand those more informative hierarchies. Through experiments on numerous data sets, we demonstrate that methods using region-based feedback can learn very good classifiers from very few and simple queries, and hence are highly effective in reducing human annotation effort needed for building classification models.

KEYWORDS

Classification; Active Learning; Learning from Label Proportions; Decision Tree Learning

1 INTRODUCTION

Learning of classification models from real-world data often requires additional human effort devoted to data annotation. Unfortunately, annotating individual instances is often labor-intensive and costly. To reduce human annotation effort, *active learning* has become one widely-applied machine learning solution. It is an interactive learning framework between learning machines and human annotators: a human iteratively provides labels for the data requested by an active learning algorithm which, in turn, builds a classifier incrementally with more labeled data. To minimize the number of labeled data, actively learning plays its role in between by requesting only those data that appear important to model refinement and learning. The efficacy of active learning has been demonstrated in numerous real-world applications [6, 23, 26, 27] as well as in theoretical work [2, 4].

Despite encouraging results of active learning research, the majority of current solutions are *instance-based*; that is, they query and learn from individual data instances. Unfortunately, instance-based approaches often come with two fundamental drawbacks. First, for more complex input spaces the number of instances one may feasibly label may still be relatively small, which can be insufficient to cover the entire data space or to represent the underlying data distribution. As a result, models induced from limited labeled data may carry significant uncertainty and bias. The second drawback is that active learning often relies on biased models to select the data instances to be labeled next; instances sampled by such models can deviate from the underlying data distribution. This is often referred to as a *sampling bias* problem [5]. Insufficient management of the sampling bias may fail the active learning process.

Besides active learning, another promising research direction that aims at saving human annotation effort is the utilization of alternative forms of efficient human feedback. This effort is complementary to active learning research. In general, human knowledge and feedback about the classification domain may take on different forms that deviate from the traditional instance-based class annotation. It is motivated by the fact that instance labeling is not always an easy, efficient way for humans to provide feedback. For example, when a physician diagnoses a patient for a possible heart disease they must peruse the patient record that consists of complex collections of lab results, symptoms, and findings. Hence, researchers have explored novel ways of soliciting easy human feedback that can guide the model building process. Examples of this research direction include *auxiliary soft-label feedback* [12–14, 26, 27], *feature feedback* [6, 17], and *group-based feedback*. The group-based feedback includes *multiple instance learning* (MIL) [1] that annotates bags of objects by indicating the presence or absence of the target class among all the instances in a bag. Another type of group-based feedback is *learning from label proportions* (LLP) [18]. It assumes class proportion statistics are given to groups of objects, estimating the proportions of the target class in each group. Both MIL and LLP are able to learn instance-based classifiers from labeled bags/groups.

To solve the annotation cost problem, we aim for efficient annotations that are easy for humans to perform but also informative for models to learn from. To do this, we study *region-based* feedback. A region is defined as a hypercubic subspace of the input data space; it is naturally described to humans by using *conjunctive patterns* that are formed by value ranges over the input features; regions can be annotated by humans using class proportion feedback, which is an estimate of the class *proportion* of the instance subpopulation represented by the region. Figure 1 illustrates this idea. The region-based annotation comes with many advantages. First, it can lead to a more efficient labeling process - annotators are capable of expressing their belief in classifying a population of instances only through one query of the region. Second, regions can better reflect the underlying data distribution and the class information.

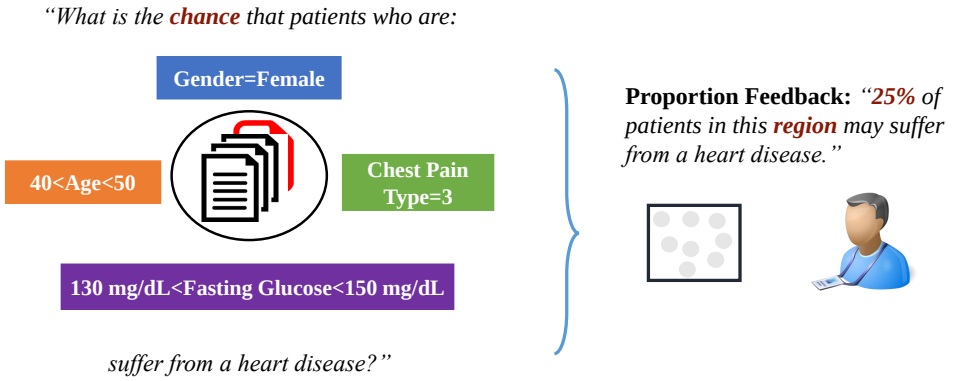

Figure 1: An example of a region-based query for the diagnosis of a heart disease [19]. A few patient instances on the left are originally recorded using 24 features. But after they are grouped, the whole group can be described by a conjunctive pattern defined using only 4 relevant features. The conjunctive patterns define a hypercubic region in the original feature space; an annotator (a physician) takes the region description and gives it a class proportion label. Individual cases are not labeled.

As we shall demonstrate, a few labeled regions are very informative to model learning. By using LLP algorithms, one can learn very accurate classifiers from a set of regions labeled with their class proportions.

In general, however, there can be infinite many regions one can define over a numeric feature space. Hence, one critical challenge would be how to identify only a small subset of regions for human annotators to label and for classifiers to learn from. Previous work has leveraged active learning heuristics to seek such informative regions. Early work [7, 19] propose to construct compact regions around the most uncertain instances. Recently, [8–11] propose to form regions in a *hierarchical* way. Their methods incrementally build a decision tree-like hierarchy of regions. The goal is to build such a hierarchy *concise* such that the leaf regions can be refined pure quickly. Although the hierarchical solution was shown to be a more fundamental way to identify informative regions, it may come with two potential limitations. First, they overly use *unsupervised* heuristics (i.e. clustering) to build a region hierarchy. However, if the structure of data is poorly aligned with the class distribution of data, the region hierarchy will end up being class-irrelevant. As a consequence, it can dramatically slow down the entire labeling and learning efficiency. Second, as a hierarchy grows deep the leaf regions are gradually described by more and more complex conjunctive patterns, which can be overwhelming for human annotators to review and assess.

