Learning of Classification Models from Group-Based Feedback

by

Zhipeng Luo

BS, Beihang University, Beijing, China, 2013

Submitted to the Graduate Faculty of

the Department of Computer Science in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTER SCIENCE

This dissertation was presented

by

Zhipeng Luo

It was defended on

August 11, 2020

and approved by

Milos Hauskrecht, PhD, Professor, Department of Computer Science

Rebecca Hwa, PhD, Professor, Department of Computer Science

Adriana Kovashka, PhD, Assistant Professor, Department of Computer Science

Zhao Ren, PhD, Assistant Professor, Department of Statistics

Dissertation Director: **Milos Hauskrecht**, PhD, Professor, Department of Computer Science

Learning of Classification Models from Group-Based Feedback

Zhipeng Luo, PhD

University of Pittsburgh, 2020

Learning of classification models in practice often relies on a nontrivial amount of human annotation effort. The most widely adopted human labeling process assigns class labels to individual data instances. However, such a process is very rigid and may end up being very time-consuming and costly to conduct in practice. Finding more effective ways to reduce human annotation effort has become critical for building machine learning systems that require human feedback.

In this thesis, we propose and investigate a new machine learning approach - **Group-Based Active Learning** - to learn classification models from limited human feedback. A group is defined by a set of instances represented by conjunctive patterns that are value ranges over the input features. Such conjunctive patterns define hypercubic *regions* of the input data space. A human annotator assesses the group solely based on its region-based description by providing an estimate of the *class proportion* for the subpopulation covered by the region. The advantage of this labeling process is that it allows a human to label many instances at the same time, which can, in turn, improve the labeling efficiency.

In general, there are infinitely many regions one can define over a real-valued input space. To identify and label groups/regions important for classification learning, we propose and develop a **Hierarchical Active Learning** framework that actively builds and labels a hierarchy of input regions. Briefly, our framework starts by identifying general regions covering substantial portions of the input data space. After that, it progressively splits the regions into smaller and smaller sub-regions and also acquires class proportion labels for the new regions. The proportion labels for these regions are used to gradually improve and refine a classification model induced by the regions. We develop three versions of the idea. The first two versions aim to build a single hierarchy of regions. One builds it statically using hierarchical clustering, while the other one builds it dynamically, similarly to the decision tree learning process. The third approach builds multiple hierarchies simultaneously, and it

offers additional flexibility for identifying more informative and simpler regions. We have conducted comprehensive empirical studies to evaluate our framework. The results show that the methods based on the region-based active learning can learn very good classifiers from a very few and simple region queries, and hence are promising for reducing human annotation effort needed for building a variety of classification models.

Keywords: Active Learning, Learning from Label Proportions, Weakly Supervised Learning, Decision Tree Learning.

Table of Contents

List of Tables

List of Figures

Preface

I spent six and a half wonderful years in Pittsburgh to pursue my Ph.D. This journey was like a marathon for me that would not be finished without the help, support, and company of many people. I would like to take this opportunity to express my most sincere gratitude and respect to all of them.

First of all, I want to thank my advisor Professor Milos Hauskrecht. Milos led me to the world of machine learning and since then has provided continuous support to my research. Milos not only taught me specific machine learning techniques and skills but more importantly, guided me through the scientific research process and deeply influenced my way of thinking. His high professional standards helped me successfully solve many real-world problems, and thus we finally published many fruitful results in highly prestigious forums. Milos is also a warm guy and he has given me a ton of encouragement. I will never forget the moment back in 2017 when I had to set up my homepage which, however, had zero publication (our first paper got rejected many times before it was accepted to IJCAI'18). I felt quite awkward about it but Milos encouraged me by saying "Patrick, don't worry, we will fill it up!" Finally, we made it! Thank you, Milos!

Apart from my research, I was also fortunate to work on a healthcare-related project that applied machine learning techniques to monitor patient management decisions in intensive care unit (ICU) (funded by NIH grants R01GM088224 and R01LM010019). This project was led by Milos, Dr. Gilles Clermont who is an experienced Critical Care Medicine physician, and Dr. Gregory Cooper who is an accomplished and highly respected professor and researcher in biomedical informatics. It was a unique experience to work with all of them and I have gained valuable and tremendous knowledge about the combination of AI and healthcare. I feel very lucky.

I would also like to thank my thesis committee members Dr. Adriana Kovashka, Dr. Rebecca Hwa, and Dr. Zhao Ren for their insightful discussions and feedback during my thesis proposal and defense. Dr. Ren is an assistant professor from the Department of Statistics at Pitt, and I knew him by taking his course Asymptotic Analysis which was

awesome. I wish I could have known him earlier so that I can learn more about statistical machine learning from him. I also want to thank Dr. Daniel Jiang who is an assistant professor from the Industrial Engineering Department at Pitt. Dr. Jiang almost served on my committee. He was such a humble and rigorous person and I knew him by taking his Reinforcement Learning course which was awesome as well.

I was also fortunate to work with several lab mates on Milos' clinical outlier monitoring and alerting project: Charmgil Hong, Salim Malakouti, Siqi Liu, and Matt Barren. I also want to thank Yanbing Xue and Jeong-Min Lee for their insightful discussions about my research. They are all excellent guys, you know, with very different "shapes", and I wish all my best blessings to their future endeavors.

Besides, I would like to give special thanks to Professor Daniel Mosse who admitted me to Pitt Computer Science and then became my initial advisor for the first two years. Daniel was the Chair of the department at that time and he gave me tremendous help on my admission. Without Daniel's aid, I would not even be able to come to the U.S. such that I could experience a new different world.

I am very grateful to have so many wonderful friends throughout my educational odyssey. I need to mention Senhua Chang and Chuhui Zhang, Xiaoyu Ge, Changsheng Liu, Duncan Yang, Jenny Lin, Yingjie Tang, Yubo Feng, Qiao Zhang, Jerry Ye, Yuhan Bi, Wei Liu, Longhao Li, Xiaozhong Zhang, Xiaoyu Liang, Haoran Zhang, Mingda Zhang, Keren Ye, Zinan Zhang, Lei Zhao, Xunsheng Yan, Jinpeng Zhou, Junyu Yang, Fangzhou Cheng and Wei Guo, Mingzhi Yu and Dengchao Meng, David Tsui, David Duan, David Neiman, Charlotte Chen, Nick VLDB, Lechong Zhou, Sihao Chen, Ruoyu Huang, Cong Xie, Ruoyu Jin, Jie Yang, Zhe Su, and Miaoshi Li, as well as my dear high school classmates who also study and work in the U.S., for marvelous times we spent together. I would not imagine how lonely I would be without their warm company and comfort. Again, best of my wishes to their future endeavors! Also, additional thanks to David Tsui and Agnes Kaminski for their careful proofreading and polishing of my thesis!

I was privileged to work as a machine learning scientist intern in Amazon Labs at Boston MA and Walmart Labs at Sunnyvale CA. I especially want to thank Duncan Yung for his referral to the latter, Dr. Peng Yang for being an awesome director, Xunfan Cai for being

an excellent mentor, as well as other teammates like Biyi Fang, Joanne Zhou, Bo Meng, and Yue Guo.

You know, The University of Pittsburgh or Pitt has the best look of a top and comprehensive university in my mind. Here, boys and girls are passionate, athletic, smart, and diligent, working together in sports clubs or study teams. They always impress me with their huge amounts of energy. I was extraordinarily fortunate to attend Pitt where I received an excellent education, research resources, health and mental care, and wellness. Its campus is splendid and atmosphere open, diverse, and friendly. Pitt is also leading in caring international students and thus I want to thank Pitt OIS, especially Richard who helped me with the international stuff. In my department, I have been well taken care of by the staff. I want to thank the administrative staff Keena Walker, Karen Dicks, Michele Thomas, and Deb Lauro; as well as the tech staff Bob Hoffman, Terry Wood, the late Russ Howard, and Adam Hobaugh who were always on standby for trouble calls and made sure the Elements cluster is running 24/7.

Finally, my super super parents! They gave and will keep giving me lifelong support for my life, study, work, and everything. I love you! Also, I want to thank my grandparents, cousins, aunts, and uncles. I really enjoy the time with them and I really miss them!

Thank you all! All the best!

1.0 Introduction

1.1 Motivation

With the rapid growth of computational power and electronic storage, large real-world datasets have been generated in various areas of science [Mjolsness and DeCoste, 2001], business [Bose and Mahapatra, 2001], technology [Jordan and Mitchell, 2015], and medicine [Hauskrecht et al., 2013, Hauskrecht et al., 2016]. Data analysis and machine learning methods have become crucial in understanding such datasets, as well as, for building data-driven models that can be applied to solve various problems in respective areas.

Supervised learning refers to the problem of learning a function (model) $f : \mathcal{X} \to \mathcal{Y}$ mapping inputs ($\mathcal{X}$) to outputs ($\mathcal{Y}$) from data. This model is most often learned from many data instances that consist of (input, output) pairs where the outputs represent desired outcomes for the inputs. To assure that the trained model f generalizes well to unseen data, the volume of the training data needs to be large enough [Friedman et al., 2001, Settles, 2012], especially when the input space $\mathcal{X}$ is high-dimensional and/or the underlying relationship between $\mathcal{X}$ and $\mathcal{Y}$ is complex.

Sometimes the input and output data are readily available. For example, email users mark "spam" flags on unwanted email messages; online-shopping users give the five-star ratings to products they have purchased. Supervised learning algorithms can use these flags or ratings to better filter junk emails or to improve the shopping experience. However, for many other sophisticated supervised learning tasks the output data must be defined by a human through a separate annotation process [Settles, 2012, Kovashka et al., 2016]. Unfortunately, this process may often incur a significant cost associated with the review or assessment of data [Nguyen et al., 2014, Valizadegan et al., 2013]. As a result, the annotation cost can dramatically limit the number of instances one may feasibly label, hurting the quality of the trained models. Because of this issue, a key challenge is to find effective ways to reduce the annotation effort while guaranteeing that models built from the limited feedback are accurate enough to be applied in practice.

1.2 Data Annotation Solutions

One widely applied solution to reduce annotation effort is *active learning* [Settles, 2012]. A typical active learning process begins with a few labeled data instances; then it repeatedly (1) learns the model from already labeled data and (2) queries new unlabeled data instances to further improve the model. So the beauty of active learning is that it can actively decide what data instances should be labeled next. Active learning has been successfully applied in domains as diverse as computer vision [Kovashka et al., 2011], bio-medical data mining [Nguyen et al., 2014, Valizadegan et al., 2013], and natural language processing [Hwa, 2004, Druck et al., 2009]. Theoretical work [Balcan et al., 2009, Dasgupta, 2011] has demonstrated its efficacy.

Despite enormous progress of active learning research, the majority of current methods focus on *instance-based* querying strategies [Cohn et al., 1996, Settles, 2012]. That is, they query labels on individual data instances and then learn models from them. Unfortunately, this may limit their applicability when targeting real-world classification tasks. There are two reasons for this:

1. First, for more complex input space the number of instances one may feasibly label may still be insufficient to properly cover the entire data space and represent the underlying data distribution. As a consequence, models induced from them may carry a lot of uncertainty and bias [Cohn et al., 1996], and hence, they may be inadequate to be deployed in practice.
2. On top of the first reason, active learning often relies on biased models to select the data instances to be labeled next. Unfortunately, such selected data can deviate from the underlying data distribution. This is often referred to as *sampling bias* problem [Dasgupta and Hsu, 2008]. Insufficient management of the sampling bias may fail the active learning process. But clearly, it is not trivial to identify representative data instances that can well reflect the true data distribution.

Besides active learning, another promising research direction that aims to save human annotation effort is the utilization of alternative forms of efficient human feedback. This

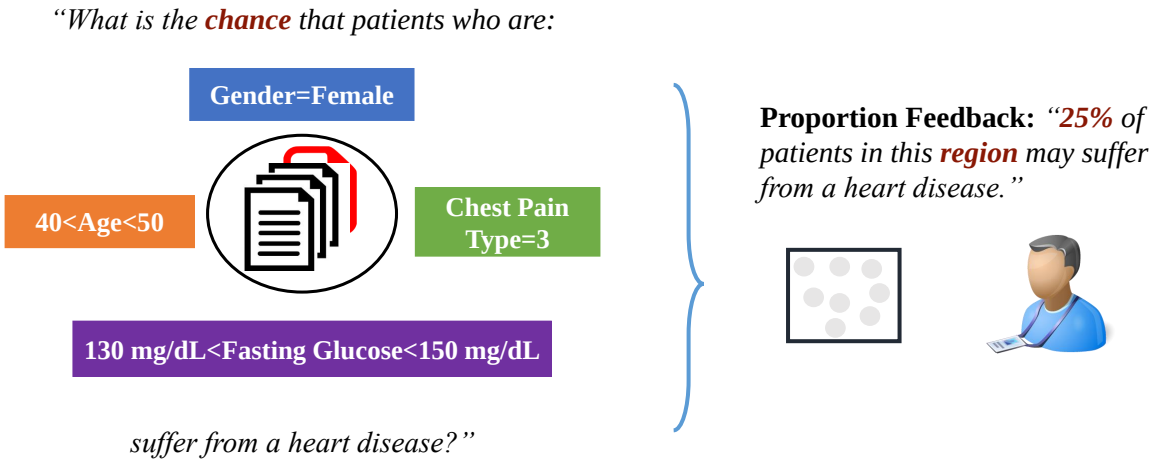

Figure 1.1: An example of a region-based query for the heart disease problem (originally introduced by [Rashidi and Cook, 2011]). A few patient instances on the left are originally recorded using 24 features. But after they are grouped, the whole group can be described by a conjunctive pattern defined using only four relevant features. The conjunctive patterns define a hyper-cubic region in the original feature space; an annotator (a physician) takes the region description and gives it a class proportion label. Note that individual cases are not seen or labeled individually.

direction is complementary to active learning research. The motivation is that in some applications the conventional instance labeling process may not be the only and the most efficient way to obtain human feedback. Sometimes instance labels can be difficult to query [Qian et al., 2013] or even infeasible to obtain [Quadrianto et al., 2009]. In general, human knowledge and feedback about the classification domain may take on different forms that deviate from standard instance-based (or case-based) class-label annotation. Some of these approaches may be converted (with more or less computational effort) to workable instance-based classification models. A trivial example is expert-defined ("if-then") rules [Zadeh, 1992, Ishibuchi et al., 1995] that specify classes of interest in the input space. They can be directly converted to classification models. Other examples of human feedback useful for building classification models include:

1. **Learning from Soft-Label Feedback**. The work by [Nguyen et al., 2011a],

[Nguyen et al., 2011b, Nguyen et al., 2014, Xue and Hauskrecht, 2017b], and [Xue and Hauskrecht, 2017a, Xue and Hauskrecht, 2018, Xue and Hauskrecht, 2019] has explored ways of enrichment of standard instance labeling with additional auxiliary information. The idea is to obtain, in addition to the class label, also *soft-label* information that reflects annotator's *uncertainty* about this class label. One way to express the soft-label feedback is to use *probabilistic* score reflecting the annotator's estimate of the probability of the target class, such as, "*The probability of the patient suffering from a heart disease is 0.75*". The soft-label information, when properly used, has been shown to improve the labeling efficiency and cut down the number of instances one has to annotate [Nguyen et al., 2011a, Nguyen et al., 2014].

2. **Learning from Relative Feedback**. Learning to rank and learning distance metrics are important components of many web applications such as product recommendation and document browsing. Typically, relative feedback among examples is obtained from online users at a low cost. For instance, [Joachims, 2002, Xing et al., 2003, Qian et al., 2013, Kovashka et al., 2012] learn ranking functions from pairwise relative feedback, i.e. "*Object A is more (adjective) than B*". We note that this approach can be also used to build classification models. Briefly, the pairwise relative feedback can help one to order instances with respect to the class-membership and use these to learn a ranking function that can be used to define a classification model. For example, a question "*Is patient A more likely to have a target disease than patient B*"when applied to pairs of patients could be used to build a classification model for disease diagnosis. Another form of relative feedback uses triplets. An example of such a feedback is "*Object A is closer to C than B is to C*". [Schultz and Joachims, 2004] and [Heim et al., 2014, Heim and Hauskrecht, 2015] use the relative triple feedback to learn either distance metrics and kernels (object similarities). In some cases providing relative feedback can be mentally less fatiguing to humans compared to providing instance labels. However, the main drawback of relative feedback is the number of queries one needs to assess before learning the ordering function. Because of that, labeling based on soft labels (see above) is typically more query efficient for learning ordering functions and classification models from n instances.

3. **Learning with Feature Feedback**. Feature feedback is another useful form of human knowledge that can lead to improved model learning. The motivation is that, every so often, humans can easily provide feature information aside from their usual annotation of instances. Such feature feedback can be incorporated into model learning to further improve the model learning. Early work on feature labeling [Hema et al., 2006, Druck et al., 2009] empirically has shown active feature labeling is more effective than active instance labeling. Recently, [Poulis and Dasgupta, 2017, Dasgupta et al., 2018] have theoretically demonstrated the learning efficiency with feature feedback.
4. **Multiple Instance Learning**. In multiple instance learning (MIL), instances are grouped into *bags*, and it is bags, rather than instances, that are labeled for training [Amores, 2013]. MIL relies on *asymmetric* label feedback. That is, a bag is labeled positive if there exists a positive instance in it; otherwise, it is labeled negative. This kind of feedback is extremely helpful in image labeling where an image (a bag) consists of a set of objects or segments (instances).Then a *positive* label means there is one or more objects of interest present in the image [Settles et al., 2008, Salmani and Sridharan, 2014]. With such a type of feedback, classifiers, either bag-based or instance-based, can be learned by using multiple instance learning algorithms, such as, multiple instance logistic regression or multiple instance support vector machines [Amores, 2013].
5. **Learning from Label Proportions**. Recently there is novel work [Du and Ling, 2010, Rashidi and Cook, 2011] (as well as the work in this thesis) that asks human annotators to provide class proportion feedback on *groups*. The group concept is similar to the bag concept in MIL but groups use proportion labels. One proportion label represents a human estimate of the proportion of the instances in the group that belong to one of the classes, or equivalently, the probability with which an instance with that class label is drawn from the group. For example, "*70% instances in this group are positive*". The essential benefit of such a feedback is that it allows annotators to label a group of instances together at the same time. Such labeled groups can be used to recover the instance-level classification rules.[Quadrianto et al., 2009, Yu et al., 2013] have developed class proportion learning algorithms to tackle this problem.

1.3 Thesis Work

In general, the work in this thesis aims to alleviate the annotation cost problem by exploring efficient human feedback that is easy for humans to perform but also informative for models to learn from. Along this line, we study *group-based* feedback from humans. Below is an overview of the key notions that are used throughout this thesis:

1. **Instance**. As in the conventional setting, a data instance is a data element that belongs to a given input data space. The input space is composed of numeric, ordinal, or categorical features. Each instance is specified by a set of feature values and also comes with a class label that belongs to a target class space.
2. **Group**. A group refers to a compact set/bag of data instances.
3. **Group Description**. To describe groups to human annotators, we use *conjunctive patterns* that are defined by value ranges of the input features. For example, in Figure 1.1, a group of patient cases may be described as: *"(Gender=Female) $\wedge$ (40<Age<50) $\wedge$ (Chest Pain Type=3) $\wedge$ (130<Fasting Glucose<150 mg/dL) etc"*.
4. **Region**. Conjunctive patterns correspond to a hypercubic *region* that is a sub-space of the input space. In general, a region defines a *subpopulation* of data instances.
5. **Group Labeling**. Groups (or regions) can be assigned a label. In our case it is a *proportion label*. When assessing a group, human annotators only need to review its region-based description and then provide a proportion label. The proportion label summarizes the proportion of positive class instances in the subpopulation represented by the region. Figure 1.1 shows an example of region annotation for a clinical domain. A physician may assess that *"25% of patients in a subpopulation defined by a region may suffer from a heart disease"*.
6. **Learning from Label Proportions**. Our ultimate goal is to learn an *instance-based* classification model from a set of labeled groups. Learning from Label Proportions (LLP) offers a variety of learning algorithms to achieve this goal. For example, in Section § 4.3 we will utilize a simple algorithm that is based on *instance sampling*. But this algorithm only works for a set of disjoint groups. Algorithms that can learn classifiers from overlapping

groups have been developed by [Quadrianto et al., 2009, Yu et al., 2013]. They can be integrated into our framework as well.