In this paper, we propose and develop a new hierarchical active learning framework that can not only identify informative and simple regions efficiently, but also effectively control the clustering heuristic. Our solution is to grow *multiple* region hierarchies simultaneously and permits learning from overlapping regions. Figure 2 illustrates the idea. The motivation is intuitive - if one hierarchy turns out to be class-irrelevant then growing multiple hierarchies offers the flexibility of finding informative regions in other hierarchies. For this purpose, we need to diversify the feature combinations when constructing different hierarchies. This is in spirit to reducing the decision tree correlations in decision forest learning [3]. To evaluate our new approach, we perform a comprehensive empirical study on 16 data sets collected from `OpenML` repository [25]. Our new solution is shown having competitive performance to the state-of-the-arts methods [11, 19] in general tasks; furthermore, it dramatically outperforms others *when the structure of data is poorly aligned with the classification task.* Besides, we also show that our method uses much simpler region-based queries. Hence, our method is shown to be query-efficient, human-friendly, and robust.

2 BACKGROUND

2.1 Motivation for Region-Based Feedback

Conjunctive patterns defined with the input data features allow model builders to compactly represent instance subpopulations for querying purpose. On the human side, proportion-based feedback allows annotators to express their uncertainty when labeling grouped data. These two properties make region-based queries efficient when used for collecting human feedback in practice. As an example, consider the heart disease classification task presented in [19]. A traditional instance-based query for determining whether the patient suffers from a heart disease would cover all the feature values recorded in the data. One example query might look like: *"Consider a patient with (sex=female) ∧ (age=39) ∧ (chest pain type=3) ∧ (fasting blood sugar=150 mg/dL) ... (20 more features). Does the patient have a heart disease?"*. The label would be a binary {*true*, *false*} response. In contrast, a region-based query using conjunctive patterns could be described by a much smaller subset of relevant features that represent a subpopulation of patients. Figure 1 has illustrated one such example.

2.2 Learning from Label Proportions (LLP)

LLP deals with how to learn instance-based classifiers from a set of bags (in our scenario, regions) that are labeled with class proportions. There are two main categories of LLP algorithms: (1) [16, 18] have developed statistical models. The main idea is to use proportion labels to approximate the sufficient statistics in

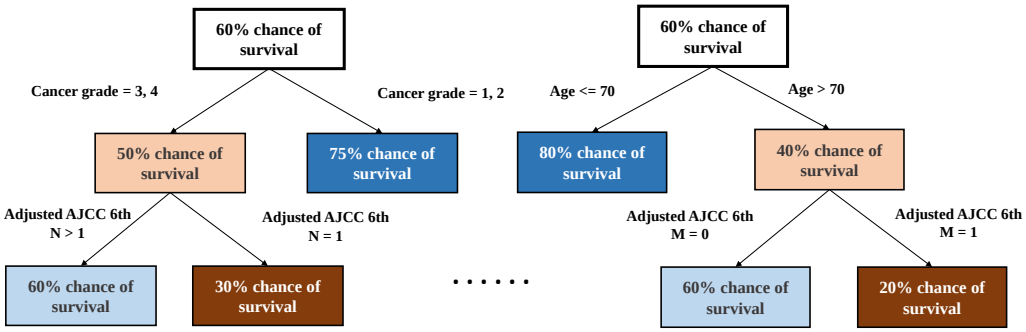

Figure 2: An illustration of our multiple-hierarchy solution HALOR applied to the survival analysis for colorectal cancer patients. The figure shows two trees that are constructed by different feature combinations. The tree on the right appears more informative than the other one.

an instance-based likelihood function; (2) [20, 28] aim at learning general classifiers. The principle is to learn a classifier well that can generate matching instance labels of which the class proportions are close to the true proportions. Moreover, [29] states that to ensure accurate classifiers induced from bags there should be sufficiently many *pure enough* bags used for training. This condition motivates the finding of pure regions in our work.

2.3 Active Learning from Labeled Regions

LLP and MIL (multiple instance learning) assume that bags naturally exist in data so they do not tackle how bags are formed or labeled. To identify informative (i.e. with high purity) regions among general data for model learning, researchers have leveraged active learning heuristics. [7, 19] first propose to construct compact regions around the most uncertain instances among an unlabeled data set. But a main problem is that such regions are intrinsically impure since instances with high uncertainty are probably distributed around the underlying decision boundary. More recently, [8–11] propose a more principled way for constructing regions. Their main idea is to incrementally build a region hierarchy, aiming at quickly refining the leaf regions pure. Their implementations are thereby named as **HAL** (Hierarchical Active learning) frameworks. The first implementation is named [9] HALG (HAL with Groups), which precompiles a hierarchy of clusters (groups) and then poses queries on the groups in a top-down manner. However, this approach is overly dominated by clustering as it only labels groups within the *fixed* hierarchy. If clustering is poorly aligned with the actual instance class distribution, the consequent hierarchy of clusters could be totally class-irrelevant. Later, they [10, 11] alleviate this problem by proposing a different approach HALR (HAL with Regions). HALR constructs a region hierarchy *dynamically* by directly splitting the entire input data space. It is similar to the decision tree learning process, but the strategy of splitting regions is determined by two heuristics. One is an *unsupervised* heuristic, which separates two clusters formed among the instances in a region; the other one is a *supervised* heuristic, which separates two classes of instances with the class labels predicted by a classifier one aims to learn. Using which heuristic in each split is determined by the quality of the classifier. When initially the classifier is immature, HALR mainly relies on the clustering heuristic to split regions. But note that the initial splits dominate the structure of the whole hierarchy. Again, if the majority of the splits are unsupervised, the consequent region hierarchy may still be class-irrelevant.

Algorithm 1 Hierarchical Active Learning Framework with Overlapping Regions (HALOR)

Input: Unlabeled data $\mathcal{U}$; Labeling budget T; # of trees K
Output: A binary classification model $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$

$\mathcal{R} \leftarrow$ initialize a root region that is unbounded;
Query the class proportion of $\mathcal{R}$;
Initialize K trees, each having a different split on $\mathcal{R}$;
Query & infer the class proportions of the child regions;
$L \leftarrow$ all the child regions of the K trees;
$t \leftarrow K + 1$ (the # of queries consumed);
repeat
  Learn the base model $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$ from L;
  Identify the most informative split (d_*, v_*, R_*);
  Split R_* from value v_* at dimension d_*;
  Query & infer the class proportions of sub-regions;
  $L \leftarrow \{L - R_*\} \cup \{R_*$'s sub-regions$\}$;
  $t \leftarrow t + 1$;
until $t = T$, i.e. the budget is used up
return $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(T)})$

Our work proposes a more robust solution **HALOR** (HAL with Overlapping Regions) that grows multiple region hierarchies and permits learning with overlapping regions. There are two essential improvements. The first one is robustness: HALOR can effectively reduce the side effects of the clustering heuristic and guarantees finding informative regions among multiple hierarchies. The second improvement is on the query complexity of regions: because of building multiple shallow hierarchies, HALOR queries the leaf regions that are defined by less complex conjunctive patterns. Simple region-based queries can greatly ease the annotation process.