The work in this thesis is closely related to MIL (*multiple instance learning*) and LLP (*learning from label proportions*). However, the main difference is that both MIL and LLP assume bags or groups are known apriori. Our work studies learning problems with general datasets where groups or regions are not explicitly defined. This poses a key question: *How do we generate groups for labeling and learning?* In general, there are exponentially many groups that can be constructed from a set of instances; more predominantly, there can be infinitely many regions that can be defined over a real-valued input space. Therefore, in order to identify only a small number of regions that are important for model learning, we need to develop an *active learning* strategy matching the problem and group label feedback. To this end, we propose and investigate a new machine learning approach - **Group-Based Active Learning**.

Compared to conventional active learning methods, our group-based active learning approach has two major distinctions. On one hand, groups can cover a more portion of the input space and thus better represent the data distribution. Hence, learning from a few large and general groups can effectively mitigate the sampling bias issue mentioned before. On the other hand, groups, especially large groups, often come with impure proportion labels that are uninformative for model learning. As shall be seen in Chapter § 3, learning with sufficiently many *pure-enough* groups is the key to recover the underlying instance-level classification rules. Therefore, the corresponding group-construction strategy needs to progressively identify smaller and purer groups among those impure groups.

Following these thoughts, we develop a **Hierarchical Active Learning** framework. The goal is to build a concise region hierarchy that can rapidly refine the leaf regions pure. Briefly, the framework starts from a few general and large groups to label and uses them to learn a classification model. After that, it repeatedly splits impure groups into smaller ones, solicits their labels from annotators, and then retrains the model. To maximize the label information gain of each split, we follow *Maximum Information Gain*, or equivalently, *Maximum Class Entropy Reduction*, the principle that is employed by the decision tree learning algorithms. However, due to the lack of instance labels, we develop two splitting

heuristics. The first one is a *supervised heuristic*. It uses the classification model we aim to learn to estimate the information gain of the splits. However, its limitation is that if the model is poor, especially during the initial active learning cycles the estimates it makes can be inaccurate and unreliable. Thus, we also consider an *unsupervised heuristic* to help estimate the gain. The unsupervised heuristic uses clustering techniques to divide a group of instances into a few clusters (child groups). If the class distribution is reasonably aligned with the structure of data, separating clusters can lead to a good separation of classes. With these two heuristics available, a key problem is how to combine them for estimating the true gain. This problem that will be explored in this thesis.

We have implemented three versions of our hierarchical active learning framework. The first two build a *single* hierarchy of groups/regions:

- **HALG**: **H**ierarchical **A**ctive **L**earning with **G**roup proportion feedback
  [Luo and Hauskrecht, 2018a, Luo and Hauskrecht, 2017a]
  (To be presented in Chapter § 4)
- **HALR**: **H**ierarchical **A**ctive **L**earning with proportion feedback on **R**egions
  [Luo and Hauskrecht, 2018b, Luo and Hauskrecht, 2019, Luo and Hauskrecht, 2017b]
  (To be presented in Chapter § 5)

These two implementations differ in the way of building the region hierarchy. HALG builds a hierarchy of regions in a static way. It pre-compiles a hierarchy of clusters that defines a set of unlabeled groups. After that it keeps selecting in a top-down manner the "purest" group from this set for labeling. The purity of groups is estimated by a classification model. HALR, in contrast, builds a hierarchy *dynamically*. It works similarly to the decision tree learning process that directly splits the entire input space. Each of the HALR's splits is co-decided by the two heuristics and their weights to decide the splits are dynamically adapted. A more detailed comparison between HALG and HALR will be provided at the end of Chapter § 5.

When building a single hierarchy, we observe that the unsupervised heuristic usually dominates the hierarchy building process. In the worst case, the hierarchy could end up being completely class-irrelevant, i.e. having no pure regions. To mitigate such an issue, we propose a more robust approach **HALOR** that grows multiple HALR hierarchies in parallel

and permits regions *overlapping* with one another.

- **HALOR**: **H**ierarchical **A**ctive **L**earning with **O**verlapping **R**egions
  [Luo and Hauskrecht, 2020]
  (To be presented in Chapter § 6)

HALOR provides additional flexibility of exploring more class-relevant regions among different hierarchies. The intuition behind HALOR is that if some hierarchies fail to find pure regions, HALOR is capable of growing different hierarchies that can potentially find more informative regions. Apart from this advantage, HALOR also reduces the complexity of region queries (i.e. the number of features used in region description) and thus simplifies the annotation process.

1.4 Contributions

The work in this thesis is centered around two hypotheses. The first hypothesis compares learning from group-based feedback versus learning from the conventional instance-based feedback under active learning setting:

H1: *Active learning of binary classification models from group-based feedback can be more query-efficient than learning from instance-based feedback.*

To measure the query efficiency we use two metrics: (1) the number of queries that are needed to build a model and (2) query complexity. Methods that consume fewer and/or simpler queries are considered more query-efficient.

The second hypothesis compares our method to the existing group-based active learning work [Du and Ling, 2010, Rashidi and Cook, 2011]. The main difference is that they construct groups arbitrarily from individual instances while we construct groups hierarchically.

H2: *Our hierarchical approach can discover more informative and simpler groups than existing methods that identify groups based upon instances, and therefore our solution is more query-efficient.*

The above two hypotheses are the main goals of this thesis to study and evaluate. We have

conducted comprehensive experiments to evaluate our solutions, as well as, multiple baseline methods. The results demonstrate the following findings:

1. When the labeling budget is severely restricted, one can learn better models from a few groups than learning from the same number of instances.
2. When the labeling budget is fairly sufficient, group-based active learning methods achieve earlier model convergence than instance-based ones.
3. Region-based queries are less complex since they only use a small subset of features in their descriptions.
4. Our hierarchical approaches can find informative and simpler groups rapidly, thereby shown to be more query-efficient than other alternative methods.

Lastly, we would like to note that parts of the work in this thesis have been published as multiple conference papers: [Luo and Hauskrecht, 2017a] in NIPS Workshop 2017, [Luo and Hauskrecht, 2017b] in FLAIRS 2017, [Luo and Hauskrecht, 2018a] in IJCAI 2018, [Luo and Hauskrecht, 2018b] in ECML 2018, [Luo and Hauskrecht, 2019] in SDM 2019, and [Luo and Hauskrecht, 2020] in CIKM 2020.

1.5 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter § 2 overviews the background of this thesis. We will start with the introduction of supervised learning and classification tasks, and then review two lines of work aimed at reducing the human annotation effort: *active learning* and *learning from alternative feedback*.
- Chapter § 3 presents a couple of *learning from label proportions* algorithms, as well as, their key conditions that ensure their learning success (i.e. recovery of instance-based classification rules from a set of labeled bags). These results lay the foundation of the work in this thesis and drive the development of our hierarchical active learning (HAL) framework.

- Chapter § 4 presents the first implementation of our HAL framework that works with hierarchical clustering and groups (HALG).
- Chapter § 5 presents the second implementation of our HAL framework that dynamically builds a hierarchy of regions (HALR).
- Chapter § 6 presents the third implementation of our HAL framework that grows multiple hierarchies of regions and permits learning from more varied regions that can overlap with one another (HALOR).
- Chapter § 7 summarizes the contributions of the thesis, limitations of the current group active learning framework and its implementations, and presents open problems and future work directions.

2.0 Background

2.1 Supervised Learning

Learning of classification models is one type of supervised learning. In general, the goal of supervised learning is to learn a function (model) $f : \mathcal{X} \to \mathcal{Y}$ from an *input* space $\mathcal{X} \subset \mathbf{R}^m$ to a *target* space $\mathcal{Y} \subset \mathbf{R}$. This function is most often learned from training data that consist of (input, output) pairs: $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. Each instance $\boldsymbol{x}_i$ is independently and identically (i.i.d.) sampled from the input space $\mathcal{X}$, and its output (or label) $y_i \in \mathcal{Y}$ represents the desired outcome of $\boldsymbol{x}_i$. For regression problems outputs are real numbers; for classification problem outputs are class labels. In this thesis, we study binary classification problems where we use $\mathcal{Y} = \{0, 1\}$ as the class labels. Binary classification problems have been widely studied both empirically [Friedman et al., 2001] and theoretically [Vapnik, 1998].

Learning of a well-performing classification model usually requires large volumes of training data [Bishop, 2006]. It is especially the case when the input space $\mathcal{X}$ is high-dimensional and/or the underlying relationship between $\mathcal{X}$ and $\mathcal{Y}$ is complex [Settles, 2012]. Sometimes the training input-output data pairs come at no or little cost. For example, historical observations of stock data or customer ratings of online products. But for many other tasks the data do not include class labels and they must be defined and provided by a human annotator [Nguyen et al., 2014, Valizadegan et al., 2013]. Unfortunately, obtaining such labeled data can be very difficult, time-consuming, or expensive. Here are a few examples:

- **Medical data mining**: In the medical domain, if a physician wants to accurately diagnose a patient (e.g. for a possible heart condition) he/she must review the patient record that consists of complex collections of results, symptoms, and findings (such as *age, BMI, glucose levels, HbA1c blood test, blood pressure, heart rate* etc). The review and the assessment of these records with respect to a specific condition may become extremely time-consuming as it often requires physicians to peruse through a large quantity of data [Nguyen et al., 2014, Hauskrecht et al., 2016].
- **Document classification**: Learning to classify documents (e.g., articles or web pages)

requires user annotation on each of the document files like "relevant" or "not relevant". Annotating hundreds or thousands of documents can be tedious and even redundant [Druck et al., 2009, Roy and McCallum, 2001].

- **Image filtering**: Image labeling, object tagging, etc. in computer vision research also requires a substantial amount of human annotation [Settles et al., 2008]. With a large quantity of data that need labeling, one often has to resort to crowd-sourcing resources of which the services are expensive [Kovashka et al., 2016].

If one has a very limited labeling budget, how could they spend the budget cost-effectively? Therefore, a key challenge would be finding a way to build high-quality classification models from limited human feedback. Broadly speaking, there are two different solutions to solve this problem: one is *active learning* that aims to reduce the number of labeled data; the other one is *learning from alternative feedback* that focuses on obtaining (from human) other than class-label information that can be effectively used build a classification model. The following two sections present them accordingly.

2.2 Active Learning

The first machine learning solution is active learning. It aims to reduce the number of labeled data by strategically choosing a small subset of the data that are important for labeling and learning. Active learning collects labeled data *iteratively* and builds models *incrementally*. In each iteration, an active learning strategy would choose only the *important* or *necessary* data to be labeled. This avoids labeling redundant examples or examples that contribute little to model improvement. Therefore, active learning works differently from traditional passive supervised learning where the training data are *randomly* generated. The efficacy of active learning has been demonstrated by empirical [Kovashka et al., 2011, Nguyen et al., 2014, Valizadegan et al., 2013, Hwa, 2004, Druck et al., 2009] and theoretical studies [Cohn et al., 1996, Balcan et al., 2009, Dasgupta, 2011].

Algorithm 1 Pool-Based Active Learning Framework

1: Collect a pool $\mathcal{U}$ of i.i.d. sampled data $\boldsymbol{x}$ that are all unlabeled ($\mathcal{U}$ is large enough)
2: Also collect an i.i.d. sample of labeled data $\mathcal{L} = \{\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l)\}\}$ ($|\mathcal{L}| \ll |\mathcal{U}|$)
3: **repeat**
4: Learn the base model $\mathcal{M}$ based on $\mathcal{L}$
5: Strategically pick one (or a batch of) instance $\boldsymbol{x}^* \in \mathcal{U}$ according to some utility function $U_{\mathcal{M}}(\boldsymbol{x})$ that is associated with $\mathcal{M}$'s performance on $\boldsymbol{x}$
6: Have $\boldsymbol{x}^*$ labeled as $(\boldsymbol{x}^*, y^*)$ and add it to $\mathcal{L}$
7: **until** certain stopping criterion is met

2.2.1 Pool-Based Active Learning

According to Settles' survey of active learning [Settles, 2012], there are many types of active learning, and *pool-based* active learning is the most widely applied one in practice. The basic assumption is that data may be abundant but their labels are scarce or expensive to obtain. For example, texts or images are widely available but their labels are not and thus need additional human annotation. The general pool-based active learning framework is summarized in Algorithm 1.

2.2.2 Query Strategies

The most interesting part of active learning affecting its performance is strategy $U_{\mathcal{M}}(\boldsymbol{x})$ in line 5 in Algorithm 1. Numerous effective heuristics have been developed for this purpose. Some of the classic ones are summarized below:

- **Uncertainty Sampling** [Lewis and Catlett, 1994]: Uncertainty sampling queries the instances about which the base model $\mathcal{M}$ is *least certain* how to label. This approach is often straightforward for probabilistic learning models. For example, when using a probabilistic model for binary classification, uncertainty sampling simply queries the instance whose posterior probability of being positive is nearest 0.5.
- **Query-by-Committee** [Seung et al., 1992]: This strategy maintains a committee $\mathcal{C} =$

$\{\mathcal{M}^{(1)}, ..., \mathcal{M}^{(c)}\}$ of models that are all trained on the same labeled data set $\mathcal{L}$ but represent competing hypotheses of the true model. Each committee member is then allowed to vote on the labeling of some query candidates. The most informative query is considered to be the instance about which they most disagree.

- **Maximum Model Change** [Roy and McCallum, 2001, Settles et al., 2008] and [Freytag et al., 2014]: This thread of work uses a decision-theoretic approach, selecting the instance that would impart the greatest change to the current model *if its label were known*. The intuition is that if some instances (when labeled) can update the model a lot in each iteration, then they would (in expectation) help the model converge quickly to an asymptotically optimal model.

2.2.3 Cluster-Driven Active Learning Strategies

Sometimes basic active learning strategies may not always work. They may even slow down or misguide the learning process. The primary drawback of the basic active learning methods is that unlabeled data instances are considered independently. As a result, an outlier example that is not representative of other unlabeled data may be selected which in turn cannot help to improve the accuracy of the model.

In order to mitigate this issue, researchers have considered combining *unsupervised data analysis* and basic querying strategies. Unsupervised data analysis helps to reveal the structure of input data. Such a structure may be also closely related to class distributions in the input space. This assumption is used for example in semi-supervised learning [Zhu et al., 2003] where labeled examples are combined with unsupervised information to learn improved classification models. Examples of relevant work include [Settles and Craven, 2008] who proposed a density-weighted method that queries the instances that are not only uncertain but also *representative* of the other data; [Dasgupta and Hsu, 2008, Urner et al., 2013, Nguyen and Smeulders, 2004] leverage clustering heuristics to drive the instance selection procedure.

2.2.4 Limitations

Despite its positive impact, one must realize that active learning may not always work. Here we stress two limitations of active learning. First, for a more complex input space, the number of instances one may feasibly label can still be insufficient to represent the underlying data distribution. If this is the case, a model induced from a limited amount of training data is likely to contain a lot of bias and uncertainty [Cohn et al., 1996]. The second limitation, on top of the first problem, is that active learning often relies on biased models to select new data instances in the successive iterations. Unfortunately, instances sampled by a biased model can significantly deviate from the underlying data distribution. This is often referred to as *sampling bias* problem [Dasgupta and Hsu, 2008]. Below is the quick intuition of sampling bias problem by Dasgupta:

"A typical active learning heuristic might start by querying a few randomly-chosen points to get a very rough idea of the decision boundary. It might then query points that are increasingly closer to its current estimate of the boundary, with the hope of rapidly honing in. However, as training proceeds and points are queried based on increasingly confident assessments of their informativeness, the training set can quickly diverge from the underlying data distribution. If this sampling bias is not properly managed, the learning process may fail to output a consistent model."

Insufficient management of the sampling bias issue may fail the active learning process.

2.3 Learning From Alternative Feedback

Aside from active learning, the second solution focuses on exploring new types of human feedback that is easy to collect and also informative to model learning. The motivation is that sometimes exact instance labels may be hard for humans to provide. Here is one example. For the "Galaxy Zoo" example shown in [Qian et al., 2013], it is extremely hard for non-expert annotators to tell which galaxy (a label) each star (an instance) belongs to. Worse, instance labels can even be infeasible to obtain [Kück and de Freitas, 2005, Quadrianto et al., 2009].

Consider situations where privacy is an issue, such as people's voting results, students' grading, patients' medical records, etc., getting each individual's information is impossible. In light of these issues, various types of alternative human feedback have been proposed and studied. In the rest of this chapter, we would survey three types of novel human feedback that are most relevant to the work in this thesis. They are (1) soft-label feedback that is auxiliary label information provided additionally to the basic instance labeling, (2) multiple instance learning (MIL), and (3) learning from label proportions (LLP). Besides, each of them can also work with a matching active learning strategy. They are, respectively, active learning from soft-label feedback, multiple instance active learning (MIAL), and active learning from label proportions (ALLP).

2.3.1 Learning From Soft-Label Feedback

[Nguyen et al., 2011a, Nguyen et al., 2011b, Nguyen et al., 2014], [Xue and Hauskrecht, 2017b, Xue and Hauskrecht, 2017a, Xue and Hauskrecht, 2018] and [Xue and Hauskrecht, 2019] have explored ways of enrichment of standard instance labeling with additional auxiliary information. That is, besides asking for instance labels they also allow the annotators to provide *soft-labels* reflecting the annotator's belief the class label is indeed true. This additional information provides not only robustness to the acquired labels but also more flexibility for humans to express their belief in the label, thereby helping learn more accurate models. Hence, the *soft-label* feedback expands on *exact* or *hard* instance-label feedback.

There are different ways to express the soft-label feedback. One way is to use *probabilistic* labels [Nguyen et al., 2011a, Nguyen et al., 2011b, Nguyen et al., 2014]. For example, when obtaining feedback from a physician on whether a patient suffers from a particular disease or not, the binary true/false feedback can be refined by inquiring about the physician's belief about the chance of the disease's presence. Say, "*The probability that this patient will have heart disease is 70%*". Another way, for example, [Xue and Hauskrecht, 2017a, Xue and Hauskrecht, 2018, Xue and Hauskrecht, 2019] proposes to use Likert-scale categories [Likert, 1932] as the auxiliary information. Briefly, Likert defined a set of ordinal

categories humans can use to provide information about the strength of agreement (or belief) in the respective class labels. With such Likert-scale labels, humans do not have to provide an exact probabilistic label or a confidence score. For the above disease example again, a physician can express that if he/she *agrees*, *weakly agrees*, *is neutral*, *weakly disagrees*, or *disagrees* that the disease will be present on that patient.