3 METHODOLOGY

In this section, we detail the implementation of our multiple-tree based approach, named HALOR, and the key steps are summarized in Algorithm 1. The essential idea is to grow multiple hierarchies

(binary trees) simultaneously to allow exploring more informative regions in different trees. In the following, we will first present how to build one tree (i.e. the number of trees $K = 1$), and then present how to build multiple trees ($K > 1$).

3.1 Preliminaries

Our goal is to learn an instance-based binary classification model $f : X \rightarrow Y = \{0, 1\}$. $X \subset \mathbf{R}^m$ is the input space and it consists of m features $\{d_1, ..., d_m\}$. The features can be numeric, nominal, and categorical. We collect a pool of unlabeled data $\mathcal{U} = \{\boldsymbol{x}_i | \boldsymbol{x}_i \in X, 1 \leq i \leq n\}$ for training, and we assume the data have been properly normalized and encoded beforehand. Without loss of generality, we proceed with the learning of a binary probabilistic model $P(y|\boldsymbol{x}; \boldsymbol{\theta})$ ($\boldsymbol{\theta}$ is the model parameter) and we refer to it as our *base* model.

A region R is a hypercubic subspace of the input space, defined as a triplet $R = (C, D, \mu)$. C is the region description by using conjunctive patterns formed by value ranges over the input features:

$$(v_{1,min} < d_1 < v_{1,max}) \wedge ... \wedge (v_{i,min} < d_i < v_{i,max}) \wedge ...$$

Absence of any feature (or value) in conjunctive patterns indicates unbounded values on that feature dimension. For example, in Figure 2, the bottom-right region on the right tree is described as "*Patient population: Age>70* $\wedge$ *Adjusted AJCC 6th M=1*", and the other feature values are unbounded. $D \subset \mathcal{U}$ is the fraction of empirical data that are satisfied by the conjunctive patterns C. μ is the class proportion of the instance population within R, annotated by a human. By assuming that instances in $\mathcal{U}$ are i.i.d. sampled according to the underlying distribution of $p(\boldsymbol{x})$, and that there are sufficiently many of them, we assume that the class proportion of the empirical instances D is also equal to μ.

3.2 Actively Building One Region Hierarchy

Building a single hierarchy of regions is similar to the standard decision tree learning process. We initialize the region hierarchy from a root region $\mathcal{R}$ that is totally unbounded. $\mathcal{R}$ also covers all the empirical data $\mathcal{U}$. Its class proportion is the prior class distribution $p(y = 1)$, annotated by a human. With such a 1-node tree defined, we incrementally grow the tree by repeatedly performing a rectangular, binary split on one of the leaf regions. Formally, suppose at each time t (i.e. the number of queries consumed so far) there are N leaf regions (all labeled) in a fringe $L = \{R_i | 1 \leq i \leq N\}$, our procedure aims at finding a triplet (d_*, v_*, R_*), meaning a split made on region $R_* \in L$ from value v_* at feature d_*. After the split, we solicit the class proportion of either sub-region from human annotators and then infer the proportion of the other. The label inference is valid because the total proportion remains unchanged during the split. Finally, we expand the fringe $L \leftarrow \{L - R_*\} \cup \{R_*\text{'s sub-regions}\}$, increment $t \leftarrow t + 1$, and re-train the base model with the new L (training to be presented in Section 3.4).

Key to our procedure is determining which R_i should be split and how to split it. Recall that in decision tree learning, the quality of splits is measured by information gain. That is, suppose one valid split (d, v) splits a region R_i into two sub-regions R^l and R^r. Then, the information gain of this split is defined as:

$$G(d, v, R_i) = I(\mu_i) - \frac{n^l}{n_i} I(\mu^l) - \frac{n^r}{n_i} I(\mu^r) \tag{1}$$

where $I(\mu) = 2\mu(1-\mu)$ is Gini-Index measurement of class entropy; μ_i, μ^l, μ^r are the class proportions of the empirical instances in R_i, R^l, R^r; n_i, n^l, n^r are the number of instances, respectively. The best split is the one that maximizes the information gain.

The reason of using Gini-Index to measure the class entropy is because it reflects the expected error when one attempts to *guess* the actual instance labels in a region R_i with label μ_i. More specifically:

(1) For each instance in R_i, sample its label as an independent Bernoulli process with the parameter = μ_i. This creates n_i sampled labels;
(2) Calculate the distribution of w_i, i.e. the number of mismatches between the sampled labels and the true labels. Although the true labels are unknown, each true label can be assumed to follow an independent Bernoulli distribution with parameter = μ_i. Therefore, the probability of mismatch for each instance also follows in independent Bernoulli distribution with parameter = $P(mismatch) = P[false\ positive] + P[false\ negative] = 2\mu_i(1-\mu_i)$. Then apparently w_i follows a Binomial distribution $Bin(n_i, 2\mu_i(1-\mu_i))$;
(3) Finally, $\mathbb{E}(w_i) = 2\mu_i(1-\mu_i)n_i$ is the expected error of guessing all the instance labels in R_i.

In a decision tree learning process, the gain is easily calculated by using the training instance labels. However, in our process we do not acquire individual data labels, thus unable to compute μ^l or μ^r. Therefore, we develop two heuristics to approximate the gain. The first one is termed *supervised* heuristic that uses the base classification model. Ideally, an accurate enough model can foresee the information gain of every candidate split by simply inferring the class proportion of any possible sub-region. Formally, suppose the base model is learned as $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$ from L, then the class proportion of any sub-region R^s ($s = l$ or r) can be inferred as:

$$\hat{\mu}^s = \frac{1}{n^s} \sum_{j=1}^{n^s} P(y_j = 1 | \boldsymbol{x}_j; \hat{\boldsymbol{\theta}}) \tag{2}$$

Hence, Eq. 1 can be approximated by the base model as:

$$G_s(d, v, R_i) = I(\mu_i) - \frac{n^l}{n_i} I(\hat{\mu}^l) - \frac{n^r}{n_i} I(\hat{\mu}^r) \tag{3}$$

Nevertheless, the base model is barely accurate initially when the supervision (i.e. human annotation) is little, and thus the supervised gain defined in Eq. 3 is unreliable. Therefore, in order to calibrate the gain calculation, we also consider another approximation - *unsupervised* gain. The essential idea is to identify two clusters in the data within a region and then find the best split that separates the two clusters apart. It is a sensible heuristic that is often used in semi-supervised learning and clustering-based active learning [5, 22, 24]. The reasoning behind is that clustering is usually assigned with the classification task, and thus the structure of data could more or less reflect the underlying class distribution. To implement an unsupervised split, we first perform a 2-means probabilistic clustering on the instances D_i within a region R_i. Then each instance will have an unsupervised probabilistic label indicating the chance of belonging to either cluster. Given these instance labels, we compute the unsupervised gain as:

$$G_u(d, v, R_i) = I(\tilde{\mu}_i) - \frac{n^l}{n_i} I(\tilde{\mu}^l) - \frac{n^r}{n_i} I(\tilde{\mu}^r) \tag{4}$$

where $\tilde{\mu}_i$, $\tilde{\mu}^l$, $\tilde{\mu}^r$ are the *cluster* proportions of the parent region R_i and the two sub-regions, respectively.