As for model learning, they develop algorithms that are based on ordinal regression and ranking. Specifically, they aim to learn a ranking function $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ from training data $\{(\boldsymbol{x}_1, y_1, u_1), ..., (\boldsymbol{x}_n, y_n, u_n)\}$. Here $\boldsymbol{x}_i$ and y_i are standard (input, output) pairs, but u_i is an additional soft label indicating the confidence that the data instance $\boldsymbol{x}_i$ falls into the class y_i. Both y_i and u_i are assigned by human annotators. The reason for learning a ranking function is because there exists an ordinal relationship among these training triplets. In other words, if two data instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are assigned ordinal labels where $u_i > u_j$, they expect that the same order should be preserved as well by the ranking function, i.e. $g(\boldsymbol{x}_i) > g(\boldsymbol{x}_j)$. Therefore, they build such an ordinal relationship as a set of constraints and add them to a standard optimization procedure. For example, a popular choice is ranking-SVM [Joachims, 2002]. Its key idea is to find the best separating hyper-plane among training data while also satisfying the ordinal constraints. Concretely:

$$\text{minimize:}$$

$$\frac{\boldsymbol{w}^T\boldsymbol{w}}{2} + B\sum_{i=1}^{n}\eta_i + C\sum_{i=1}^{n}\sum_{j\neq i}^{n}\xi_{j,i}$$

$$\text{subject to:}$$

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 - \eta_i \qquad \forall i$$

$$\boldsymbol{w}^T\boldsymbol{x}_i \geq \boldsymbol{w}^T\boldsymbol{x}_j + 1 - \xi_{j,i} \qquad \forall (u_i > u_j)$$

$$\eta_i, \xi_{j,i} \geq 0 \qquad \forall i, j$$

In the objective function above, the first term is a regularizer of $\boldsymbol{w}$; the second term (single sum) defines the hinge loss on binary labels; the third term (double sum) defines the loss function between each ordered pair suggested by the soft labels. Once the minimizer $\hat{\boldsymbol{w}}$ is learned, the ranking function $g(\boldsymbol{x}) = \hat{\boldsymbol{w}}^T\boldsymbol{x}$ will be determined. And finally, a classification

function $y = f(\boldsymbol{x})$ can be further built upon $g(\boldsymbol{x})$ by determining a threshold on the ranking line.

2.3.1.1 Active Learning With Auxiliary Label Information

To further save human annotation effort, [Xue and Hauskrecht, 2017a, Xue and Hauskrecht, 2018] and [Xue and Hauskrecht, 2019] propose to actively query the labels (both the class labels and the auxiliary) for the training triplets $\{(\boldsymbol{x}_1, y_1, u_1), ..., (\boldsymbol{x}_n, y_n, u_n)\}$. For example, in the work of [Xue and Hauskrecht, 2017a] they use *expected model change* (EMC) as the active learning strategy. EWC works similarly to the *maximum model change* introduced earlier, and it measures to what extent each unlabeled instance would change the model if it were labeled. As the true label is unknown before it is queried, they approximate it with a label distribution that is inferred by the classification model they aim to learn.

With more details, suppose one has already learned a classification model $f_{\mathcal{L}}$ from current labeled data $\mathcal{L}$. Also assume that there are m possible Likert-scale ordinal categories for $\boldsymbol{x}$. Based on current model $f = f_{\mathcal{L}}$, the probability that $\boldsymbol{x}$ is assigned a Likert-scale label $u = 1, 2, .., m$ is inferred as $p_f(u|\boldsymbol{x})$. With this label distribution, one can estimate the expected model change as follows. For each u, label an instance $\boldsymbol{x}$ as $(\boldsymbol{x}, y, u)$[1], add the triplet to $\mathcal{L}$, and learn an add-one model $f_{\mathcal{L}\cup(\boldsymbol{x},y,u)}$. The model change of $f_{\mathcal{L}\cup(\boldsymbol{x},y,u)}$ compared to $f_{\mathcal{L}}$ is computed as $\delta(\boldsymbol{x}, u)$. m Likert-scale labels give rise to m model changes $\delta(\mathbf{x}, u)$, $u = 1, ..., m$. Then, the expected model change $\Delta_f(\boldsymbol{x})$ of $\boldsymbol{x}$ based on current model f is calculated as:

$$\Delta_f(\boldsymbol{x}) = \sum_{u=1}^{m} p_f(u|\boldsymbol{x})\delta(\boldsymbol{x}, u)$$

Finally, the instance $\boldsymbol{x}^*$ that leads to the maximum expected model change will be selected for labeling. Please note that this $\Delta_f(\boldsymbol{x})$ is one instantiation of the general utility function $U_{\mathcal{M}}(\boldsymbol{x})$ in line 5 in Algorithm 1 within the framework of pool-based active learning (Section § 2.2.1).

[1]They work with binary classification models, so $y = 0$ or 1 does not matter here.

2.3.2 Multiple Instance Learning

In multiple instance learning (MIL) [Amores, 2013], instances are naturally grouped into *bags* (or sets, groups, regions), and it is bags, rather than instances, that are labeled for training. For example, in context-based image retrieval, an image (bag) is a set of smaller segments or objects (instances); in document classification, a document (bag) is a set of paragraphs or sentences (instances). To label bags, humans provide *asymmetric* feedback. That is, a bag is labeled positive if there exists at least one positive instance in it; otherwise, the bag is labeled negative. It is rational to use asymmetric labeling because the *positive* label may have a semantic meaning that if some instance of interest is present in the bag or not. Some example queries are: "*Is there a cat in that image?*", or "*Is there any offensive sentence written in this document?*". The MIL setting was first formalized by [Dietterich et al., 1997] in the context of drug activity prediction and has since been applied to a variety of tasks, such as text classification [Andrews et al., 2003, Ray and Craven, 2005] and content-based image retrieval [Maron and Lozano-Pérez, 1998, Andrews et al., 2003, Rahmani and Goldman, 2006].

As for model learning, either instance-based or bag-based classifiers can be trained. In this review, we focus on learning an instance-based classifier $y = f(\boldsymbol{x}) \in \{0, 1\}$ from a set of training bags. Since now labels are only provided to bags, the learning algorithms must construct a bag-level classifier $F(B)$ from the instance-level classifier f by some operator. A general operator is *aggregation* [Amores, 2013], given by the following formula:

$$F(B) = \frac{f(\boldsymbol{x}_1) \circ f(\boldsymbol{x}_2) \circ ... \circ f(\boldsymbol{x}_b)}{Z}$$

where a bag $B = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}$ contains b instances; $\circ$ denotes an aggregation operator, specific to an MIL algorithm; Z is a normalization factor. One popular choice of $\circ$ is to use *max* operation:

$$F(B) = \max_{\boldsymbol{x}_i \in B} f(\boldsymbol{x}_i)$$

It makes sense because of the asymmetric labeling - that every positive bag contains at least one positive instance while all negative bags contain none. This *max* operation was first developed by [Dietterich et al., 1997] and then followed by many popular MIL algorithms, such as MI-SVM [Andrews et al., 2003], MI-Logistic Regression [Settles et al., 2008].

2.3.2.1 Multiple Instance Active Learning On top of MIL, multiple instance active learning (MIAL) has been proposed and studied in combination with active learning [Settles, 2012, Salmani and Sridharan, 2014]. MIAL follows the basic settings of MIL, and its novelty lies at that bag labels are actively collected (occasionally, instance labels may be queried if necessary). In Settles et al's work, they build the active learning strategy upon uncertainty sampling [Lewis and Catlett, 1994]. To demonstrate this, it is convenient to present their model first - an MI-Logistic Regression (MILR) model.

Suppose there are a number of bags $\{B_1, ..., B_n\}$ and each bag B_i consists of n_i instances $B_i = \{\boldsymbol{x}_{i1}, ..., \boldsymbol{x}_{in_i}\}$. An instance-level Logistic Regression is formulated as:

$$o_{ij} = P(y_{ij} = 1|\boldsymbol{x}_{ij}; \boldsymbol{w}) = \frac{1}{1 + \exp\{-(\boldsymbol{w}^T \boldsymbol{x}_{ij} + b)\}}$$

where $\boldsymbol{w}$ is a vector of parameters. In order to aggregate these instance-level class probabilities into a bag-level class probability, MILR uses *softmax*:

$$o_i = P(y_i = 1|B_i; \boldsymbol{w}) = \text{softmax}_\alpha(o_{i1}, ..., o_{in_i}) = \frac{\sum_j o_{ij} \exp\{\alpha o_{ij}\}}{\sum_j \exp\{\alpha o_{ij}\}}$$

where α is a hyperparameter that determines to what extent the softmax approximates a max function. With such a model, MILR is able to predict the class probability of either bags or instances. So the uncertainty score of a bag or an instance can be further given by an entropy measurement. They use Gini-Index:

$$U_{\boldsymbol{w}}(B_i) = 2o_i(1 - o_i), \qquad U_{\boldsymbol{w}}(\boldsymbol{x}_{ij}) = 2o_{ij}(1 - o_{ij})$$

With the $U_{\boldsymbol{w}}(.)$ function, they can actively select the most *uncertain* bags or instances for labeling. This $U_{\boldsymbol{w}}(.)$ function is also one instantiation of the general utility function $U_{\mathcal{M}}(\boldsymbol{x})$ in line 5 in Algorithm 1 within the framework of pool-based active learning (Section § 2.2.1). In terms of applications, they apply MIAL to context-based image retrieval and text classification and they show that MIAL could quickly find the objects of interest after consuming very few queries.

2.3.3 Learning From Label Proportions

Similar to multiple instance learning, *learning from label proportions* (LLP) also assumes that groups (or bags) of instances are available apriori and the label information is provided to groups. However, one essential difference is that LLP assumes using *proportion* labels that can provide richer class information. In general, a proportion label looks like: "*This group is a% positive (or negative)*", where $a \in [0, 100]$.

The use of proportion labels in LLP is well-motivated in many real-world learning problems. For example, consider activities such as elections, online purchasing, and students' grading where privacy is an issue, collecting individual's label information is impossible. However, collecting some *summarized* statistics, such as the mean or median among a group of people appears more feasible and thus easier to obtain [Quadrianto et al., 2009, Yu et al., 2013, Patrini et al., 2014, Rueping, 2010]. The proportion feedback used in LLP reflects the proportion of the instances in a group that belong to one of the classes. For example, in terms of the voting results of a county, the proportion label could be *"65% residents in Allegheny county vote for the Democratic Party"*. One could also think of the proportion feedback as one type of *soft-label* feedback that is provided to groups.

The goal of LLP is to learn instance-based classification models from a set of labeled groups. For example, given the voting results of some counties, one may learn a classifier that can predict the voting preference of any individual resident. More formally, an LLP learning problem can be formulated as follows:

- The training data are a set of labeled groups $\{(G_1, \mu_1), ..., (G_n, \mu_n)\}$, where G_i is a *bag* or *group* that contains n_i instances: $G_i = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_i}\}$. μ_i reflects the class proportions of the instances in group G_i. For example, in binary classification setting, each $\mu_i \in [0, 1]$ is a scalar that represents the proportion of the instances that belong to either the *positive* or *negative* class. In a multi-class classification problem, $\boldsymbol{\mu}_i$ would be a vector of class proportions.
- LLP algorithms output an instance-based classification model that is capable of making inference on any data instance $\boldsymbol{x}$. The classifier can be a deterministic function $f : \mathcal{X} \to \mathcal{Y}$ or a probabilistic model $P(y|\boldsymbol{x}; \boldsymbol{\theta})$.

- There are four categories of LLP algorithms that have been developed:
  1. The first category of algorithms models LLP problems within a statistical learning framework. They use the given proportion labels to approximate the sufficient statistics of a likelihood function [Quadrianto et al., 2009, Patrini et al., 2014]. Consider learning a conditional exponential models:

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \exp\{\langle\phi(\boldsymbol{x},y),\boldsymbol{\theta}\rangle - g(\boldsymbol{\theta}|\boldsymbol{x})\}$$

  where $\phi(\boldsymbol{x},y) : \mathcal{X}\times\mathcal{Y}\to H$ is a feature map to new space H, and g is the normalization factor (or log-partition function). If one *had* collected an instance-based training sample $(X,Y) = \{(\boldsymbol{x}_1,y_1),...,(\boldsymbol{x}_n,y_n)\}$, the log-likelihood function would be:

$$\log p(Y|X;\boldsymbol{\theta}) = \sum_{i=1}^{n}\{\langle\phi(\boldsymbol{x}_i,y_i),\boldsymbol{\theta}\rangle - g(\boldsymbol{\theta}|\boldsymbol{x}_i)\} = n\langle\mu_{XY},\boldsymbol{\theta}\rangle - \sum_{i=1}^{n} g(\boldsymbol{\theta}|\boldsymbol{x}_i)$$

  where $\mu_{XY} = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i,y_i)$ is the *sufficient statistics*. However, because instance training pairs are not observed, the sufficient statistics cannot be computed exactly. So the strategy is to use labeled groups $\{(G_1,\mu_1),...,(G_n,\mu_n)\}$ to estimate the sufficient statistics μ_{XY} as $\hat{\mu}_{XY}$ and then replace μ_{XY} with $\hat{\mu}_{XY}$ in the original log-likelihood function. Finally, standard optimization procedure can be applied to learn the model. More details will be presented in Chapter § 3.
  2. The second category of algorithms attempts to learn a model that can generate instance labels of which the class proportions are close to the given label proportions [Rueping, 2010, Yu et al., 2013, Kotzias et al., 2015]. Consider learning a Support Vector Machine: $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$. [Rueping, 2010] uses the following scaling function which is a special case of Platt [Platt et al., 1999] scaling to represent the probability of classifying $\boldsymbol{x}$ as $y = 1$:

$$p = \sigma(g(\boldsymbol{x})) = \frac{1}{1+\exp\{-g(\boldsymbol{x})\}}$$

  Then for any labeled group (G_i,μ_i), they try to fit the given label μ_i:

$$\forall i : \frac{1}{|G_i|}\sum_{i=1}^{n_i}\sigma(g(\boldsymbol{x})) \approx \mu_i$$

Therefore, the final optimization procedure is given by:

$$\text{minimize:}$$

$$\frac{\boldsymbol{w}^T\boldsymbol{w}}{2} + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to:}$$

$$\frac{1}{|G_i|}\sum_{i=1}^{n_i}\sigma(g(\boldsymbol{x})) \geq \mu_i - \xi_i, \qquad \forall i$$

$$\frac{1}{|G_i|}\sum_{i=1}^{n_i}\sigma(g(\boldsymbol{x})) \leq \mu_i + \xi_i^*, \qquad \forall i$$

$$\xi_i, \xi_i^* \geq 0, \qquad \forall i$$

Again, in Chapter § 3 we will provide more details.

3. The third category of algorithms is based on instance sampling. The key idea is to sample sufficiently many labeled instances from the labeled groups, and then feed the sampled instances to an instance-based learning algorithm. The primary advantage of this approach is that the learning is not restricted to a specific model family. However, a major limitation is that it cannot handle groups that can overlap with one another. We will detail this algorithm in Section § 4.3.
4. The last category of algorithms is based on instance weighting ([Du and Ling, 2010] and [Luo and Hauskrecht, 2017b]). The idea is to create a sample of training data with weights. The weight of each data instance reflects the proportion label of the group to which the instance belongs. For example, in a binary classification problem, each instance $\boldsymbol{x}_{ij}$ in a group (G_i, μ_i) is extended and duplicated into two triplets: $(\boldsymbol{x}_{ij}, y_{ij} = 1, w_{ij} = \mu_i)$ and $(\boldsymbol{x}_{ij}, y_{ij} = 0, w_{ij} = 1 - \mu_i)$, where the third element w is the weight of that triplet for training. With such a training sample, any model learning algorithm that permits data-weighting could apply. Weighted Logistic Regression and weighted SVM are representative examples.

2.3.4 Active Learning From Label Proportions

We hitherto have surveyed several lines of work that are associated with learning from group-based feedback. They are multiple instance learning (MIL), multiple instance active learning (MIAL), and learning from label proportions (LLP). We note that all of the work above does not tackle the problem that how groups are formed or labeled. This limits their applicability to many other learning tasks where no explicit groups are available. To bridge this gap, researchers propose to actively construct groups from instances for labeling and learning ([Du and Ling, 2010, Rashidi and Cook, 2011] and the work in this thesis). We name this direction of work *active learning from label proportions* or ALLP in short.

2.3.4.1 Motivation To describe groups to human annotators, ALLP uses *conjunctive patterns* that are value ranges over the input features. The use of conjunctive patterns is driven by the fact that sometimes labeling a group of examples on a higher abstract level appears easier, more friendly, and more effective than labeling individual instances. For example, in the medical domain, when a physician diagnoses a patient for a possible heart condition he/she must review the patient record that consists of complex collections of results, symptoms, and findings. The review and the assessment of these records concerning a specific condition may become extremely time-consuming, as it often requires physicians to peruse a large quantity of data. Below we borrow the heart-disease example introduced in [Rashidi and Cook, 2011] that compares instance-based feedback versus group-based feedback:

An instance query example: *"Assess whether the patient that is (sex=female) ∧ (age=39) ∧ (chest pain type=3) ∧ (fasting blood sugar=150 mg/dL) ... (20 more features) suffers from a heart disease?"*. The answer to this query is binary, reflecting a physician's diagnosis of this patient.

As we can see, each data record has numerous features and some of them are valued with high precision numbers that can be very intricate for human annotators to assess. By comparison, a group query that summarizes the relevant conditions of a subpopulation of patients reads more concise and informative.

Algorithm 2 Existing Group-Based Active Learning Frameworks: AGQ+ and RIQY

1: Collect a pool $\mathcal{U}$ of i.i.d. sampled data $\boldsymbol{x}$ that are all unlabeled
2: Also collect a small set of labeled data $\mathcal{L} = \{\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l)\}\}$
3: **repeat**
4: Learn the base classification model based $\mathcal{M}$ from $\mathcal{L}$
5: *Uncertainty sampling*: pick the most uncertain instance $\boldsymbol{x}^* \in \mathcal{U}$ based on $U_{\mathcal{M}}(\boldsymbol{x})$
6: Embody a small neighbor (group) of instances G^* that are centered on $\boldsymbol{x}^*$
7: Induce the most relevant *conjunctive patterns* that can best represent the group G^*
8: Have a human annotator to assign a proportion label μ^* to G^*
9: For each instance $\boldsymbol{x}$ in G^*, either give it a certain class label based on μ^* (RIQY) or directly use μ^* as a *soft* label of $\boldsymbol{x}$ (AGQ+)
10: Add all such labeled instances in G^* to $\mathcal{L}$
11: **until** certain stopping criterion is met

A group query example: *"What is the proportion of patients who are (sex=female) ∧ (40<age<50) ∧ (chest pain type=3) and (fasting blood sugar within [130,150] mg/dL) ... (not necessarily using all the features) suffer from a heart disease?"*. The answer to this query is an empirical assessment of the proportion, say, *"about 70% patients within this population have heart disease"*.

2.3.4.2 Early Work The idea of ALLP was first proposed by [Du and Ling, 2010] and [Rashidi and Cook, 2011]. They construct groups from the instances that are suggested to query by a conventional active learning strategy. In particular, they form each group as a compact set of instances that are centered around the most uncertain instance $\boldsymbol{x}^*$. They developed two similar methods **AGQ+** and **RIQY**, respectively, as summarized in Algorithm 2. Specifically, there are three steps: (1) embody a small group of nearest instances around $\boldsymbol{x}^*$ from a large pool of unlabeled data; (2) induce the most relevant region that best represents the group; (3) solicit from human annotators a class proportion label for that region. As for model learning, AGQ+ assigns probabilistic labels to instances and then adopts a weighted instance learning algorithm. RIQY instead propagates the major class

Table 2.1: Summary of Related Work.

	Instance-Based Feedback & Learning	Group-Based Feedback & Learning
Passively	Standard supervised learning § 2.1	MIL (§ 2.3.2) and LLP(§ 2.3.3)
Actively	Instance-based active learning § 2.2	MIAL (§ 2.3.2) and **ALLP**(§ 2.3.4)

label of a group to all the instances inside. As all the instance have deterministic labels, they can be fed to any instance-based learning algorithm.

2.4 Thesis Contributions

We would like to summarize the related work presented in this chapter and then place our work among them. We first introduced supervised learning where the training data are randomly (passively) generated. Then we compared it to active learning where the training data are strategically collected and models are trained incrementally. After that, we presented various types of alternative human feedback that are useful for label-efficient model learning. They were learning from soft-label feedback, multiple instance learning (MIL), and learning from label proportions (LLP). We also showed how each form of feedback could be actively acquired so as to further reduce human annotation effort. All of the work above can be categorized in Table 2.1:

The work in this thesis contributes to the research of ALLP. The main features of ALLP are:

1. The goal of ALLP is to learn instance-based classification models from group-based feedback.
2. ALLP deals with the problem of how to identify groups among instances that are important to model learning.
3. Groups constructed by ALLP are described to human annotators by using conjunctive patterns that correspond to data regions.

Existing work in ALLP is limited and there are only two methods, AGQ+ and RIQY, that have been developed. Their group-construction strategy is to form a group of instances that are the nearest neighbor of the most uncertain instance. However, we argue that their methods seem to be ad-hoc and may not be able to find meaningful groups. We think that meaningful groups should be (1) understandable to human annotators and (2) informative to model learning. Regarding (1), how do they control the size of the formed groups? And for (2), how do they ensure the returned group labels are pure enough? Unfortunately, a fact is that since the most uncertain instance often lies around the underlying decision boundary the instances around it probably belong to different classes. As a result, a group formed from such a set of instances is likely to have an impure proportion label. Therefore, we believe there should be a more principled and systematic way of identifying meaningful groups. This motivates our hierarchical solutions. In the following chapters, we will present three implementations of our hierarchical active learning idea: HALG (Chapter § 4), HALR (Chapter § 5), and HALOR (Chapter § 6).

3.0 Learning From Label Proportions

In this chapter, we present the learning from label proportions (LLP) algorithms since they lay the foundation of our framework. LLP manages to learn instance-based models solely based on labeled bags (in our case, groups and regions) of examples. Meanwhile, there are certain conditions that are needed to ensure the learning success. Those conditions will be the key factors that drive our active group-construction strategy.

3.1 Problem Setting

LLP applies to learning of multi-class classification models, but in this thesis we only consider its application to *binary* classification tasks. A binary classifier can be defined as one of the following cases: a probabilistic discriminant function $p(y|\boldsymbol{x};\boldsymbol{\theta})$; a general discriminant function $f(\boldsymbol{x};\boldsymbol{w}) : \mathcal{X} \to \mathbf{R}$; or a directly defined mapping $f(\boldsymbol{x};\boldsymbol{w}) : \mathcal{X} \to \mathcal{Y}$. Each $\boldsymbol{x}$ represents one instance from the input space $\mathcal{X} \subset \mathbf{R}^m$ and $y \in \mathcal{Y} = \{0, 1\}$ denotes the instance's label. The training data are N labeled bags $\{(B_i, \pi_i)\}_{i=1}^{N}$. Each bag $B_i = \{\boldsymbol{x}_{ij}\}_{j=1}^{n_i}$ contains n_i instances and has a label proportion $\pi_i \in [0, 1]$. Let us assume all π_i represents the proportion of the positive class, aka label 1. So we denote by $\pi_{iy} := \pi_i$ when $y = 1$ and $\pi_{iy} := 1 - \pi_i$ when $y = 0$.

As mentioned before, there are mainly four categories of algorithms and here we present the first two: a statistical learning framework and a general learning framework. The former uses proportion labels to approximate the sufficient statistics in a log-likelihood function [Quadrianto et al., 2009, Patrini et al., 2014]; the latter learns a classifier that generates matching instance labels of which the class proportions are close to the given proportions [Kück and de Freitas, 2005, Rueping, 2010, Yu et al., 2013, Yu et al., 2014]. For each of the frameworks, we will also discuss the assumptions that ensure its learning success. Basically, the key lies at (1) how the instances are distributed within the training bags, and (2) how the bags are distributed. In general, arbitrarily distributed instances or bags *may not* be

able to recover the underlying classification rules.

3.2 A Statistical Learning Framework

The basic idea of the statistical approach [Quadrianto et al., 2009, Patrini et al., 2014] is to estimate what is called a **mean map operator** that is sufficient to learn a classifier. We follow the work of [Quadrianto et al., 2009] to explain the idea. Assume the learning of a conditional exponential model:

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \exp(\langle\phi(\boldsymbol{x},y),\boldsymbol{\theta}\rangle - g(\boldsymbol{\theta}|\boldsymbol{x})) \tag{3.1}$$

where $\phi(\boldsymbol{x},y)$ is a feature mapping to a *Reproducing Kernel Hilbert Space* and $g(\boldsymbol{\theta}|\boldsymbol{x})$ is the log-partition function. If we *could* observe a set of independently and identically distributed training instances $(X,Y) = \{(\boldsymbol{x}_j,y_j)\}_{j=1}^n$ that are sampled from a distribution $p(\boldsymbol{x},y)$ on $\mathcal{X}\times\mathcal{Y}$, then the conditional log-likelihood function is:

$$\log p(Y|X,\boldsymbol{\theta}) = \sum_{j=1}^{n}\{\langle\phi(\boldsymbol{x}_j,y_j),\boldsymbol{\theta}\rangle - g(\boldsymbol{\theta}|\boldsymbol{x}_j)\} = n\langle\mu_{XY},\theta\rangle - \sum_{j=1}^{n} g(\boldsymbol{\theta}|\boldsymbol{x}_j) \tag{3.2}$$

where $\mu_{XY} = \frac{1}{n}\sum_{i=j}^{n}\phi(\boldsymbol{x}_j,y_j)$ is the *empirical* mean in the feature space. Notice that μ_{XY} is the *sufficient statistics* to the objective function and thus it makes LLP learning possible without knowing the labels of individual instances. In order to avoid over-fitting one commonly maximizes the log-likelihood penalized by a prior $p(\boldsymbol{\theta})$. Finally, the optimization problem becomes:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\left[\sum_{j=1}^{n} g(\boldsymbol{\theta}|\boldsymbol{x}_j) - n\langle\mu_{XY},\theta\rangle + \lambda\|\boldsymbol{\theta}\|^2\right] \tag{3.3}$$

However, μ_{XY} is unknown because we cannot observe the instance labels. But notice a fact that under mild conditions the empirical mean μ_{XY} is statistically well behaved and it converges to the population mean $\mu_{xy} \doteq \mathbf{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\phi(\boldsymbol{x},y)$ at rate $O(n^{-\frac{1}{2}})$. Hence, the solution is to estimate the population mean μ_{xy} first and use it as a proxy for μ_{XY}, and only then solve Equation (3.3) with the estimated $\hat{\mu}_{XY}$.

3.2.1 Estimation Of The Mean Operator

μ_{xy} can be decomposed into a sum of conditional expectations:

$$\mu_{xy} = \sum_{y \in \mathcal{Y}} p(y) \mu_x^{class}[y, y] \tag{3.4}$$

where:

- $p(y)$ is the prior distribution of $y \in \mathcal{Y}$. It is assumed known in that it can be seen as the class proportion of a special bag that contains the *whole* instance population.
- $\mu_x^{class}[y, y] = \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|y)} \phi(\boldsymbol{x}, y)$ is the conditional expectation. Quantity $\mu_x^{class}[y, y'] := \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|y)} \phi(\boldsymbol{x}, y')$ denotes the expectation of $\phi(\boldsymbol{x}, y')$ conditioning on $p(\boldsymbol{x}|y)$.

In order to compute $\mu_x^{class}[y, y]$, a bag-based quantity $\mu_x^{set}[i, y']$ is introduced which is the conditional expectation based on bag i:

$$\mu_x^{set}[i, y'] := \mathbf{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|i)} \phi(\boldsymbol{x}, y') \tag{3.5}$$

Note the distribution $p(\boldsymbol{x}|i)$ can be decomposed as:

$$p(\boldsymbol{x}|i) = \sum_{y \in \mathcal{Y}} p(\boldsymbol{x}, y|i) = \sum_{y \in \mathcal{Y}} [p(\boldsymbol{x}|y, i) p(y|i)] \tag{3.6}$$

where $p(y|i) := \pi_{iy}$ is observed. In order to proceed, a crucial assumption has to be made on $p(\boldsymbol{x}|y, i)$ - a conditional independence that $p(\boldsymbol{x}|y, i) = p(\boldsymbol{x}|y)$. In other words, they assume that the conditional distribution of $\boldsymbol{x}$ is independent of the bag index i, as long as the label y is known. After all, we want the distributions within each class to be independent of which bag they can be found in. If this were not the case it would be impossible to infer about the distribution on the test set from the (biased) distributions over the bags. *Such an assumption bridges the gap between the instance-based classifier to be learned and the bag-level labels that are given.*

Then, Equation (3.5) can be further derived as:

$$\mu_x^{set}[i, y'] \overset{(3.6)}{=} \sum_{y} \pi_{iy} \mu_x^{class}[y, y'] \tag{3.7}$$

It means that $\mu_x^{set}[i, y']$ is a linear combination of $\mu_x^{class}[y, y']$. So we rewrite Equation (3.7) into a linear equation system:

$$\boldsymbol{\mu}_x^{set} = \boldsymbol{\pi}\boldsymbol{\mu}_x^{class} \tag{3.8}$$

where $\boldsymbol{\mu}_x^{set}$, $\boldsymbol{\pi}$ and $\boldsymbol{\mu}_x^{class}$ are in matrix form. Then $\boldsymbol{\mu}_x^{class}$ is solved as:

$$\boldsymbol{\mu}_x^{class} = (\boldsymbol{\pi}^\top\boldsymbol{\pi})^{-1}\boldsymbol{\pi}^\top\boldsymbol{\mu}_x^{set} \tag{3.9}$$

Lastly, $\mu_x^{set}[i, y']$ is estimated by its empirical value $\mu_X^{set}[i, y']$, since that $\mu_X^{set}[i, y']$ also converges to $\mu_x^{set}[i, y']$ at rate $O(n_i^{-\frac{1}{2}})$:

$$\mu_X^{set}[i, y'] := \frac{1}{n_i}\sum_{\boldsymbol{x}\in B_i}\phi(\boldsymbol{x}, y') \tag{3.10}$$

Then, Equation (3.9) is rewritten as:

$$\hat{\boldsymbol{\mu}}_x^{class} = (\boldsymbol{\pi}^\top\boldsymbol{\pi})^{-1}\boldsymbol{\pi}^\top\boldsymbol{\mu}_X^{set} \tag{3.11}$$

Finally,

$$\hat{\mu}_{XY} = \sum_{y\in\mathcal{Y}} p(y)\hat{\mu}_x^{class}[y, y] \tag{3.12}$$

and we can plug it into the Equation (3.3) to perform the optimization procedure.

3.2.2 Assumptions Of The Statistical Learning Framework

Overall, the statistical learning algorithm works as follows: it uses the empirical means ($\mu_X^{set}[i, y']$) on the bags $\{B_i\}$ to approximate the expectations with respect to the bag distribution ($\mu_x^{set}[i, y']$); then it uses the latter to compute the expectations with respect to a given label ($\mu_x^{class}[y, y']$); finally, it uses the means conditional on the label distribution to obtain μ_{xy} that is a good proxy for μ_{XY}, i.e.

$$\mu_X^{set}[i, y'] \longrightarrow \mu_x^{set}[i, y'] \longrightarrow \mu_x^{class}[y, y'] \longrightarrow \mu_{xy} \longrightarrow \mu_{XY} \tag{3.13}$$

The middle two steps in the sequence follow from linear algebra. But more importantly, the first and last steps in the chain require *uniform convergence* results. The last convergence is relatively easy to satisfy because it does not involve bags:

$$\mu_{XY} := \frac{1}{n}\sum_{j=1}^{n}\phi(\boldsymbol{x}_j, y_j) \longrightarrow \mu_{xy} := \mathbf{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\phi(\boldsymbol{x}, y) \tag{3.14}$$

This convergence holds if (1) the instances are i.i.d. sampled from $p(\boldsymbol{x}, y)$ and (2) the feature map ϕ is a *Reproducing Kernel Hilbert Space* [Bartlett and Mendelson, 2002].
The first convergence in the chain is really the key:

$$\mu_X^{set}[i, y'] := \frac{1}{n_i}\sum_{\boldsymbol{x}\in B_i}\phi(\boldsymbol{x}, y) \longrightarrow \mu_x^{set}[i, y'] := \mathbf{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|i)}\phi(\boldsymbol{x}, y') \tag{3.15}$$

In order to make it happen, they have assumed the conditional independence on bags that was discussed earlier:

$$p(\boldsymbol{x}|i) = \sum_{y\in\mathcal{Y}}[p(\boldsymbol{x}|y, i)p(y|i)] = \sum_{y\in\mathcal{Y}}[p(\boldsymbol{x}|y)p(y|i)] \tag{3.16}$$

The assumption $p(\boldsymbol{x}|y, i) = p(\boldsymbol{x}|y)$ states that "*within each bag, the distribution of the instances conditioned on class should be identical to the unbagged instance distribution conditioned on class*". In other words, the conditional distribution of instances is independent of the bags as long as the class is known. Therefore, if bags and the inside instances are not formed in this way, they are not able to learn an instance-level model.

3.3 A General Learning Framework

The statistical LLP solution comes with many useful properties. Unfortunately, it is not easy to apply in practice because of the restrictions placed on the family of models it supports and somewhat rigid requirements on bag formation. In this section, we present another more general and flexible framework that learns various families of classification models.

3.3.1 The Framework

The idea of the approach is simple: *Find the best classifier that generates instance labels of which the class proportions are in agreement with the true proportions assigned to the training bags.* Let us assume an instance-based classifier $y = f(\boldsymbol{x}; \boldsymbol{w}) : \mathcal{X} \to \mathcal{Y} = \{0, 1\}$. For each bag B_i with a proportion class label π_i, the error of the classifier f on the the bag is:

$$\varepsilon_p(B_i, \pi_i; \boldsymbol{w}) = \left|\left| \frac{\sum_{j=1}^{n_i} f(\boldsymbol{x}_j; \boldsymbol{w})}{n_i} - \pi_i \right|\right|_p \tag{3.17}$$

where ε_p is based on norm p. Similarly, for probabilistic model we have:

$$\varepsilon_p(B_i, \pi_i; \boldsymbol{\theta}) = \left|\left| \frac{\sum_{j=1}^{n_i} p(y|\boldsymbol{x}_j; \boldsymbol{\theta})}{n_i} - \pi_i \right|\right|_p \tag{3.18}$$

Hence, the total loss function over all bags $L = \{(B_i, \pi_i)\}_{i=1}^{N}$ is:

$$\mathcal{L}(L; \boldsymbol{w}) = \sum_{i=1}^{N} \varepsilon_p(B_i, \pi_i; \boldsymbol{w}) \tag{3.19}$$

If we consider the sizes of the bags, as well as regularization term $R(\boldsymbol{w})$, finally the loss function becomes:

$$\mathcal{L}(L; \boldsymbol{w}) = \sum_{i=1}^{N} \frac{n_i}{n} \left[\varepsilon_p(B_i, \pi_i; \boldsymbol{w})\right] + R(\boldsymbol{w}) \tag{3.20}$$

Let us look at a specific example given by [Rueping, 2010] for learning a linear Support Vector Machine: $f(\boldsymbol{x}; \boldsymbol{w}) = \text{sign}(g(\boldsymbol{x}; \boldsymbol{w}))$ where $g(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$. With a scaling factor defined below, the probability of classifying $\boldsymbol{x}$ as $y = 1$ is:

$$p = \sigma(g(\boldsymbol{x}; \boldsymbol{w})) = \frac{1}{1 + \exp\{-g(\boldsymbol{x}; \boldsymbol{w})\}} \tag{3.21}$$

Then for any labeled bag (B_i, π_i), they try to fit the model consistent to the given label π_i:

$$\forall i : \frac{1}{n_i} \sum_{j=1}^{n_i} \sigma(g(\boldsymbol{x}_j; \boldsymbol{w})) \approx \mu_i \tag{3.22}$$

Therefore, the final optimization procedure is given by:

$$\text{minimize:} \tag{3.23}$$

$$\frac{\boldsymbol{w}^T \boldsymbol{w}}{2} + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \tag{3.24}$$

$$\text{subject to:} \tag{3.25}$$

$$\frac{1}{n_i} \sum_{j=1}^{n_i} \sigma(g(\boldsymbol{x}_j; \boldsymbol{w})) \geq \mu_i - \xi_i, \qquad \forall i \tag{3.26}$$

$$\frac{1}{n_i} \sum_{j=1}^{n_i} \sigma(g(\boldsymbol{x}_j; \boldsymbol{w})) \leq \mu_i + \xi_i^*, \qquad \forall i \tag{3.27}$$

$$\xi_i, \xi_i^* \geq 0, \qquad \forall i \tag{3.28}$$

Multiple research papers have developed and promoted the general LLP framework [Kück and de Freitas, 2005, Rueping, 2010, Yu et al., 2013]. They place no assumptions on the data and bags. Furthermore, the work of [Rueping, 2010, Yu et al., 2013] demonstrated empirically on a number of data sets that the general framework often learns better models compared to the statistical LLP approach by [Quadrianto et al., 2009].

3.3.2 Assumptions Of The General Learning Framework

Despite very few restrictions made to the framework, an apparent but challenging question would be, does this framework always work? Can it learn models from any set of arbitrarily formed bags? Unfortunately, the answer is negative. The work of [Yu et al., 2014] gives a few theoretical insights regarding this issue. We review their main findings in the following texts.

The goal of LLP learning is to learn a classifier that has a low *generalization error* when it is applied to the classification of any instance from the input domain. [Yu et al., 2014] first gives an error analysis of *bag-level generalization error*. Then, an analysis of *instance-level*

generalization error can be built upon it. The former tells when a model can achieve a low generalization error for predicting bag proportions; then given such a model, the latter tells the conditions that guarantee a low generalization error of that model for predicting instance labels.

3.3.2.1 Generalization Error For Predicting Bag Proportions Suppose there are N training bags $\{(B_i, \pi_i)\}_{i=1}^{N}$ that are generated i.i.d. from a probability distribution D over *bags*. For simplicity, assume each bag has the same number r of instances. Let $\mathcal{H}$ denote a hypothesis class for classifying instances where each $h \in \mathcal{H}$ is a mapping $h : \mathcal{X} \to \mathcal{Y}$. The learning task is to find the $h^* \in \mathcal{H}$ that best fits the training bag proportions:

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} \varepsilon(B_i, \pi_i; h) \tag{3.29}$$

where

$$\varepsilon(B_i, \pi_i; h) = \left| \frac{\sum_{j=1}^{r} h(\boldsymbol{x}_j)}{r} - \pi_i \right| \tag{3.30}$$

To bound the generalization error [Yu et al., 2014] build upon the results from Empirical Risk Minimization (ERM) framework. Let $err_S(h) = \frac{1}{N} \sum_{i=1}^{N} \varepsilon(B_i, \pi_i; h)$ denote the empirical bag proportion error and $err_D(h) = \mathbf{E}_{B \sim D} [\varepsilon(B, \pi; h)]$ the generalization error. One can prove the following theorem *sample complexity* bound for N:

Theorem 1. *for any* $0 < \delta, \epsilon < 1$ *and* $h \in \mathcal{H}$*, with probability at least* $1 - \delta$*,* $err_D(h) \leq err_S(h) + \epsilon$ *if:*

$$N \geq \frac{64}{\epsilon^2}(2\mathbf{VC}(\mathcal{H}) \log(12r/\epsilon) + \log(4/\delta)) \tag{3.31}$$

in which $\mathbf{VC}(\mathcal{H})$ *is the VC dimension of the instance label hypothesis class* $\mathcal{H}$*,* N *the number of training bags, and* r *the bag size.*

The above theorem shows that the generalization error of bag proportions can be bounded by the empirical proportion error if there are *sufficiently* many bags in training.