Now, with supervised gain (Eq. 3) and unsupervised gain (Eq. 4) defined above, we need to combine them and balance their effects. The principle is to weight the supervised gain more when the base model becomes more accurate and makes more reliable inference. Hence, we propose using a weighted sum as the final gain of a split (d, v) made on a region R_i:

$$G_{s+u}(d, v, R_i) = \alpha_s^{(t)} G_s(d, v, R_i) + (1 - \alpha_s^{(t)}) G_u(d, v, R_i) \quad (5)$$

where $\alpha_s^{(t)} \in [0, 1]$ is the weight of supervised gain. Intuitively, $\alpha_s^{(t)}$ should be small initially when the model is poor and then gradually increases. To this end, we set $\alpha_s^{(t)} \propto [1/\epsilon^{(t)}]^2$ where $\epsilon^{(t)}$ is the training error of fitting the base model to the labeled regions in L:

$$\epsilon^{(t)} = \sum_{i=1}^{N} \frac{n_i}{n} |\hat{\mu}_i - \mu_i| \quad (6)$$

where μ_i is the true class proportion of R_i and $\hat{\mu}_i$ is the inferred proportion given by the base model (Eq. 2); $n = \sum_{i=1}^{N} n_i = |\mathcal{U}|$ is the number of all the instances. Using the inverse of $\epsilon^{(t)}$ at order 2 is because the $I(\mu) = 2\mu(1 - \mu)$ is of order 2 on μ. Finally, we determine the most informative split as follows:

$$(d_*, v_*, R_*) = \arg \max_{(d,v)\ splits\ R_i \in L} \{\frac{n_i}{n} \cdot G_{s+u}(d, v, R_i)\} \quad (7)$$

The gain is further weighted by region size because we also prefer to split a larger region so as to impose a higher impact on model update.

Remarks. Compared to the latest work [11] in terms of building a single hierarchy, we have two improvements. Firstly, we have designed a one-step procedure to explicitly identify the most informative split (Eq. 7). The previous work, however, finds the best split only within a most uncertain region. However, high uncertainty in the parent region does not necessarily guarantee any informative split made in the end, in that the child regions could also be equally uncertain. Secondly, the previous work treats the two splitting heuristics independently and employs a Bandit algorithm for selecting which heuristic to use. However, Bandit algorithms require exploration steps that may waste the initial queries on the supervised heuristic that is yet known to be poor in the beginning. Our solution, by comparison, combines the two heuristics into one formula (Eq. 5) and nicely balances them by using a weight $\alpha_s^{(t)}$ that is directly associated with the current model performance.

3.3 Building Multiple Region Hierarchies

There are two limitations of growing a single hierarchy. Firstly, the initial splits made on the root region $\mathcal{R}$ are mainly determined by the unsupervised heuristic. A possible failure case is that clustering may always dominate the splits and consequently, builds a class-irrelevant tree. Secondly, the query complexity of regions increases rapidly as the tree grows deeper. To alleviate these issues, we now present a robust multiple-tree solution. The whole process resembles building a single tree, but additionally, we need to control the similarity among the regions appearing in different trees.

To start, we initialize a small number of one-split trees ($K \approx 2, 3$). They all start with the same root region $\mathcal{R}$, but each has a split on a different feature dimension. These initial K splits are done unsupervisedly. That is, we first perform 2-means clustering on all the instances in $\mathcal{R}$, and then find the top-K features, along with their best splits, that can separate the two clusters the most (Eq. 4). After each split, we query or infer the class proportions of the sub-regions and put them into a fringe L. Therefore, the initialization phase outputs K one-split trees, totally $2K$ labeled regions in L, and $K + 1$ queries consumed (1 query for the root region $\mathcal{R}$). After initialization, similarly as before we incrementally grow the trees by repeatedly splitting one leaf region in L. The most informative split identified by Eq. 7 can be directly applied here, but with a pre-check named **region deduplication procedure** that prevents generating duplicate regions that already exist in L. Duplicate regions are defined as follows:

Definition 3.1. Given two regions $R_1 = (C_1, D_1, \mu_1)$ and $R_2 = (C_2, D_2, \mu_2)$. Denote by F_1, F_2 the set of unique features that are used in conjunctive patterns C_1, C_2. Denote by $D = D_1 \cap D_2$ the intersection of data instances $D1$, $D2$. If (1) $F_1 \equiv F_2$ and (2) either $|D|/|D_1|$ or $|D|/|D_2|$ is greater than a threshold $\gamma \in [0, 1]$, then R_1 and R_2 are duplicate.

Whenever executing Eq. 7, we first apply the deduplication procedure that discards any invalid split (d, v) if it generates at least one sub-region that is duplicate to any existing region in L. An efficient implementation of this procedure is to first select out the valid features and the associated split values, and then only compute gains for these valid splits.

3.4 Learning a Model from Labeled Regions

The last part remained is to explain how we employ a general LLP (learning from label proportion) algorithm to learn the base classification model $P(y|\boldsymbol{x}; \boldsymbol{\theta})$ from labeled regions L. The main idea is to learn the model properly such that it can generate matching instance labels of which the class proportions are close to the true proportions. Suppose at a time t there are N labeled regions in L that are the leaf regions from all region hierarchies. Each region R_i contains n_i instances and has a proportion label μ_i. Denote by $\hat{\mu}_i$ the estimated proportion given by the base model (Eq. 2) and $n = \sum_{i=1}^{N} n_i$. Then, the loss function for model learning can be defined as:

$$\mathcal{L}(L; \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N} \frac{n_i}{n} (\hat{\mu}_i - \mu_i)^2 + \lambda \mathbf{R}(\boldsymbol{\theta}) \quad (8)$$

where $\mathbf{R}(\boldsymbol{\theta})$ is the regularization term penalizing model complexity. Particular to this paper, we use L_2 penalty: $\mathbf{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$. Finding of the best parameter $\hat{\boldsymbol{\theta}}$ can be worked out by a standard optimization program. We use a gradient-based optimization approach. The loss function defined above is trained by minimizing a squared loss via L-BFGS [15].