3.3.2.2 An Error Bound Of Instance Label Prediction The second step is to bound the generalization error for predicting instance labels by the generalization error of predicting bag proportions. Key to this bridging is the concept *purity* of bags. Intuitively, if a classifier h can predict bag proportions very well, then it should also predict well the instance labels that come from *pure* bags. For example, bags with proportions 0 or 1 are the easiest cases. Formally, suppose one has learned a classifier h that can predict bag proportions well: $P_{B\sim D}(\varepsilon(B,\pi;h)\leq\epsilon)\geq 1-\delta$. Define the purity of bags: for $0<\eta<1$, a bag is $(1-\eta)$ *pure* if at least a fraction $(1-\eta)$ of all instances have the same label. Then the following theorem states:

Theorem 2. *Let h be a hypothesis satisfying $P_{B\sim D}(\varepsilon(B,\pi;h)\leq\epsilon)\geq 1-\delta$ for some $0<\epsilon,\delta<1$. Assume the probability that a bag is $(1-\eta)$ pure is at least $1-\rho$ for some $0<\eta,\rho<1$. Then with probability at least $(1-\eta-\rho)$, h classifies correctly at least a fraction $(1-2\delta-\epsilon)$ of instances in that bag.*

Therefore, the *purity* of bags is the key factor in predicting the instance labels. Furthermore:

Lemma 3. *There exists a distribution D over all bags of size r and a learner h such that $P_{B\sim D}(\varepsilon((B,\pi);h)=0)=1$, and each bag is $(1-\eta)$-pure but h mis-classifies a fraction 2η instances of each bag.*

The above lemma shows an extreme case when LLP fails. That is, there exist bags with label proportion 50% (they are the least pure), and a hypothesis h which can achieve *zero* bag proportion prediction error, yet with 100% instance label error. In other words, it is impossible to learn or recover the instance labels from such bags.

3.4 Chapter Summary

We have presented two different frameworks for LLP learning. The first one is a statistical learning approach and it uses proportion labels to approximate the sufficient statistics of a likelihood function that does not require instance labels. To guarantee the learning success,

there are two conditions. First, there should be sufficiently many bags and instances (within each bag) that are i.i.d. sampled for training. This condition is required by the law of uniform convergence. And the second condition is the conditional independence $p(\boldsymbol{x}|y,i) = p(\boldsymbol{x}|y)$ where i is the index of a training bag. This restricts the way the bags should be formed.

The second framework is a general one that applies to the learning of various types of classification models. However, to ensure the learning success, its analysis states: in order to learn a good instance-based model from bags, one needs to train the model with *sufficiently many pure regions*. This conclusion is the key that drives our active group-construction strategy. That is, we should identify pure groups out of a given set of instances as quickly as possible.

4.0 Hierarchical Active Learning From Group Proportion Feedback: HALG

4.1 Introduction

HALG [Luo and Hauskrecht, 2018a] (**H**ierarchical **A**ctive **L**earning with **G**roup proportion feedback) is our first attempt to implement the hierarchical active learning idea. Speaking at a high level, this implementation relies on a pre-compiled hierarchical clustering that guides us in identifying groups. Figure 4.1 illustrates an example of hierarchical clustering. At the beginning of this framework, a hierarchical clustering is done on a large pool of unlabeled instances and outputs a hierarchy of clusters (unlabeled groups). Then proceeding from the top levels to the lower ones, we actively select the most *influential* groups to be labeled. The influence is measured by how much the groups will update the base classification model once they get labeled. This active selection strategy borrows the merit of *maximum model change* criterion [Roy and McCallum, 2001, Freytag et al., 2014] as we introduced in Section § 2.2.2.

4.1.1 Framework Overview

HALG is designed to learn a binary classification model $P(y|\boldsymbol{x};\boldsymbol{\theta})$ from group proportion feedback. It forms a hierarchical tree of unlabeled groups first and then iteratively queries the proportion label of groups in a top-down manner, behaving like a breadth-first search. So we are maintaining a fringe of labeled groups for learning. In each iteration, we actively choose a group in the fringe to split, query its child groups, and then update the fringe. Every time the fringe is refined, the base classification model $P(y|\boldsymbol{x};\boldsymbol{\theta})$ will be re-trained as well. The framework overview is summarized as in Algorithm 3. The following three sections will detail the implementation of HALG. The next section would introduce the basic group concepts; the second section will present our group learning algorithm that learns an instance-based classifier from labeled groups; the third section will explain our active learning strategy for selecting groups for labeling.

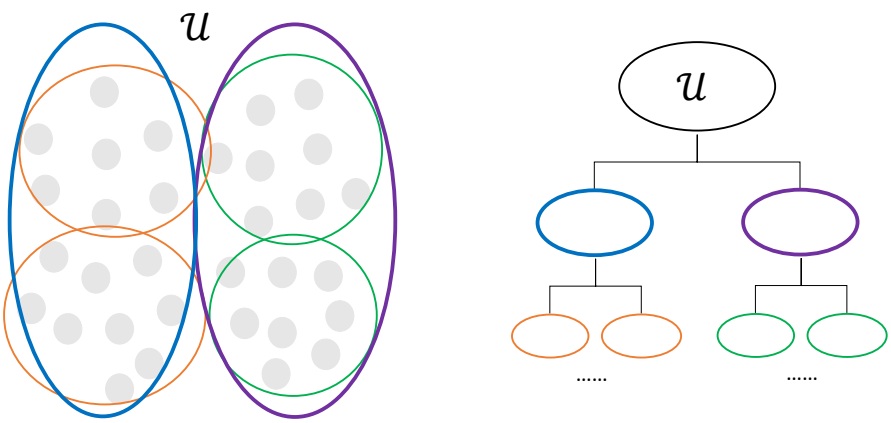

Figure 4.1: Hierarchical clustering used in HALG. It shows the top 3 levels of the cluster hierarchy ($\mathcal{U}$ is a large pool of unlabeled instances and also represents the root cluster). In general, hierarchical clustering is one clustering algorithm that groups similar objects into clusters. The endpoint is a set of clusters where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other. One essential distinction of hierarchical clustering is that it outputs a hierarchy of clusters, and one can choose different granularity of clustering by pruning the hierarchy at different levels.

4.2 Group-Related Concepts

- **Group.** We start with collecting a pool of abundant unlabeled data instances $\mathcal{U} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ where each $\boldsymbol{x}_i$ is a vector that is randomly sampled from an input space $\mathcal{X} \subset \mathbf{R}^m$. Then, we use the term *group* to denote a set of instances $G \subset \mathcal{U}$. Although a group G could be any subset of $\mathcal{U}$, we prefer G to be a set of similar instances such that it could be compactly represented to humans.
- **The Initial Tree of Clusters.** After collecting $\mathcal{U}$, we then perform standard hierarchical clustering (using the ward linkage [Ward Jr, 1963]) on $\mathcal{U}$ to output a tree T of clusters. However, it is possible (and sometimes likely) that some of the clusters could be arbitrarily shaped which makes it hard for humans to assess. Therefore, we need to modify the tree to only preserve clusters that can be *compactly* represented to humans. How do we define the *compactness*? RIQY [Rashidi and Cook, 2011] has provided a so-

Algorithm 3 The HALG Framework

Input: An unlabeled data pool $\mathcal{U}$; A labeling budget

Output: A binary classification model $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$

1: $T \leftarrow$ Perform hierarchical clustering on $\mathcal{U}$
2: $T_G \leftarrow$ Adjust T to form a new tree of groups
3: Query $(T_G)'s\ root$;
4: Fringe $F^{(1)} \leftarrow \{(T_G)'s\ root\}$;
5: Active learning time $t \leftarrow 1$;
6: **repeat**
7: Learn $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$ from current $F^{(t)}$
8: Split a group G_* in $F^{(t)}$ based on $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$
9: Query the labels of G_*'s children from labelers
10: $F^{(t+\Delta t)} \leftarrow \{F^{(t)} - G_*\} \cup \{G_*\text{'s children}\}$
11: $t \leftarrow t + \Delta t$ ($\Delta t = \#$ of G_*'s children)
12: **until** the labeling budget runs out
13: **return** $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$

lution that is to use *conjunctive patterns* to represent groups, and we adopt this solution in our work as well.

- **Group Description.** Conjunctive patterns are a set of values ranges over the input space features that are joint by *and* operation ($\wedge$). As we have seen earlier, for example, a group of patient instances may be described as: *"(sex=female) $\wedge$ (40<age<50) $\wedge$ (chest pain type=3) $\wedge$ (fasting blood sugar within [130,150] mg/dL) ... (not necessarily using all the features)"*. This representation matches precisely the hypercube definition of *regions* of a typical decision tree learning algorithm. Hence, we employ a C4.5 [Quinlan, 2014] classifier to automatically learn a compact description of the cluster of instances that are found through hierarchical clustering. With more details, if we want to learn the description of a cluster G, we mark all the instances in G as **1** and the rest of data instances $(\mathcal{U} - G)$ as **0**. Then a C4.5 classifier will output a set of hypercubes $\mathcal{C}(G)$ that could potentially describe G. The match (or fit) of each hypercube $c \in \mathcal{C}(G)$ to G can

be assessed in terms of (1) *precision* which measures the proportion of data in c that are actually coming from G, and (2) *recall* which measures the proportion of data in G that c can capture. As both metrics are important, we adopt $F1score = \frac{2 \times precision \times recall}{precision + recall}$ to be the final quality metric to select the hypercube that best fits the description of cluster G. That is, the description of G is the best-fit hypercube:

$$\arg \max_{c \in \mathcal{C}(G)} F1score(c) \tag{4.1}$$

- **The Final Tree of Groups.** When C4.5 algorithm is used to learn the hypercubes that represent the clusters in the original hierarchical tree T, there may be some clusters that are not matched well by hypercube-shaped regions. In such cases, their best-matched hypercubes may come with intolerably low $F1scores$. To mitigate this issue, we modify the original tree structure T to form a new tree T_G such that only well-fitted hypercube-like groups are preserved in T_G. More formally, we say that a cluster is *hypercube-like* if it can be approximated by hypercube with a minimum $precision(\geq 0.5)$ and $recall(\geq 0.5)$. Our goal is to preserve and approximate only hypercube-like regions in the original tree. We implement this idea by starting from the root of the tree T and by checking in a top-down fashion if the descendant clusters are hypercube-like. If a descendant cluster is not hypercube-like we exclude it from the tree by directly reconnecting its parent with its children clusters. As a result, the original binary tree T may become a multi-nary tree T_G in which the clusters are all hypercube-like. We call T_G a tree of groups and use it for the subsequent active learning process.
- **Group Proportion Feedback.** The human assessment of each group in T_G is made via a proportion label that is an estimate of the proportion of instances in the group that belong to one of the classes. For example, people could say that *"90% of instances in a certain group are positive"*; or alternatively, we can interpret the proportion label as an instance-level likelihood. For instance, *"Instances in such group are 90% likely to be positive"*. To assess the label of each group, annotators will only need to review the description of the best hypercube-like region that matches it, and thus they do not have to explore individual data instances in the group.

- **The Fringe of Groups.** After we form the group hierarchy T_G, a top-down active learning process begins. In each active learning cycle, we maintain a fringe F of labeled groups that is a complete partition of all the unlabeled data $\mathcal{U}$. Initially, we make one query to obtain the label of (T_G)'s root that can be interpreted as the prior probability of classes, and put the labeled root into $F^{(t)}$ at $t = 1$. Here t is the active learning time-step, basically counting the number of group queries made so far. As t increases, finer and finer groups and their proportion labels will replace their parents in $F^{(t)}$.

In the following sections, we explain how to learn a model from labeled groups in $F^{(t)}$ and how the model will assist us in choosing the group that should be split next.

4.3 Learning From Group Proportion Feedback

Our goal is to learn from $F^{(t)}$ a classification model $P(y|\boldsymbol{x};\boldsymbol{\theta})$ that is an instance-level discriminative classifier. We call it a *base* model. Suppose there are N labeled groups in the fringe: $F^{(t)} = \{(G_i, \mu_i)\}_{i=1}^{N}$, where G_i denotes a group and μ_i its proportion label. Each group $G_i = \{\boldsymbol{x}_{ij}\}_{j=1}^{n_i}$ contains n_i instances and its label $\mu_i \in [0, 1]$ is assumed to represent the *positive* class proportion in a binary classification setting. Our learning algorithm does this by (1) sampling sufficiently many labeled instances from the labeled groups, and (2) feeding them to any instance-level classification learning procedure.

4.3.1 A Sampling Procedure Of Instance Labels

We create a bootstrap sample $S^{(t)} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{K}$ of K labeled instances from $F^{(t)}$. Each $\boldsymbol{x}_k$ is uniformly sampled with replacement from the unlabeled data pool $\mathcal{U}$, and y_k is sampled from an independent Bernoulli process with parameter equal to μ_i which is the proportion label of group G_i that $\boldsymbol{x}_k$ resides in.

4.3.2 A Learning Algorithm: Maximum Likelihood Estimation

With the sample $S^{(t)}$, we can estimate the model parameter vector as $\hat{\boldsymbol{\theta}}^{(t)}$ through maximum likelihood estimation (MLE). Although $\hat{\boldsymbol{\theta}}^{(t)}$ may vary because of the randomness in $S^{(t)}$, from the theoretic standpoint, when moderate assumptions are satisfied (those required by *Uniform Convergence* and *Uniform Central Limit Theorem* [Neter et al., 1996, Van der Vaart, 2000]), the MLE solution $\hat{\boldsymbol{\theta}}^{(t)}$ asymptotically follows a normal distribution $\{\boldsymbol{x}_k\}^{(t)}$:

$$\hat{\boldsymbol{\theta}}^{(t)} \xrightarrow{D} \mathcal{N}(\boldsymbol{\theta}^{(t)}, \boldsymbol{\Sigma}^{(t)}) \tag{4.2}$$

Here $\boldsymbol{\theta}^{(t)} = \mathbb{E}[\hat{\boldsymbol{\theta}}^{(t)}]$ is the converged parameter when $K \to \infty$, and the variance $\boldsymbol{\Sigma}^{(t)}$ is the inverse of Fisher information matrix $\mathcal{I}_K(\boldsymbol{\theta}^{(t)})$ combined with the actual finite sample size K. So the randomness of $\hat{\boldsymbol{\theta}}^{(t)}$ is bounded by this asymptotic normal distribution and the larger K is, the smaller the variance would be.

In practice, however, $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\Sigma}^{(t)}$ are unknown. Yet, usually $\boldsymbol{\Sigma}^{(t)}$ is approximated by the the MLE estimator as $\hat{\boldsymbol{\Sigma}}^{(t)}$ (for example, when calculating the confidence interval of $\hat{\boldsymbol{\theta}}^{(t)}$). Furthermore, when K is large, the difference between $\hat{\boldsymbol{\theta}}^{(t)}$ and $\boldsymbol{\theta}^{(t)}$ is sufficiently small (or equivalently, the confidence interval of $\hat{\boldsymbol{\theta}}^{(t)}$ is very tight). So as another approximation, we replace $\boldsymbol{\theta}^{(t)}$ by $\hat{\boldsymbol{\theta}}^{(t)}$ in the normal distribution. That is, it is appropriate to assume the following asymptotic distribution holds when K is sufficiently large:

$$\hat{\boldsymbol{\theta}}^{(t)} \xrightarrow{D} \mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\Sigma}}^{(t)}) \tag{4.3}$$

In our experiments, each label is sampled from 5 to 10 times depending on data sets and this gives us $K \sim 10^4$ magnitude which is large enough to provide a very small $\hat{\boldsymbol{\Sigma}}^{(t)}$.

4.4 Active Refinement Of Groups

In this section, we explain our active learning strategy that refines the fringe $F^{(t)}$ at time t to generate $F^{(t+\Delta t)}$, where Δt is the number of new queries made. The gist of our approach

is to split the most influential group $G_* \in F^{(t)}$ w.r.t. updating the base model, query the labels of its child groups, and then replace G_* with its child groups in the fringe.

Intuitively, we want to split a large and/or impure group such that the label uncertainty of large groups can be reduced. Moreover, we would also like the current model $\boldsymbol{\theta}^{(t)}$ to have a big update after the split such that it converges quickly to $\boldsymbol{\theta}^{(\infty)}$. Therefore, we adopt *maximum model change* criterion as our active learning strategy. The key idea is to split the group such that the model distribution $\mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\Sigma}}^{(t)})$ can be updated most from time t to $(t + \Delta t)$. To achieve this goal, we need to *guess* what would happen after we split each group. More specifically, we need to (1) infer the most probable labels of each G_i's child groups and (2) estimate how much the model distribution will change if we replace G_i with its child groups (with inferred labels) in the fringe for learning.

4.4.1 Group Label Inference

One reasonable way to infer the label of a child group is to use the empirical mean statistics of all the instances inferred by the base model. Formally, let us suppose each group G_i in the current fringe $F^{(t)}$ has C_i child groups and each child group $G_{ic} = \{\boldsymbol{x}_{(ic)j}\}_{j=1}^{n_{ic}}$ has n_{ic} many instances for $c \in [1, C_i]$. The label of each child group $\hat{\mu}_{ic}$ can be inferred as:

$$\hat{\mu}_{ic} = \frac{1}{n_{ic}} \sum_{j \in [1, n_{ic}]} P(y_{(ic)j} = 1 | \boldsymbol{x}_{(ic)j}; \hat{\boldsymbol{\theta}}^{(t)}) \tag{4.4}$$

Then we can create a new fringe $F_{[i]}^{(t)} = \{F^{(t)} - (G_i, \mu_i)\} \cup \{(G_{ic}, \hat{\mu}_{ic})\}_{c=1}^{C_i}$, feed it to our parameter optimization algorithm, and obtain a new model distribution denoted by $\mathcal{N}(\hat{\boldsymbol{\theta}}_{[i]}^{(t)}, \hat{\boldsymbol{\Sigma}}_{[i]}^{(t)})$. So this $\mathcal{N}(\hat{\boldsymbol{\theta}}_{[i]}^{(t)}, \hat{\boldsymbol{\Sigma}}_{[i]}^{(t)})$ represents what the new model distribution would look like if we *were* to split the group G_i. Before we compare $\mathcal{N}(\hat{\boldsymbol{\theta}}_{[i]}^{(t)}, \hat{\boldsymbol{\Sigma}}_{[i]}^{(t)})$ to $\mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\Sigma}}^{(t)})$, one important note is that we should fix $\{\boldsymbol{x}_k\}$ in the sample $S^{(t)}$ for re-learning, as the two asymptotic normal distributions are comparable only if they are learned conditioned on the same $\{\boldsymbol{x}_k\}$. So when learning $\mathcal{N}(\hat{\boldsymbol{\theta}}_{[i]}^{(t)}, \hat{\boldsymbol{\Sigma}}_{[i]}^{(t)})$, only the labels in group G_i are re-sampled.

4.4.2 Model Change Measurement

In terms of estimating the model change, we use KL-divergence to measure the distribution change from the current model distribution $\mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\Sigma}}^{(t)})$ to $\mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}_{[i]}, \hat{\boldsymbol{\Sigma}}^{(t)}_{[i]})$ for each group G_i. Finally, we select the group G_* with the maximum change to split:

$$G_* = \arg\max_{G_i \in F^{(t)}} D_{KL}(\mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}_{[i]}, \hat{\boldsymbol{\Sigma}}^{(t)}_{[i]}) \;||\; \mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\Sigma}}^{(t)})) \tag{4.5}$$

After the split, G_*'s children $\{G_{*c}\}$ are sent for querying and a new fringe is updated as:

$$F^{(t+\Delta t)} = \{F^{(t)} - (G_*, \mu_*)\} \cup \{(G_{*c}, \mu_{*c})\}_{c=1}^{C_*} \tag{4.6}$$

A minor caveat when calculating the model change is that when the number of groups in $F^{(t)}$, i.e. N, is large, it may be time-consuming to solve the N optimization procedures sequentially. However, we note that as the these N procedures are independent, parallel processing can be appropriately deployed to reduce the total runtime. In the next chapter, we will provide the runtime comparison between HALG and HALR.

4.5 Experiments

We conduct an empirical study to evaluate our proposed approach HALG on 9 general binary classification data sets collected from UCI machine learning repository [Asuncion and Newman, 2007]. The main purpose of this study is to research how efficiently (in terms of the number of queries) our HALG framework can learn classification models in cost-sensitive tasks. Besides, we will also care about how complex the group queries would be, in terms of how many features are used in their conjunctive patterns.

Table 4.1: 9 UCI data sets.

Dataset	# of Data	# of Features	Major Class %	Feature Type
Seismic	2584	18	93%	Num, Ord, Cat
Ozone	1847	72	93%	Numeric
Pima	768	8	65%	Num, Categorical
Spam	4601	57	60%	Numeric, Ordinal
Music	1059	68	53%	Numeric
Wine	4898	11	67%	Numeric
Wine_{ub}	1895	11	95%	Numeric
Gamma	5000	10	65%	Numeric
SUSY	5000	18	55%	Numeric

4.5.1 Data Sets

The 9 data sets come from a variety of real life applications:

- **Seismic**: Predict if seismic bumps are in hazardous state
- **Ozone**: Detect ozone level on some days
- **Pima**: Diagnose diabetes disease among Indian women
- **Spam**: Classify spam commercial emails
- **Music**: Find the geographical origin of music
- **Wine**: Predict wine quality
- **Gamma**: Detect γ particles in Cherenkov telescope
- **SUSY**: Distinguish a signal or background process

Table 4.1 summarizes the basic statistics of the data sets. Some have been widely used in the previous active learning work: `Wine` [Nguyen et al., 2014, Xue and Hauskrecht, 2017a] and `Pima` [Rashidi and Cook, 2011]; some are high-dimensional: `Ozone, Spam, Music`; some carry highly unbalanced class distribution: `Seismic, Ozone`, `Wine unbalance` (simulated from `Wine`).

4.5.2 Baselines

We compare our HALG to three baselines:

1. **DWUS**: Density-Weighted Uncertainty Sampling which combines uncertainty sampling and the structure in data [Settles, 2012] to decide queries;
2. **RIQY**: the state-of-the-art active learning with group proportion feedback [Rashidi and Cook, 2011];
3. **HS**: Hierarchical Sampling [Dasgupta and Hsu, 2008]

4.5.3 Experimental Settings

4.5.3.1 Data Split To run the experiments, we split each data set into three disjoint subsets: the initial labeled data set (about 1%-2% of all available data), a test data set (about 25% of data) and a training data set $\mathcal{U}$ (the rest of the data) that is used for training and active learning. Please note only DWUS and RIQY require the initial labeled data to start training.

4.5.3.2 Group Proportion Label Feedback To mimic the human labeling process, we use empirical labeled instances to simulate group proportion feedback, as if it were given by humans. That is, to obtain the proportion label for a group G, we count the empirical instances and their labels in G to estimate the class proportions. We note that the same method was applied to test RIQY in its original paper.

4.5.3.3 Evaluation Metrics We adopt Area Under the Receiver Operating Characteristic curve (AUC) to evaluate the quality of the learned classification model (in our case Logistic regression) on the test data. Our graphs plot the AUC scores sequentially as the number of queries gradually increases to 200. All results are averaged over 20 runs in different random splits. When generating the results we also assumed the different types of queries are equivalent in terms of their costs, although in reality different query types may carry different costs. For example, a group query may be easier for objects like medical records and patient data, but not for images. However, we note that in reality different query types

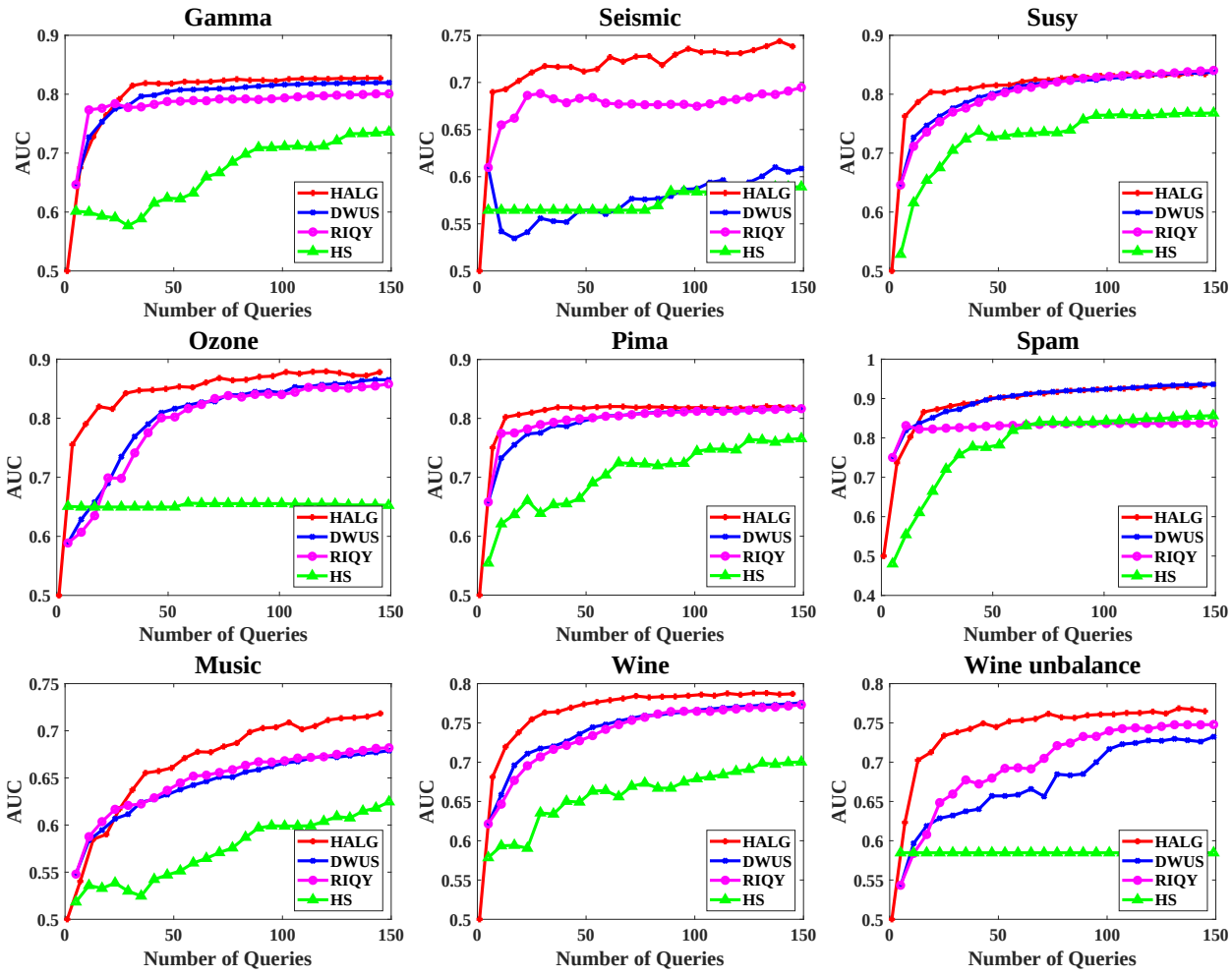

Figure 4.2: Performances of different methods on 9 UCI data sets.

may carry different costs. In some cases, say when presenting and labeling images, instance queries are much easier to assess. On the other hand, a group query may be easier for objects like medical records and patient data.

4.5.4 Experiment Results

The main results are shown in Figure 4.2. Overall, our HALG approach (in red line) is able to outperform other methods on the majority of the data sets and is close to the best method on the remaining sets. It comes with two advantages. First, initially when the labeling budget is severely limited, learning with labeled groups is superior to learning with

Table 4.2: The averaged feature reduction rate (FRR) of group queries.

Dataset	**FRR**	**Dataset**	**FRR**
Wine	42%	Spam	60%
Ozone	89%	Music	88%
Gamma	34%	SUSY	61%
Seismic	61%	Pima	40%

the same number of labeled instances, simply because generic group queries can provide richer class information than specific instance queries. Second, the initial steep slopes and early convergence in our learning curves lend great credence to our active learning strategy that it is capable of selecting the most influential group to split; consequently, it can accelerate the convergence rate of the method.

Although here we only show the performance of HALG after a finite number of queries, asymptotically it is able to converge to whatever an instance-based method would converge. It is due to the fact that at the very bottom level of the hierarchical tree, the groups are all singleton groups and their labels degenerate to instance labels. Hence, when the number of queries goes to infinite, groups will shrink to instances, and thus learning from those labeled groups would be equivalent to learning from labeled instances.

4.5.4.1 Unbalanced Classes For data sets `Seismic, Ozone` and `Wine unbalance` with unbalanced class distribution, our method performs even better as it could capture properly the minority class information via proportion labels. In contrast, the instance-based methods may find these proportions very slowly. Also note that hierarchical sampling (HS) completely failed because it always determines the labels of unlabeled instances by the majority vote if they belong to *pure enough* (but not entirely pure) clusters, and hence it may miss to capture the minority class information.

4.5.4.2 Complexity Of Group Queries

Our last experiment aims to analyze the benefit our group queries in terms of the query complexity. We assess this complexity by using *feature reduction rate* which is defined as:

$$1 - \frac{\text{the number of features used to describe } G}{\text{the total number of features}} \tag{4.7}$$

This definition clearly reflects the savings due to the description of the group G relative to the complexity of the full feature space. The results in Table 4.2 suggest that on average, it only takes about one third to one half of features to distinguish one group from the other groups. This can considerably simplify the interaction with human annotators especially when data objects are high-dimensional and when the active learning queries need to present only the features relevant for the group and its query.

4.6 Chapter Summary

In this chapter, we presented HALG, a hierarchical framework that actively learns instance-based classification models from group proportion feedback. In particular, we have addressed three problems that are related to groups:

1. Q: **How do we identify groups for labeling and learning?**
   A: We leverage hierarchical clustering to guide us finding potential groups. Following a pre-compiled hierarchy of unlabeled groups in a top-down fashion, we use *maximum model change* strategy to actively choose groups for labeling and learning.
2. Q: **How do we present groups to human annotators for assessment?**
   A: Describe groups by conjunctive patterns that are value ranges over the input space features.
3. Q: **How do we learn instance-based models from group proportion feedback?**
   A: Sample sufficiently many labeled instances from labeled groups and feed them to *maximum likelihood estimation* algorithm.

In terms of application, our framework is best suited to learning scenarios when instance labeling is hard but assessing a group of instances is more feasible and easier. According to our experiment results, when models are fed with the same number of group queries or instance queries, actively learning classification models from group proportion feedback is able to train more accurate models than from traditional instance feedback. Hence, the results are promising in supporting our first hypothesis **H1** (Section § 1.4). Besides, our method also outperforms RIQY which is the state-of-the-arts active learning framework based on group proportion feedback. The results lend credence to the fact that our hierarchical way of forming groups can identify meaning groups more quickly. Therefore, the results also provide positive evidence to our second hypothesis **H2** (Section § 1.4).

5.0 Hierarchical Active Learning From Region Proportion Feedback: HALR

5.1 Introduction

In the previous chapter, we presented HALG, the first implementation of our hierarchical active learning framework. We have seen that HALG is able to identify good groups based on hierarchical clustering. However, we also notice that it is hierarchical clustering itself, which gives us a *pre-compiled* and a *fixed* hierarchy of groups, that limits its potential of finding even more meaningful groups. The reason is that the group hierarchy may be completely irrelevant to the underlying class distribution since it is formed *unsupervisedly* (clustering). The consequence would be that many of those groups formed by hierarchical clustering can still have very impure proportion labels (i.e. close to 0.5). Hence, fixed group hierarchy may not help at all improve the learning efficiency.

Once we have accepted that groups can be described by conjunctive patterns that correspond regions in the input data space, we can *directly* and *dynamically* split the input space into smaller sub-regions for querying and learning. In this way, we are able to explore more diverse regions that can be more relevant to the class distribution among instances; that is, identify *purer* regions in *fewer* iterations, and hence accelerate the whole active learning process.

To implement this new intuition, we develop a *region-based* active learning framework called HALR [Luo and Hauskrecht, 2018b, Luo and Hauskrecht, 2019] (**H**ierarchical **A**ctive **L**earning with proportion feedback on **R**egions) that learns classification models from region proportion feedback. Briefly, our framework can actively build a hierarchical tree of regions with the aim to refine the leaf regions to be as *pure* as possible after very *few* splits and queries made. More specifically, our HALR starts from an unbounded region that covers the entire input feature space $\mathcal{X}$ and this region initializes as the root of the tree. Then we grow this tree incrementally by splitting the most *uncertain* leaf region into two sub-regions. Whenever the new regions are generated, their proportion labels are either directly assigned by a human annotator or inferred by the proportion constraint. The general picture

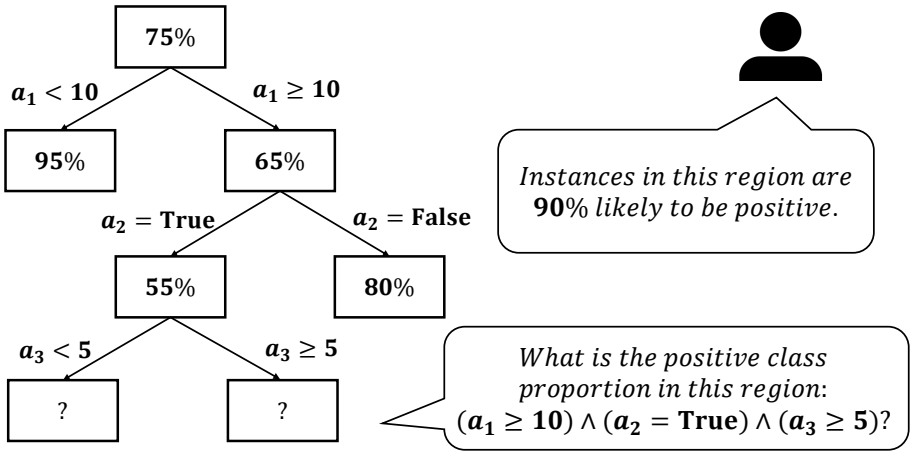

Figure 5.1: An example of building a hierarchical tree of regions which is conceptually equivalent to a decision tree. The left shows a snapshot of the tree structure after $t = 3$ splits, generated from the root region on the top level. Each rectangle represents a region and the percentage number means its proportion label (for positive class). Each link is a value constraint on some dimension a_i and is inherited to all the descendant regions. To query the proportion label of a new region (say the right one on the lowest level), we describe it by using conjunctive patterns shown on the bottom right, and a human annotator will assign a label to it according to its description. The label of the complementary region (the one on the left) will be inferred according to the constraint between its parent's and sibling's labels.

is illustrated in Figure 5.1. In the end, our algorithm outputs a hierarchical tree of labeled regions that can be either (1) directly used as a decision tree classifier, or alternatively, (2) be used to learn many other classification models by learning from label proportion (LLP) algorithm (Chapter § 3) or by the sampling-based learning algorithm proposed in HALG (Section § 4.3).

The crucial part of our algorithm is to develop a strategy that splits the leaf regions *without* knowing any labeled instances. To meet this challenge, we propose two instance-based splitting heuristics that use *approximated* instance labels for splitting regions. The first one is a *supervised* heuristic that relies on the base classification model we aim to learn. The second one is an *unsupervised* heuristic that uses k-means clustering. Which heuristic should

be chosen to split a specific region is a key question. We have implemented two different procedures to solve this problem. In [Luo and Hauskrecht, 2018b], we design a competition procedure which dynamically tests the two heuristics and chooses the better one (Section § 5.4); later in [Luo and Hauskrecht, 2019], we develop a more efficient procedure which is based on Multi-Arm Bandits (MAB) algorithms (Section § 5.5). We will detail and compare them in the following sections.

5.1.1 Framework Overview

Overall, the HALR framework is summarized in Algorithm 4. The goal of HALR is still to learn a binary classification model $P(y|\boldsymbol{x};\boldsymbol{\theta})$ from region proportion feedback. Compared to HALG, there are two improvements in HALR. First, the hierarchy of regions is formed dynamically. It is explicitly controlled by a supervised heuristic and an unsupervised heuristic. The second is on active learning strategy. In HALG, we adopted *maximum model change* which selects groups that could potentially improve the base model most. As we noted in the last paragraph in Section § 4.3, this strategy is computationally expensive. In HALR, we design a simple yet effective procedure for active region selection and split. This is done in two steps: (1) choose the most *uncertain* region (Section § 5.2), and (2) actively split it into two sub-regions. As mentioned, there are two different procedures to determine the split dimension and value. One is a a competition procedure (Section § 5.4) and the other is a more advanced one based on a MAB algorithm (Section § 5.5).

5.2 Uncertainty Of Regions

A region, which is defined by conjunctive patterns, represents a hypercubic sub-space in the input space $\mathcal{X}$. It may also contain a set of empirically instances that belong to $\mathcal{U}$. A region is assessed by human annotators who return a class proportion label to it.

Given the definition of regions we now want to define a score that would help us decide which region should be split next in each active learning cycle. One sensible way is to use

Algorithm 4 The HALR Framework

Input: An unlabeled data pool $\mathcal{U}$; A labeling budget

Output: A binary classification model $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$

1: $T \leftarrow$ Build a 1-node tree whose root region is the entire feature space of $\mathcal{U}$;
2: Query the proportion label of T's root;
3: Leaf nodes $L^{(1)} \leftarrow \{T$'s root$\}$;
4: Active learning time $t \leftarrow 1$;
5: **repeat**
6: Train the base model $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$ with current leaf nodes $L^{(t)}$;
7: Choose a most *uncertain* region R_* in $L^{(t)}$ to be split;
8: Divide R_* into two sub-regions (it is co-decided by probabilistic clustering and probabilistic classification (based on $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$);
9: Query or infer the proportion labels of the sub-regions derived from R_*;
10: $L^{(t+1)} \leftarrow \{L^{(t)} - R_*\} \cup \{R'_*s$ sub-regions$\}$;
11: $t \leftarrow t + 1$
12: **until** the labeling budget runs out
13: **return** $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$

the uncertainty (or impurity) of regions. This idea has been successfully used in decision tree learning process. Here, the impurity is measured in terms of the entropy (C4.5) or the Gini-Index (CART) scores. With the help of the impurity measure one can build a decision tree recursively where in each step one leaf region is split along one of the input dimensions. By comparing all possible splits for all eligible leaf regions, the best region and the best split that leads to the maximum reduction in uncertainty, or the maximum information gain, can be identified. Unfortunately, this process applied in the decision tree learning to assess uncertainty and gain requires instance labels, and hence it cannot be replicated in our framework where instance labels are unknown.

Another issue to consider in the development of the region splitting criteria is that the information gain ignores the region size. Here the region size is defined as the empirical number of instances contained in a region. Intuitively, the largest benefit from the split

should be realized when not only the impure regions but also large regions are split. In light of this, we propose a new *uncertainty* score that takes into account both the size and the proportion label in deciding which region should be split next.