4 EXPERIMENTS

We conduct a comprehensive study to empirically evaluate our approach HALOR and its variants. The purpose is to see (1) how efficiently (in terms of number of queries) that HALOR can learn

Table 1: 16 binary classifications data sets.

Dataset	# of Data	# of Features	Major Class%	Feature Type	About
musk	6598	**167**	84.59%	Numeric, Ordinal, Categorical	Molecules classification
satellite	5100	36	**98.53**%	Numeric	Satellite images classification
bank	45211	16	88.30%	Numeric, Ordinal, Categorical	Financial accounts classification
ozone	2534	**72**	**93.69**%	Numeric	Ozone levels detection
kc1	2109	21	84.54%	Numeric	NASA software fraud detection
pc1	1109	21	**93.06**%	Numeric	NASA software fraud detection
wdbc	569	30	62.74%	Numeric	Breast cancer images
eye	14980	14	55.12%	Numeric	Eye states recognition
ilpd	583	10	71.35%	Numeric, Categorical	Indian liver patients
biodeg	1055	41	66.26%	Numeric	Chemicals classification
phishing	11055	30	55.69%	Ordinal, Categorical	Phishing websites detection
nomao	34465	**118**	71.44%	Numeric, Ordinal, Categorical	Places deduplication
climate	540	20	**91.48**%	Numeric	Climate model prediction
hill-valley	1212	**100**	50.00%	Numeric	Hill valley recognition
click	39948	9	83.16%	Numeric	Web ad clicks prediction
telescope	19020	10	64.84%	Numeric	Images from Gamma telescope

classification models, and (2) how complex the region queries used by HALOR can be.

4.1 Data Sets

We collect 16 the most influential data sets that are ranked by `OpenML` machine learning repository [25]. They come from a variety of real-life fields. Table 1 summarizes the basic information and statistics. Those sets with high-dimensional feature spaces or with unbalanced class distribution are marked bold.

4.2 Methods Compared

We compare our solution HALOR-K (K trees) to 5 other representative active learning approaches: two instance-based methods that also use clustering heuristic: HS (Hierarchical Sampling) [5] and DWUS (Density-Weighted Uncertainty Sampling) [21]; three region-based methods: RIQY [19], HALG [9] and HALR [11].

(1) **HS** performs hierarchical clustering on a pool of unlabeled data and then relies on the hierarchy to select instances. By assuming that the hierarchy structure is well aligned with the actual class labels, HS only queries instances from those impure clusters that are (estimated) of high class entropy.

(2) **DWUS** is a conventional active learning approach. Its strategy is to query instances that are not only uncertain but also representative of other unlabeled data.

(3) **RIQY** is the first active learning work that uses region-based queries. The framework builds upon DWUS. The difference is that RIQY's query is made to a region which is formed upon a compact neighbor of $\boldsymbol{x}^*$ that is suggested by the DWUS algorithm. However, the regions found by RIQY are often impure because the neighbor around an uncertain $\boldsymbol{x}^*$ are probably impure as well.

(4) To overcome the issues of RIQY, **HALG** and **HALR** are recent works that propose to form and query regions hierarchically. HALG, like HS, compiles a fixed hierarchy of clusters first and then queries regions formed by the clusters. HALR, however, constructs regions dynamically and reduces the bias brought by the clustering heuristic. This promotes the learning efficiency of HALR, which is the state-of-the-arts.

4.3 Experimental Settings

We split each data set into three disjoint parts: the initial labeled dataset (about 1%-2% of all available data), a test dataset (about 25% of data) and an unlabeled dataset $\mathcal{U}$ (the rest) used as training data. Note that only HS, DWUS, and RIQY require the initial labeled data to start training while other HAL methods do not. To simulate region proportion labels from human annotators, we use empirically labeled instances to do so. To label a region, we count the instances that fall into the region and report their class label fraction as the class proportion. Note that the instance labels are only used to generate region labels, never exposed to region construction or learning algorithm. Such a simulation method has been frequently used in RIQY and LLP studies. We evaluate all the methods based on two metrics: (1) model performance measured by AUC (the Area Under the Receiver Operating Characteristic curve) and (2) query complexity (i.e. how many features have been used in a query). We adopt Logistic Regression as the base model. In the following, we will show plots of AUC score and query complexity against the number of queries $t < 200$. To best visualize each plot, we omit the remaining tails of curves after most methods have converged. To reduce the experiment randomness, all results are averaged over 20 runs with different training/test data splits. Each method's hyper-parameters have been fine-tuned using cross-validation. Our HALOR method uses $\gamma = 0.1$, and the regularization parameter λ in LLP learning is chosen mildly between $[10^{-5}, 10^{-3}]$.

4.4 Model Performance Results

The main results are shown in Figure 3, and the rankings of all methods are summarized in Table 2. Overall, HALR and HALOR are leading the best performance; RIQY, DWUS and HALG follow; HS ranks last. Below are some general observations:

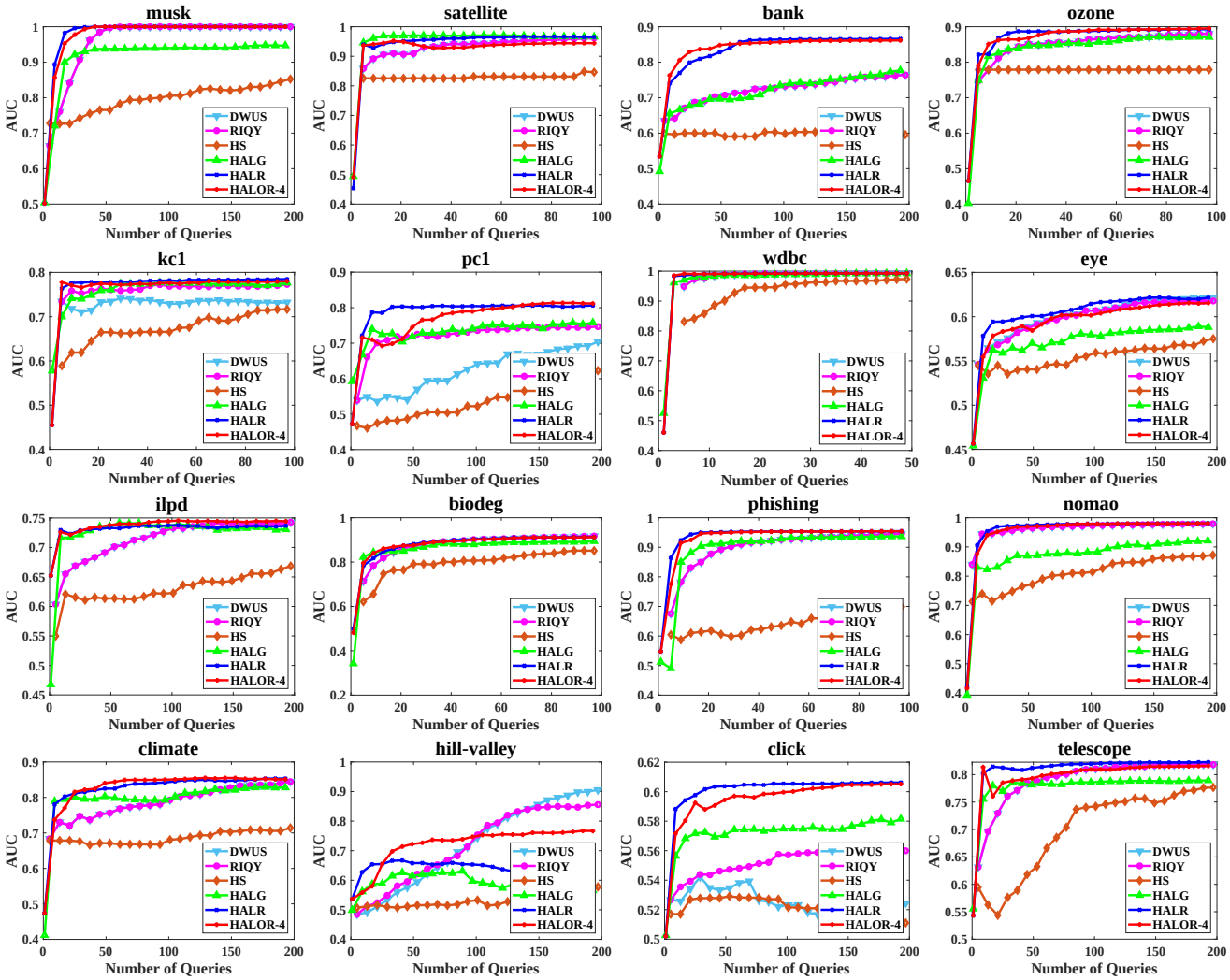

Figure 3: The AUC performance of all methods tested on 16 data sets.

(1) Region-based methods (RIQY, HALs) vs. instance-based methods (HS, DWUS). We observe that initially when only a very few queries are available (<10 queries), learning with regions outputs better models than learning with instances. The reason is that one general region with class proportion is usually more class-informative than one labeled instance. Later as more queries are made, purer regions emerge quickly and accelerate the model convergence. Such observations demonstrate the advantages of using regions for active querying and learning.

(2) Among all the region-based methods (RIQY, HALs), we see that HAL frameworks, especially HALR and HALOR, generally perform better. This lends credence to the fact that HALs construct more informative regions than RIQY.

(3) Finally, among the three HAL frameworks, HALR and HALOR perform better than HALG. It is because HALR and HALOR construct regions *dynamically* and they can better control the bias brought by clustering heuristic. Furthermore, comparing HALR and our approach HALOR, we see that HALR performs slightly better. The probable reason is because in many datasets the structure of data is more or less aligned with their class distributions, and therefore, building one tree guided by clustering heuristic (HALR) would be no harm to those datasets. Nevertheless, we must point out that HALR does fail on data set `hill-valley`, where clustering is totally irreverent to the actual class distribution. Then, HALR has no remedy for such situation because a single tree has been overly dominated by the clustering heuristic. In contrast, our solution HALOR has effectively mitigated this problem by growing multiple trees. If one tree fails, we can always build new trees to find more class-relevant regions. Therefore, HALOR is fundamentally more robust. Well, the side effect of growing multiple trees is that it incurs a slight performance drop initially, in that it needs queries to explore new trees. But we also see HALOR then catches up rapidly and can potentially outperform HALR.

Table 2: Ranking of all methods on all data sets.

Dataset	HS	DWUS	RIQY	HALG	HALR	HALOR
musk	4	2	2	3	1	1
satellite	3	2	2	1	1	1
bank	3	2	2	2	1	1
ozone	3	2	2	2	1	1
kc1	3	2	2	2	1	1
pc1	5	4	3	3	1	2
wdbc	2	1	1	1	1	1
eye	4	2	2	3	1	2
ilpd	3	2	2	1	1	1
biodeg	2	1	1	1	1	1
phishing	4	3	3	2	1	1
nomao	3	1	1	2	1	1
climate	4	3	3	2	1	1
hill-valley	6	1	2	5	4	3
click	6	5	4	3	1	2
telescope	5	3	3	4	1	2
Average	**3.72**	**2.11**	**2.06**	**2.33**	**1.22**	**1.39**
Std	**1.45**	**1.10**	**0.84**	**1.05**	**0.71**	**0.59**

4.5 Query Complexity Results

Another important measurement closely related to annotation cost is the complexity of queries. In general, high-dimension instances described by numerous features can overwhelm annotators. By comparison, well-built region-based queries using much fewer features can be easier for annotators to review and assess. To measure this aspect, we define the complexity of a region query as:

$$\text{Complexity}(query) = \frac{\#\text{ of features used in the } query}{\text{total \# of all features}} \quad (9)$$

According to the definition above, the complexity of a query can range from 0 to 1. Being 1 means using up all the features, while close to 0 means using very few features in the query. Figure 4 shows the query complexities of all region-based methods. We omit instance-based methods since instance queries use all the features so their query complexity is constantly 1. Overall, our method HALOR uses the least number of features in its queries. For high-dimensional data sets such as `musk` (167 features), `nomao` (118) and `hill-valley` (100), HALOR needs merely a few features (<5%) to describe the regions, and such a property would make the labeling process a lot easier. HALR uses more complex queries because it grows only one tree and the query complexity increases more with the depth of the tree. For HALG, we see more variations because the region descriptions are automatically learned by a rule inducer which may return imperfect and more complex regions. Lastly, we see RIQY's curves are very close to 1. It is because RIQY tends to construct very small and compact regions around specific instances, and thus it leads to induce regions of high complexity defined by many conditions (features).

4.6 Effects of Number of Trees in HALOR

The hyper-parameter K in HALOR-K approach determines how many trees are grown in parallel. Figures 5 and 6 show how the model performance and query complexity vary with K. To save space, we only plot the results for the top 8 data sets where the performance of different variants of HALOR differ the most (the remaining 8 datasets show only little difference with different K). The figures and results therein illustrate a couple of trade-offs in HALOR. First, for query complexity, apparently, growing more trees in HALOR translates into simpler region and hence simpler queries. In terms of model performance, it initially takes more queries for HALOR to explore the feature information for larger K. This incurs performance drop in the initial (exploration) phase. However, growing more trees has a higher potential of improving model performance later. The results further suggest that usually 2 or 3 trees in HALOR would be good enough to learn models well from the trees. See the performance boosting from HALR to HALOR-2 on `hill-valley`.