Suppose that at time t there are $N^{(t)}$ leaf regions $L^{(t)} = \{(R_i, \mu_i)\}_{i=1}^{N^{(t)}}$ where each region $R_i = \{\boldsymbol{x}_{ij}\}_{j=1}^{n_i}$ has n_i instances and has been assigned a label $\mu_i \in [0, 1]$ representing the *positive* class proportion, our goal is to choose the most uncertain region R_* to split. The uncertainty of each region R_i is defined as the expected number of wrong labels (denoted by w_i) if we randomly guess the class labels of all instances in R_i based on its proportion label μ_i. In particular, the procedure to calculate uncertainty is explained as follows:

1. For each instance in R_i, sample its label as an independent Bernoulli process with the parameter $= \mu_i$. 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 the parameter $= \mu_i$. 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. And use the expectation $\mathbb{E}(w_i) = 2\mu_i(1 - \mu_i)n_i$ as the uncertainty of R_i.

This uncertainty defined above clearly shows that larger n_i or more uncertain μ_i (closer to 0.5) leads to more uncertainty of region R_i. Please note here $2\mu_i(1 - \mu_i)$ matches exactly the definition of Gini-Index, so throughout our work we will choose Gini-Index as the gain measurement for later use. Finally, we select $R_* = \arg\max_{R_i \in L^{(t)}} \mathbb{E}(w_i)$ to be the most uncertain region to split at current active learning cycle t.

5.3 Two Heuristics For Splitting Regions

Now given the region R_*, we need to determine what input dimension to split and what value should be used to define the split. Since there are no labeled instances in our framework,

we develop two heuristics to drive the split.

5.3.1 Supervised Heuristic

Our first heuristic is *supervised* and it relies on the base classification model. In various active learning algorithms, the base model plays an important role in determining which data should be queried next. An example is the classic Uncertainty Sampling approach [Settles, 2012]. The base model reflects the current belief of the class distribution on instances and thus its guidance on the region splitting cannot be ignored. Formally, at learning time t, the base model is learned as $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$ so each instance $\boldsymbol{x}_{*j}$ will have a **S**upervised probabilistic label p_j^S reflecting the likelihood of belonging to one of the two classes. Here $p_j^S = P(y = 1|\boldsymbol{x}_{*j}; \hat{\boldsymbol{\theta}}^{(t)})$. Given these instance-level labels, a standard decision tree splitting procedure based on information gain can be now directly applied to split R_*. Here we use Gini-Index and say this procedure gives us the empirically optimal split of R_* from value v^S on dimension a^S based on the set of probabilistic unsupervised labels $\{p_j^S\}$.

5.3.2 Unsupervised Heuristic

The second heuristic is *unsupervised*. It is based on probabilistic clustering. Clustering is a simple yet often effective guidance. The assumption behind it is that similar data instances tend to carry similar class labels and it has been used frequently in semi-supervised learning [Zhu et al., 2003]. In other words, dissimilar data are likely to fall into different classes and thus the region splits should be driven by the underlying structure of data. To implement this idea, we perform a 2-means probabilistic clustering on the instances $\{\boldsymbol{x}_{*j}\}_{j=1}^{n_*}$ in R_*, assuming there is mix of two cluster centers in $\{\boldsymbol{x}_{*j}\}$ and the probabilities of cluster membership are given by Expectation and Maximization (EM) algorithm. Hence, each instance $\boldsymbol{x}_{*j}$ will have an **U**nsupervised probabilistic label p_j^U indicating the chance of belonging to one of the two clusters. Similarly, given these instance-level labels Gini-Index-based gain can again be applied to split R_* and say it gives the best split from value v^U on dimension a^U.

Table 5.1 summarizes the pros and cons of the two heuristics. Initially when the super-

Table 5.1: Comparison of the two heuristics.

	Supervised Heuristic	**Unsupervised Heuristic**
Pros	Gives instance-level estimates which directly reflect the class distribution	Relies on the semi-supervised assumption which is often effective
Cons	But initially these estimates are poor simply because the supervision is little	But this assumption may not hold all the time

vision is scarce, the base model trained can be very likely to make biased decisions. This problem was formally stated as *sampling bias* by Dasgupta *et. al.* [Dasgupta and Hsu, 2008] and they leverage hierarchical clustering to assist the base model. In our framework, we use clustering also as an unsupervised heuristic to alleviate the bias issue. However, the unsupervised heuristic may not always work well in the long run. Therefore, the best option appears to be the combination of the two heuristics.

5.4 Combination Of The Two Heuristics: A Competition Procedure

To combine the two heuristics, we first introduce a competition procedure as shown in Algorithm 5. The key idea is to perform a test split on each of the proposed splits separately and compare their actual gains. Larger gain is better and so the final split will take whatever the corresponding heuristic suggests. We also maintain a list H that records the winning history of the heuristics in the past splits and this H will be used to test whether the supervised heuristic is doing significant better than the unsupervised one in the long run. If the test result is significant, it marks that our base model is good enough to make splitting decisions alone; from then on, every region split will only be determined by the supervised heuristic. That is, the following procedure will *not* be called any more once we believe the supervised heuristic is performing significantly better and the final split will directly take the supervised proposal.

Algorithm 5 The competition procedure of choosing a heuristic

Input: Unsupervised split (a_U, v_U); Supervised split (a_S, v_S); Winning history H

Output: The final split (a_F, v_F); updated history H; Binomial test result of supervised heuristic

1: Binomial test result $r \leftarrow$ *Not significant*
2: **if** $a_U = a_S$ and $v_U = v_S$ **then**
3: $a_F \leftarrow a_S$; $v_F \leftarrow v_S$;
4: **else**
5: Do a test split on (a_U, v_U) and get its gain G_U;
6: Do a test split on (a_S, v_S) and get its gain G_S;
7: **if** $G_U > G_S$ **then**
8: Append *"Unsupervised heuristic wins"* to H;
9: $a_F \leftarrow a_U$; $v_F \leftarrow v_U$;
10: **else**
11: Append *"Supervised heuristic wins"* to H;
12: $a_F \leftarrow a_S$; $v_F \leftarrow v_S$;
13: Test result $r \leftarrow$ Binomial test (Algorithm 7) on H;
14: **end if**
15: **end if**
16: **return** (a_F, v_F), H and r

5.4.1 Test Split

The test split and the calculation of the gain procedure called in line 5 or 6 is identical to the evaluation of a standard decision tree splitting. Algorithm 6 shows how to calculate the gain G_S of the test split on R_* proposed by the supervised heuristic. The gain of G_U can be calculated similarly.

Algorithm 6 Evaluation of One Region Split

Input: A labeled region (R_*, μ_*); A splitting heuristic $a \in \{a^U, a^S\}$

Output: the information gain G_a after splitting R_*

1: Split R_* from value v on dimension d suggested by heuristic a into two sub-regions R^L and R^R;

2: Route each instance in R_* to R^L or R^R by testing the feature value of the instance on dimension d either $< v$ or $\geq v$;

3: Query the proportion label of either sub-region. Say R^L is annotated by human with a label μ^L;

4: Infer the label μ^R of R^R. This does not require a human assessment. Because of the proportion label constraint: $n^L\mu^L + n^R\mu^R = n_*\mu_*$ with $n^L + n^R = n_*$, where n^L, n^R and n_* are the number of instances contained in R^L, R^R and R_*, μ_R is calculated as: $(n_*\mu_* - n^L\mu^L)/n^R$;

5: Apply Gini-Index to calculate the information gain:

$$G_a = I(\mu_*) - \frac{n^L}{n_*} I(\mu^L) - \frac{n^R}{n_*} I(\mu^R)$$

where $I(\mu) = 2\mu(1-\mu)$.

6: **return** G_a

5.4.2 Binomial Test

Algorithm 7 provides the detail of the Binomial test that decides whether the supervised heuristic is doing significantly better than the unsupervised one. The null hypothesis H_0 means the supervised heuristic is doing equally well or worse than the unsupervised heuristic in the latest W trials. In other words, the winning chance of the supervised heuristic p_S is ≤ 0.5. Under H_0, the number of supervised wins B^* follows a Binomial distribution $Bin(W, 0.5)$ and we do a right-tailed test of B^* to carry out the p-value. We reject H_0 if the p-value is less than a given confidence level α and choose the alternative.

To make the test more conservative, multiple such tests with different window sizes can be done simultaneously. To ensure the same family wise error rate α, Bonferroni correction

Algorithm 7 Binomial test of the supervised heuristic

Input: Winning history of heuristics H; Window size W; Significance level α

Output: *Significant* or *Not significant*

1: **if** $length(H) < W$ **then**
2: **return** *Not significant*
3: **end if**
4: H_0: winning chance of supervised heuristic $p_S \leq 0.5$ in the last W trials in H;
5: H_A: $p_S > 0.5$
6: Test statistic $B^* \leftarrow$ number of supervised wins in the last W outcomes;
7: $p\_value \leftarrow$ do binomial test on B^*;
8: **return** *Significant* if $p\_value < \alpha$ else *Not significant*

can be applied. In our implementation, we combine a short term window $W_S = 5$ and a long term window $W_L = 10$ with the same family wise $\alpha = 0.05$. The purpose of conducting two tests together is to ensure that the supervised heuristic indeed has a stable performance.

5.5 Combination Of The Two Heuristics: An MAB-Based Procedure

5.5.1 Interpretation of HALR As An MAB Problem

In the previous section, we have seen the completion procedure that dynamically tests the two splitting heuristics and then chooses the better one to perform the final split. In other words, before the Binomial test gives *Significant* outcome each split would consume *two* queries. This immediately leads a question: can we improve the splitting efficiency such that every split only takes just *one* query? In this section, we give a positive answer by proposing a more advanced splitting procedure.

We notice that our "dynamic heuristic choosing" puzzle can be formulated as a Multi-Arm Bandit (MAB) problem. MAB is a rich and multi-disciplinary area studied extensively in Statistics, Economics, Operations Research, and Computer Science [Slivkins, 2018]. It is

used to model a plethora of *dynamic* optimization problems under uncertainty. The basic setting is simple: there is a fixed and finite set of actions, aka K arms and each arm can be pulled (chosen) with a unit cost but gives the puller a reward which is independently and randomly generated from a unknown reward distribution. Each arm may have a different reward distribution and the only way we know about the distribution is only through the sample rewards we have pulled. Given a fixed number of times T that one can pull, the goal is to find a good policy (i.e. at each time $t = 1, 2, ..., T$ decides which arm to pull) such that the total sum of rewards can be maximized. In this dynamic learning setting, there exists a tension between the acquisition cost of new information (*exploration*) and the generation of instantaneous rewards based on the existing information (*exploitation*). A simple algorithm called *uniform exploration* is to pull each arm several times and then always choose the arm with the maximum averaged rewards to pull for the rest times. This approach has a major flaw that the exploration schedule does not depend on the history of the observed rewards. It is usually better to *adapt* exploration to the observed rewards. Hence, a lot of good algorithms belonging to *adaptive* solution set have been developed [Slivkins, 2018].

Thus, if the framework HALR is reformulated as a MAB problem, it is not hard to see: (1) the total number of active learning cycles is equal to T and each cycle t corresponds to each time index; (2) there are $K = 2$ arms that are the two region splitting heuristics *unsupervised heuristic* and *supervised heuristic*; (3) the reward of choosing each heuristic is the information gain brought by that heuristic; (4) at each cycle t, we should develop a good procedure (policy) that chooses one heuristic to split the most uncertain region. In HALR, the policy is one type of uniform exploration. In this section, we implement a near-optimal adaptive policy proposed by [Besbes et al., 2014] and we name this advanced method as **A*HALR**.

5.5.2 An Adaptive Heuristic-Choosing Policy

To solve the heuristic-choosing problem, we propose an *adaptive* policy based on Multi-Arm Bandits (MAB). First let us reformulate our hierarchical active learning framework (Algorithm 4) as a MAB problem:

1. The total number of active learning cycles T is equal to the total number of pulls in MAB;
2. There are only two heuristics $\{a^U, a^S\}$ (aka $K = 2$ arms) to choose from, where a^U or a^S denotes the unsupervised heuristic or supervised heuristic.
3. In each cycle $t = 1, 2, ..., T$, a policy $\pi(t) : [T] \to \{a^U, a^S\}$ decides which heuristic to use;
4. The reward X_t^π, calculated by Algorithm 6, is the information gain after splitting R_* guided by heuristic $a = \pi(t)$. We assume X_t^π is a random variable drawn from a unknown distribution $P_{\phi^{(t)}}$, where $\phi^{(t)}$ is the mean reward at time t. This $\phi^{(t)}$ can vary for different heuristics, so it is further denoted by $\phi_a^{(t)}$, where $a \in \{a^U, a^S\}$. Another important fact is that the superscript t reflects that the mean rewards for both heuristics *can* change over time. So it is categorized as MAB with *non-stationary* or *stochastic* rewards [Besbes et al., 2014]. In this sense, $\phi_a^{(t)}$ is also a random variable.
5. In the paper of [Besbes et al., 2014], $\phi_a^{(t)}$ is assumed to be bounded by a *variation budget* $\mathcal{B} = \{V_t : t = 1, 2, ..., T\}$ which is a non-decreasing sequence of positive real numbers such that $V_1 = 0$, $KV_t \leq t$ for all t. Then the possible values of the mean reward sequence for both heuristics, denoted by $\phi = ((\phi_U^{(1)}, ..., \phi_U^{(T)})^\mathbf{T}, \phi_S^{(1)}, ..., \phi_S^{(T)})^\mathbf{T})^\mathbf{T}$, fall into the corresponding *temporal uncertainty set* $\mathcal{V}$:

$$\mathcal{V} = \{\phi \in [0, 1]^{K \times T} : \sum_{t=1}^{T-1} \sup_a |\phi_a^{(t+1)} - \phi_a^{(t)}| \leq V_T\}$$

   Remarks: the reasoning of using $\mathcal{B}$ and $\mathcal{V}$ is to describe the magnitude that how $\phi_a^{(t)}$ changes over time. V_T bounds the maximum sum of those changes over T. So V_T should in general be designed as a function of T. We will study and compare $\phi_a^{(t)}$ for both our heuristics later.
6. The quality of each policy π is measured by *regret* compared to a *dynamic oracle* as the worst-case difference between the expected performance of choosing at each cycle t the heuristic which has the highest expected reward $\phi_*^{(t)}$ at t (the dynamic oracle performance) and the expected performance under policy π. That is:

$$\mathcal{R}^\pi(\mathcal{V}, T) = \sup_{\phi \in \mathcal{V}} \{\sum_{t=1}^{T} \phi_*^{(t)} - \mathbb{E}^\pi[\sum_{t=1}^{T} \phi_\pi^{(t)}]\}$$

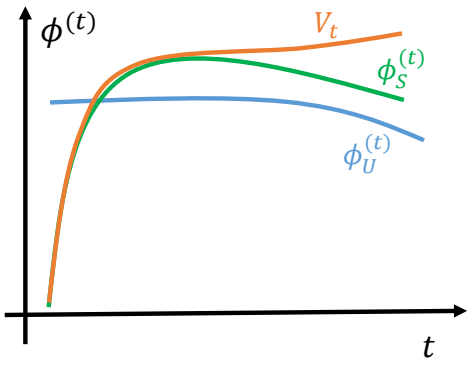

Figure 5.2: The variation of mean rewards over time.

7. Finally, given the prior knowledge of $\mathcal{B}$ and $\mathcal{V}$, our goal is to find a good policy π that minimizes its regret $\mathcal{R}^{\pi}(\mathcal{V}, T)$. In our problem, it means that we should construct a hierarchical tree of regions where the leaf regions can be refined as *pure* as possible after very *few* splits and queries made. That is, the maximum information gain is realized.

5.5.2.1 Mean Rewards Of The Two Heuristics As we have seen, to implement an adaptive policy requires the design of a variation budget $\mathcal{B}$ and a temporal uncertainty set $\mathcal{V}$. So a question is: how do the mean rewards $\phi_a^{(t)}$ of the both heuristics change over time? Recall in Section 5.3, we have the prior knowledge that at very beginning (when t is very small), unsupervised heuristic is very likely to perform better ($\phi_U^{(t)} > \phi_S^{(t)}$); however, when more regions are labeled (as t increases), supervised heuristic will catch up and may probably exceed unsupervised heuristic ($\phi_U^{(t)} < \phi_S^{(t)}$). So their changing trends can be roughly plotted in Figure 5.2. Several remarks:

1. The mean reward of unsupervised heuristic $\phi_U^{(t)}$ should be stable over time as it does not really vary regardless of how many labels are given.
2. The mean reward of supervised heuristic $\phi_S^{(t)}$ will increase when more labels are revealed but its changing trend (slope) may decrease as the base classifier starts to converge. Previous work [Luo and Hauskrecht, 2018b] has shown similar curves of the performance increase of the base models.

Algorithm 8 A Near-Optimal Policy π

Input: $\gamma \in [0, 1]$; A batch size ΔT.

for all batch of cycles with size $= \Delta T$ **do**
  Set $w_U^{(t)} = w_S^{(t)} = 1$ for both heuristics;
  for all cycle indexed at t in current batch **do**
    $p_U^{(t)} = (1-\gamma)\frac{w_U^{(t)}}{w_U^{(t)}+w_S^{(t)}} + \gamma/K$;
    $p_S^{(t)} = 1 - p_U^{(t)}$;
    Choose a heuristic a randomly according to the distribution $\{p_U^{(t)}, p_U^{(t)}\}$;
    Split R_* suggested by a;
    Receive a reward $X_a^{(t)}$ according to Algorithm 6;
    Boost $w_a^{(t+1)} \leftarrow w_a^{(t)} \exp\{\frac{\gamma X_a^{(t)}}{p_a^{(t)} K}\}$;
    Keep $w_{a'}^{(t+1)} = w_{a'}^{(t)}$ for $a' \neq a$;
  end for
end for

3. Both curves drop in the end as the leaf regions become purer and purer (so with less gain). Of course, the tails are of less interest.
4. So how to choose the variation budget function V_T mainly depends on the mean reward curve of supervised heuristic. According to the plot, the shape of $\phi_S^{(t)}$ can be bounded by a log function. Well, to be more conservative, we choose a looser bound of a square-root function: $V_T = O(T^{1/2})$.

5.5.2.2 A Near-Optimal Adaptive Policy We implement a near-optimal policy of choosing heuristics suggested by [Besbes et al., 2014] with $V_T = O(T^{1/2})$ assumed. The key to dealing with MAB with non-stationary rewards is to break the whole T cycles into batches of size $= \Delta T$ and then explore/exploit heuristics independently in each batch. Each heuristic is chosen with a mixed probability of Boltzmann distribution and uniform distribution, balanced by a constant γ. Within each batch, initially each heuristic is chosen with equal chance and then the probability of the chosen one will be boosted by the reward obtained. Algorithm 8 details this policy. Its near-optimality is proved by Theorem 2 in [Besbes et al., 2014]

when the batch size $\Delta T = \lceil (K \log K)^{1/3} (T/V_T)^{2/3}$ and with $\gamma = \min\{1, (\frac{K \log K}{(e-1)\Delta T})^{1/2}\}$. In our experiments, we set these two hyper-parameters accordingly.

5.6 Experiments

In this section, we test to what extent that our new framework HALR, A*HALR are able to improve upon HALG. We thereby use the same data sets and follow the same experiment settings that were used in HALG (Section § 4.5). We also include RIQY which was the second best method behind HALG.

5.6.1 Main Results

The main results are shown in Figure 5.3. Our graphs plot the AUC scores after each $t < T = 200$ region queries are posed which is large enough for all methods to converge. And to best visualize each plot, we omit the remaining tails of curves after most methods have converged. Overall, our new implementations HALR and A*HALR are able to outperform other methods on majority of the datasets and are close to HALG on the remaining sets (`Pima` and `Music`).