5 CONCLUSIONS

We study a novel hierarchical active learning (HAL) framework that works with region-based queries. We develop a robust implementation of HAL that is to grow multiple region hierarchies simultaneously. The essential benefits are: (1) it allows identifying more informative splits among multiple difference trees; (2) it reduces the query complexity of regions and thus makes human annotation easier. Based on large number of results, we conclude that HAL is a label-efficient framework for general binary classification problems. Our multi-tree solution HALOR is shown to be more robust than previous single-tree based implementations. Therefore, HALOR is effective in learning classification models while consuming very little and simple human feedback.

ACKNOWLEDGMENTS

The work presented was supported by NIH grant R01GM088224. The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of NIH.

REFERENCES

[1] Jaume Amores. 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence* 201 (2013), 81–105.
[2] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. 2009. Agnostic active learning. *J. Comput. System Sci.* 75, 1 (2009), 78–89.
[3] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[4] Sanjoy Dasgupta. 2011. Two faces of active learning. *Theoretical computer science* 412, 19 (2011), 1767–1781.
[5] Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *Proceedings of the 25th ICML*. ACM, 208–215.
[6] Gregory Druck, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 81–90.
[7] Jun Du and Charles X Ling. 2010. Asking generalized queries to domain experts to improve learning. *Knowledge and Data Engineering, IEEE Transactions* 22, 6 (2010), 812–825.
[8] Zhipeng Luo and Milos Hauskrecht. 2017. Active learning of classification models from soft-labeled groups. In *Advances in Neural Information Processing Systems, Learning from Limited Data Workshop*.
[9] Zhipeng Luo and Milos Hauskrecht. 2018. Hierarchical Active Learning with Group Proportion Feedback. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI'18*. 2532–2538. https://doi.org/10.24963/ijcai.2018/351
[10] Zhipeng Luo and Milos Hauskrecht. 2018. Hierarchical active learning with proportion feedback on regions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 464–480.
[11] Zhipeng Luo and Milos Hauskrecht. 2019. Region-Based Active Learning with Hierarchical and Adaptive Region Construction. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 441–449.

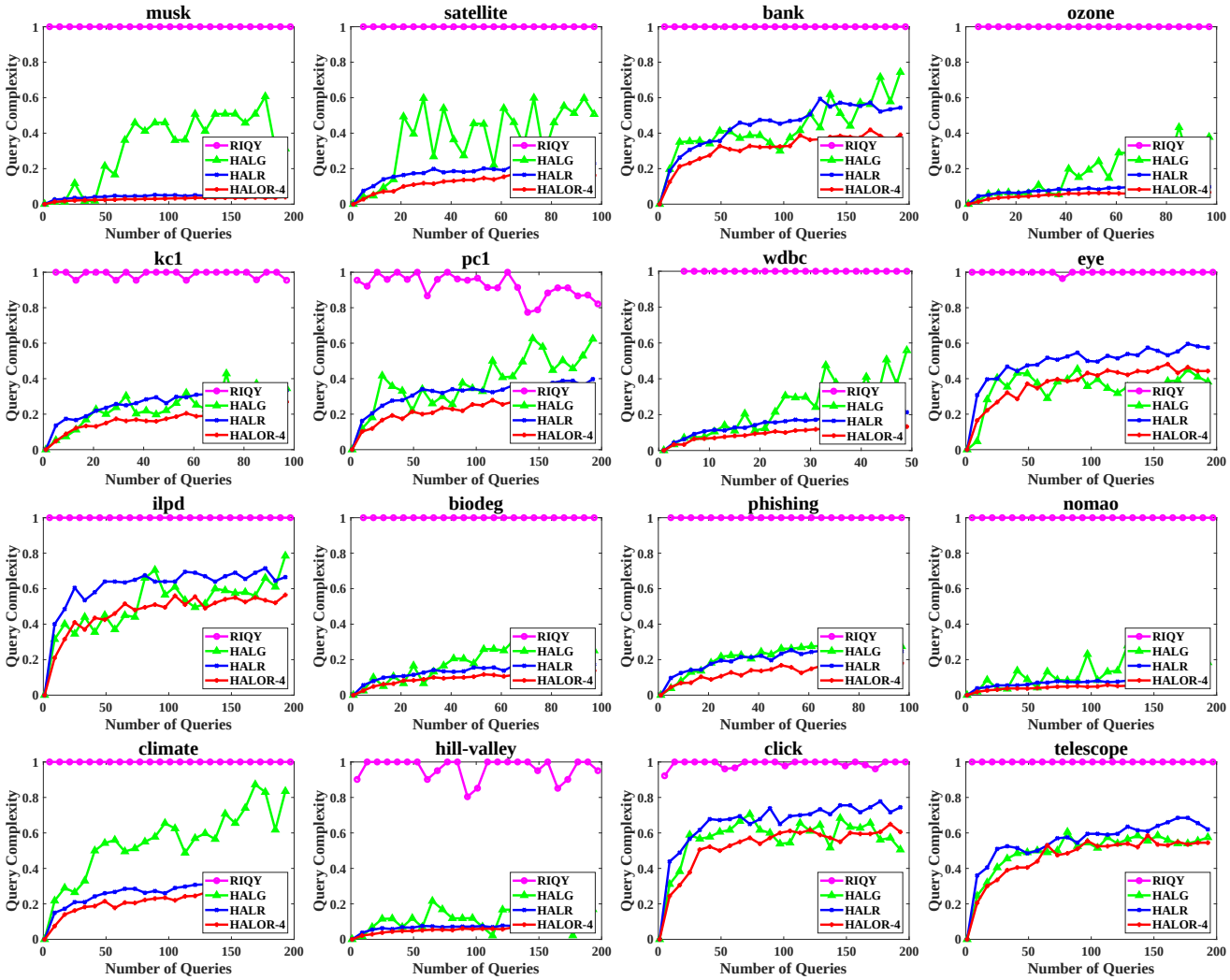

Figure 4: Query complexity of region-based methods.

[12] Quang Nguyen, Hamed Valizadegan, and Milos Hauskrecht. 2011. Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 477–486.

[13] Quang Nguyen, Hamed Valizadegan, and Milos Hauskrecht. 2014. Learning classification models with soft-label information. *Journal of the American Medical Informatics Association* 21, 3 (2014), 501–508.

[14] Quang Nguyen, Hamed Valizadegan, Amy Seybert, and Milos Hauskrecht. 2011. Sample-efficient learning with auxiliary class-label information. In *AMIA Annual Symposium Proceedings*, Vol. 2011. American Medical Informatics Association, 1004.

[15] Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

[16] Giorgio Patrini, Richard Nock, Paul Rivera, and Tiberio Caetano. 2014. (Almost) no label no cry. In *NIPS*. 190–198.

[17] Stefanos Poulis and Sanjoy Dasgupta. 2017. Learning with feature feedback: from theory to practice. In *Artificial Intelligence and Statistics*. 1104–1113.

[18] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. 2009. Estimating labels from label proportions. *Journal of Machine Learning Research* 10, Oct (2009), 2349–2374.

[19] Parisa Rashidi and Diane J Cook. 2011. Ask me better questions: active learning queries based on rule induction. In *Proceedings of the 17th ACM SIGKDD*. ACM, 904–912.

[20] Stefan Rueping. 2010. SVM classifier estimation from group probabilities. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 911–918.

[21] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.

[22] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 1070–1079.

[23] Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in neural information processing systems*. 1289–1296.

[24] Ruth Urner, Sharon Wulff, and Shai Ben-David. 2013. PLAL: Cluster-based active learning. In *Conference on Learning Theory*. 376–397.

[25] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[26] Yanbing Xue and Milos Hauskrecht. 2017. Active Learning of Classification Models with Likert-Scale Feedback. In *SIAM Data Mining Conference, 2017*. SIAM.

[27] Yanbing Xue and Milos Hauskrecht. 2019. Active learning of multi-class classification models from ordered class sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5589–5596.

[28] Felix Yu, Dong Liu, Sanjiv Kumar, Jebara Tony, and Shih-Fu Chang. 2013. ∝SVM for Learning with Label Proportions. In *ICML*. 504–512.

[29] Felix X Yu, Krzysztof Choromanski, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. 2014. On learning from label proportions. *arXiv preprint arXiv:1402.5902* (2014).

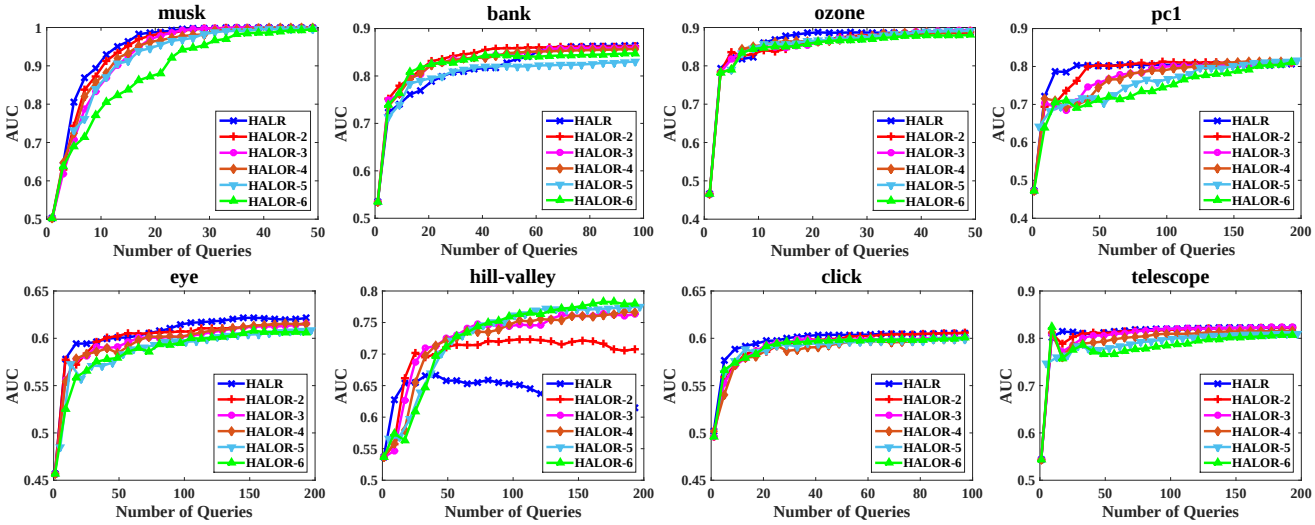

Figure 5: Model performance of HALOR with various number of trees.

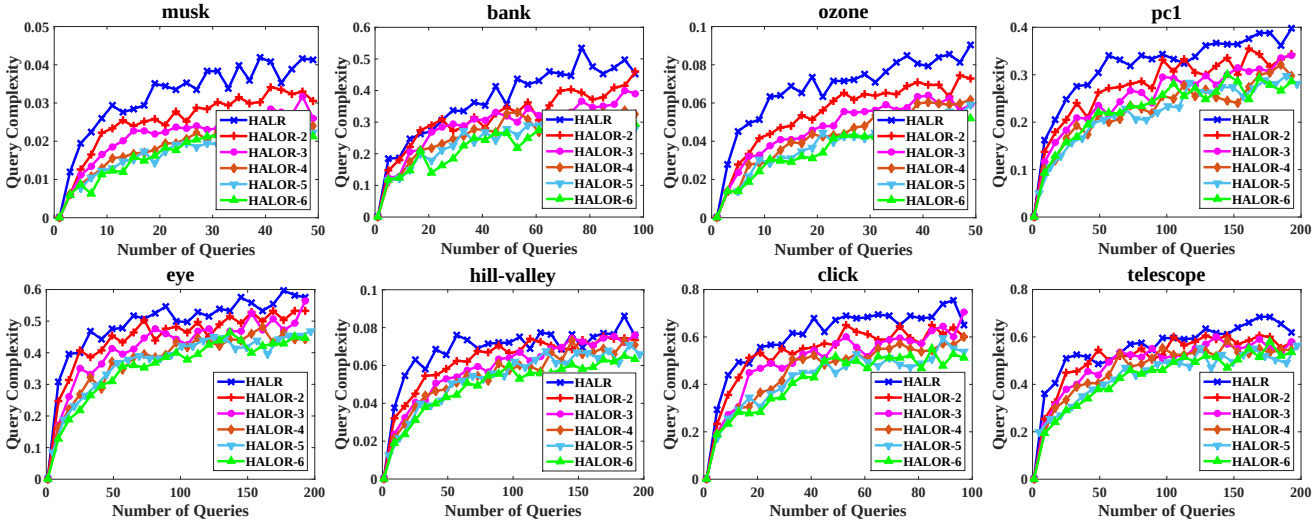

Figure 6: Query complexity of HALOR with various number of trees.