Compared to HALG, among 9 data sets, HALR wins on 7 data sets, and loses on 2 sets (`Pima` and `Music`). So in general, HALR shows a better performance than HALG. This is primarily attributed to HALR's new region-construction strategy rather than HALG's learning with groups. The active learning that HALR uses is capable of finding the most uncertain region and splitting it by explicitly using supervised heuristic and unsupervised heuristic. Consequently, it can further accelerate the model convergence rate.

Moreover, the advanced version A*HALR demonstrates an even better performance, A*HALR is able to outperform HALR on 7 datasets and ties on `Music` and `Messidor`. It is expected since A*HALR uses an adaptive Bandit algorithm which often works more efficiently than a uniform exploration approach that was used to HALR. Compared to HALG, sometimes we lose (on datasets `Pima` and `Music`). One reason could be that for particular

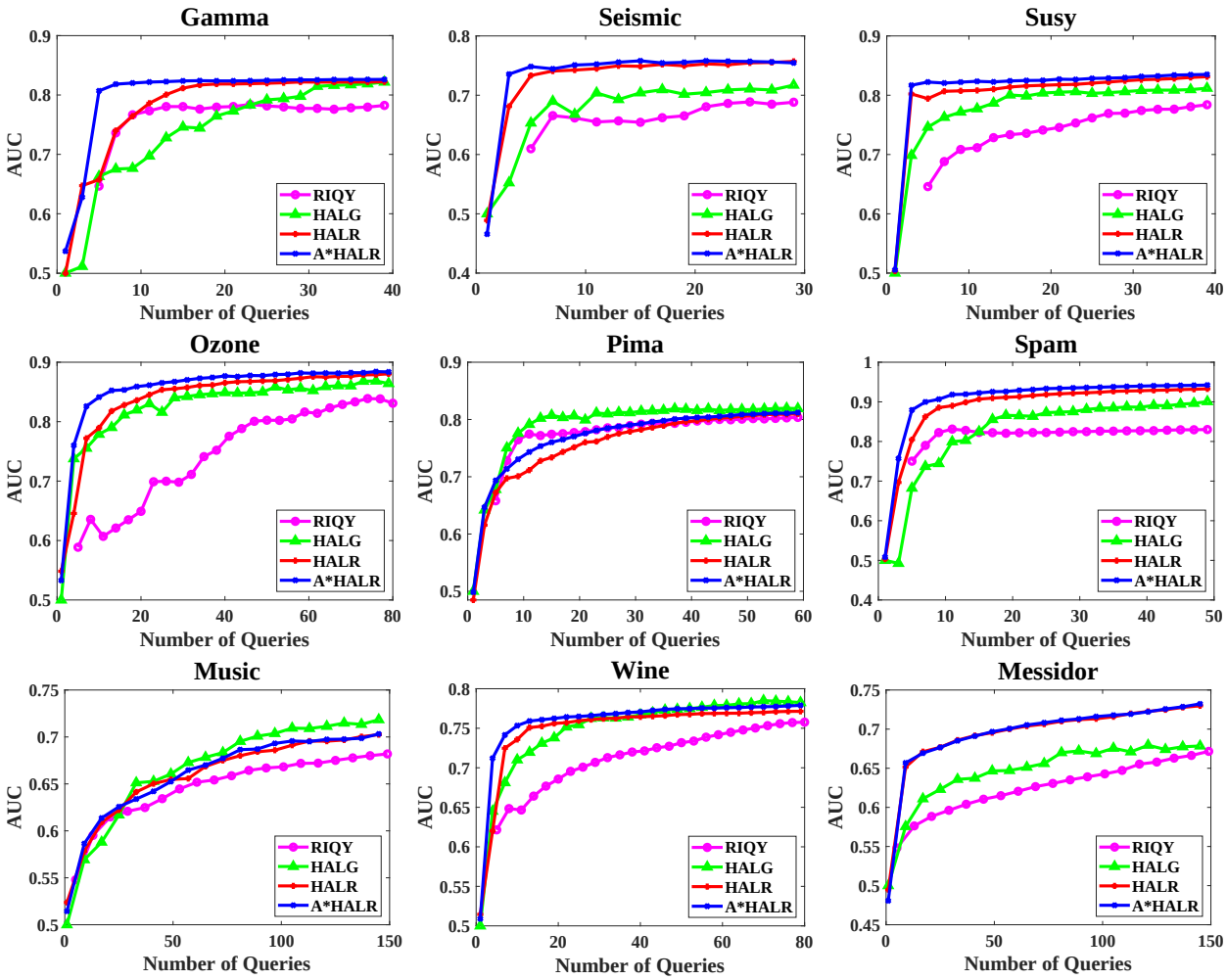

Figure 5.3: Performances of different methods on 9 UCI data sets.

datasets if clustering is highly associated with classification, then HALG (where clustering dominates) may be able to learn more efficiently. But in general, these results lend great credence to A*HALR's adaptive region division algorithm which in principle benefits from the near-optimal heuristic-choosing policy developed to solve Multi-Arm Bandit problems with non-stationary rewards.

5.6.2 Runtime Results

Besides, we also compare the actual runtime of HALG and HALR. As mentioned earlier, HALG is computational expensive because it needs to compute model change for every

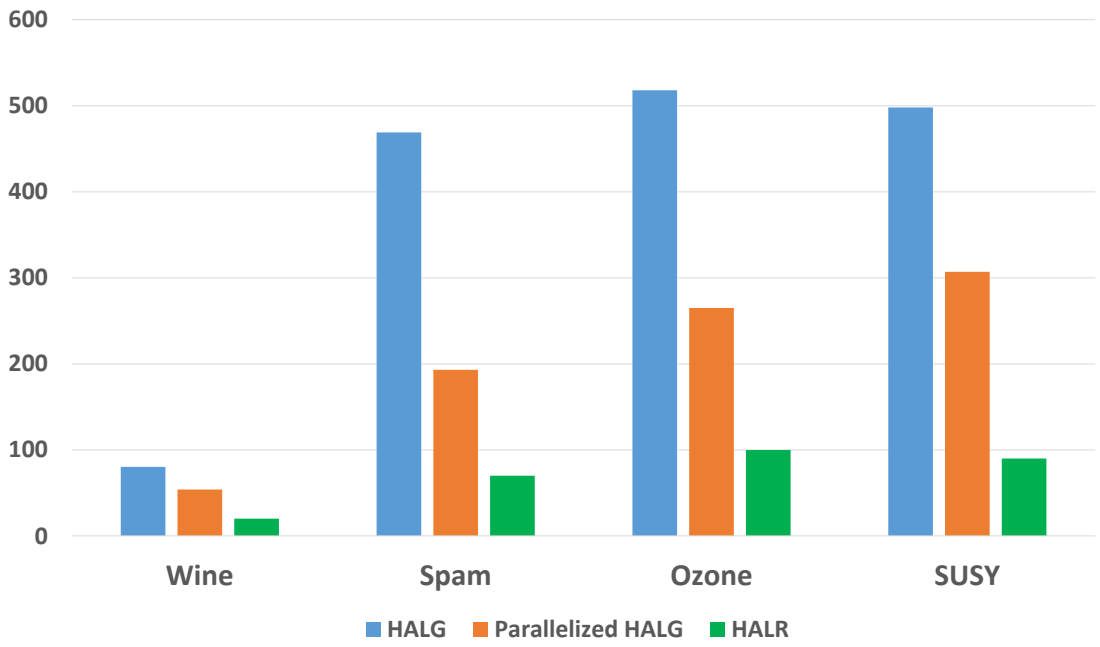

Figure 5.4: Total runtime (seconds) of different methods on the first 100 queries.

candidate group in every cycle. One remedy would be to parallelize the calculation of model change for different candidate groups as each group is independent to others.

To illustrate how much time it takes for all methods we show in Figure 5.4 the total runtime (in seconds) that is needed to pose the first 100 queries. The chosen 4 data sets are the most complex or the largest sets among all the data sets so they can best represent the runtime of different methods. Apparently according to the results, HALR is able save the runtime significantly compared to HALG or even parallelized HALG.

5.7 Chapter Summary

In this chapter, we presented HALR, a second framework that can actively learn instance-based classification models from region (group) proportion feedback. The high-level idea is to directly split the input space and dynamically form regions for querying, much like the

decision tree learning algorithm. But unlike decision tree learning, we do not have labeled instances to determine the splits. So we develop a region-based procedure for active region selection and split. This is done in two steps: (1) choose the most *uncertain* region, and (2) actively split it into two sub-regions. The split dimension and value are determined by a competition procedure (HALR) or an MAB-Based procedure (A*HALR) with unsupervised heuristic and supervised heuristic.

Compared to HALG, there are two essential benefits. First, while HALG identifies groups within a fixed group hierarchy, HALR allows more flexibility to explore meaningful groups. The positive outcome is that since each region selection and split is explicitly controlled by the base model (supervised heuristic), HALR is able to find regions that are more relevant to the class distribution. Hence, HALR is able to identify purer regions more quickly. Second, the active learning strategy *maximum model change* used in HALG requires much more computation power than the active procedure used in HALR. Indeed, according to our experiment results, HALR is able to learn more efficiently than HALG, no matter in terms of number of queries, or runtime.

Lastly, we would like to note that the two hypothesis **H1** and **H2** (Section § 1.4) are also supported by the results. The two representative instance-based active learning methods DWUS and HS have been shown inferior to HALG previously, and thereby perform less well than HALR; compared to the representative group-based active learning method RIQY, both HALG and HALR outperform it, and thus our hierarchical active learning framework has been demonstrated more query-efficient.

6.0 Hierarchical Active Learning With Overlapping Regions: HALOR

6.1 Introduction

6.1.1 Limitations Of Building Single Hierarchies

In our second completed work HALR, we have seen the potential of learning models from *regions* with proportion feedback. The regions identified for learning are formed hierarchically. To build such region hierarchy, we repeatedly split and query one *leaf* region (initially it is the region region $\mathcal{X}$), and then learn from the new set of leaf regions that are disjoint to each other. While this hierarchical solution works well, we do notice there is one major limitation in current solutions. The problem of HALR is that if some splits are uninformative (i.e. little information gain after the split), we cannot "revert" such splits. As those uninformative splits are already made, they will permanently affect the successive hierarchy building process. As we have seen in HALR, the initial several splits dominate the construction of the whole hierarchy. It is a dangerous situation because initially there is very little supervision given so the splits are often determined by clustering (unsupervised heuristic) that may be irrelevant to the underlying class distribution. Therefore, the initial splits overly dominate the hierarchy building process, and thus the regions formed may not be meaningful to humans or to model learning. Another drawback of HALR is that the leaf regions can be progressively more complex and specific as the hierarchy grows deeper; a negative effect of this is that the conjunctive patterns may become very difficult for humans to review and assess.

To further understand the limitation, let's consider one example. Suppose in a learning task we are interested in predicting when a particular disease will be present on a patient. The input space $\mathcal{X}$ consists of patient features. Assume the first split performed on the root region $\mathcal{X}$ is "$Age \leq 40$" or "$Age > 40$". Then we can imagine that all successive queries must include either "$Age \leq 40$" or "$Age > 40$" in their conjunctive patterns. In the worst situation, if the first split is uninformative, i.e. not mattering at all for physicians to diagnose

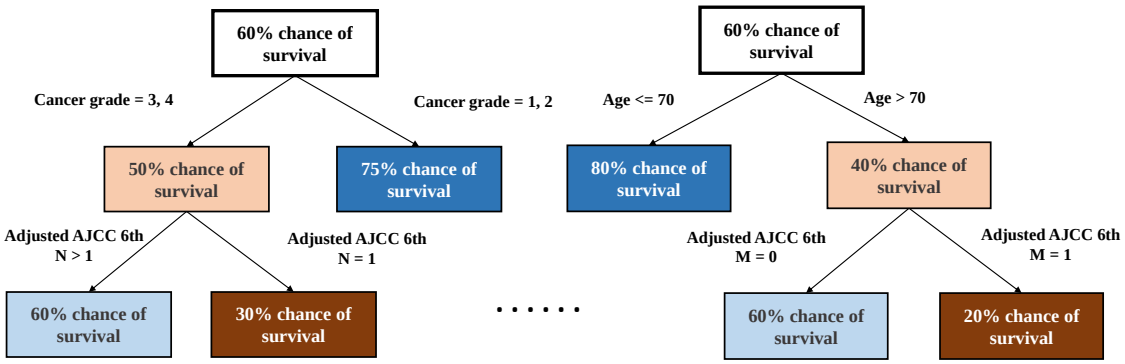

Figure 6.1: An illustration of the multiple-hierarchy solution HALOR applied to the survival analysis for colorectal cancer patients. The figure shows two trees that are constructed using different feature combinations.

the disease, then all the successive queries will be very awkward for human annotators. They may complain that *"I have said that patient's age greater or less 40 is not important; why do you keep sending me queries with this information?"*.

6.1.2 Framework Overview

In order to mitigate the above issues, we propose and develop a more robust approach **HALOR** that grows multiple HALR trees in parallel and permits learning with multiple **O**verlapping regions. Algorithm 9 gives the main steps. The solution is intuitive - if one hierarchical tree turns out to be uninformative or class-irrelevant, we can grow different trees to find more informative regions. For this purpose, we need to diversify the feature combinations in constructing different hierarchies. Figure 6.1 illustrates the idea. Hence, the multiple-hierarchy solution HALOR appears to be more robust. In addition to this, we note that by growing multiple trees we have multiple different regions to choose from, and in general, the complexity of all possible regions we can query and want to assess is not as high when compared to the approach with one tree hierarchy. As a result, the multiple-tree solution has a tendency to rely on less complex regions and thus makes it easier for humans to annotate such regions. Lastly, another difference of HALOR is on the learning algorithm. Since now regions created by HALOR can overlap to each other, the sampling

Algorithm 9 HALOR Framework

Input: Unlabeled data pool $\mathcal{U}$; Labeling budget $\mathcal{B}$; # of trees K

Output: A binary classification model $P(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})$

Initialize a root region $\mathcal{R}$
Query the class proportion μ of $\mathcal{R}$
Initialize K trees, each having one split on $\mathcal{R}$
Query or infer the class proportions of all the child regions
Maintain a fringe $F^{(K+1)}$ of all the leaf regions of the K trees
Active learning timer $t \leftarrow K+1$ (# of queries)
repeat
  Learn the base model $P(y|\boldsymbol{x};\hat{\boldsymbol{\theta}}^{(t)})$ from $F^{(t)}$
  Identify the most informative split (d_*, v_*, R_*), where R_* from $F^{(t)}$
  Split R_* from value v_* at dimension d_*
  Query or infer the class proportion of the new sub-regions
  $F^{(t+1)} \leftarrow \{F^{(t)} - (R_*, \mu_*)\} \cup \{R_*\text{'s labeled sub-regions}\}$
  $t \leftarrow t+1$
until the budget $\mathcal{B}$ runs out
return $P(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})$

based algorithm (Section § 4.3) used in HALG and HALR does not apply to HALOR. To this end, we will leverage a standard LLP learning algorithm introduced in Chapter § 3. In the following text, we provide the details of HALOR and its implementation.

6.2 Building Of One Hierarchy

We first talk about how to construct one HALOR hierarchy of regions. Building K different HALOR hierarchies can be further extended.

6.2.1 Preliminaries

Before we proceed, let us quickly review some notations used before and then introduce new ones. Our goal is to learn a binary classification model $f : \mathcal{X} \to \mathcal{Y} = \{0, 1\}$ which can make label inference for any instance $\boldsymbol{x} \in \mathcal{X}$. $\mathcal{X} \subset \mathbf{R}^m$ is the input data space and it consists of m features $\{d_1, ..., d_m\}$. The features can be numeric, nominal and categorical. Without loss of generality, we proceed with learning of a binary probabilistic model $P(y|\boldsymbol{x}; \boldsymbol{\theta})$, and we refer to it as our *base* model. To perform the actual LLP learning, we collect a pool of unlabeled data $\mathcal{U} = \{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathcal{X}, 1 \leq i \leq n\}$ as our training data, and we assume the data have been properly normalized and encoded according to their feature types.

A region R is a hypercubic subspace of the input space $\mathcal{X}$, defined as a pair $R = (C, D)$. C is the region description, realized by using conjunctive patterns that are joint value ranges over the input features:

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

Absence of any feature (or value) in conjunctive patterns indicates unbounded values on that feature dimension. $D \subset \mathcal{U}$ is the faction of empirical data of which the feature values are satisfied by the conjunctive patterns C. Each region R also comes with a label μ that is class proportion of the instance subpopulation represented by that region. By assuming that instances in $\mathcal{U}$ are i.i.d. sampled according to the underlying distribution of data $p(\boldsymbol{x})$, and that there are sufficiently many of them, we assume that the class proportion of the empirical instances D is close to μ and the difference is negligible.

Algorithmically, HALOR starts from an unbounded region $\mathcal{R}$ that covers the entire $\mathcal{X}$ space and this region serves as the root of the hierarchy (or tree). $\mathcal{R}$ also covers all data instance $\mathcal{U}$ we have collected. Thus $\mathcal{R}$ can be denoted by $\mathcal{R} = (\{\}, \mathcal{U})$ where the conjunctive pattern set is empty because of no value constraint defining it. The class proportion of $\mathcal{R}$ 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 step $t = 1, 2, ...$ (t = the number of queries consumed so far) there are $N^{(t)} = t$ labeled leaf regions in a fringe:

$F^{(t)} = \{(R_i, \mu_i) | 1 \leq i \leq N^{(t)}\}$. Our procedure aims to find the most influential split (d_*, v_*, R_*) which refers to the split of a region R_* in fringe $F^{(t)}$ using feature dimension d_* and its value v_*. The influence of a split is measured by how much class entropy it can reduce. 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 $F^{(t+1)} \leftarrow \{F^{(t)} - (R_*, \mu_*)\} \cup \{(R_*)'s \text{ sub-regions}\}$, and re-train the base model with the new $F^{(t+1)}$ by LLP algorithm.

6.2.2 A Comparison To HALR

Building one HALOR hierarchy is very similar to building HALR except for the following two differences (or improvements). First, we notice that the ultimate goal of HALR's active learning strategy is to find a very informative split that is made on some region. Yet HALR does it in two separate steps: (1) identifies the most uncertain region and then (2) seeks the most informative split only on that region. While it is sensible to split the most uncertain region, it may not produce the most information gain in the end, in that large uncertainty in parent region does not guarantee an informative split in the end. An extreme case is that the new sub-regions might be of the same uncertainty as their parent, and consequently the information gain would be zero. Having realized such a shortcoming, in HALOR, we combine the region selection and split procedures as *one step* and explicitly study the information gain globally with every possible split made on every candidate leaf region.

The second difference is that HALR treats the two splitting heuristics as two independent actions. That is, in each iteration they only choose one of the two heuristics to split the most uncertain region. They formulate the action-choosing puzzle as a Multi-Arm Bandit (MAB) problem and a MAB algorithm can strategically choose the better heuristic to perform the split. The key to a MAB framework is the trade-off between exploration and exploitation; HALR used a naive uniform exploration strategy which blindly tests each heuristic and choose the heuristic with more information gain; A*HALR improved the algorithm by using an adaptive exploration strategy that smartly selects a better heuristic based on the past

performance of the two heuristics. Either of the MAB algorithms, however, requires exploration steps that waste initial queries on exploring supervised heuristic that is yet known to be poor in the beginning. In HALOR, we combine the two heuristics into *one formula* and thus avoid the action-choosing puzzle. We explicitly balance the two heuristics by using a *supervised weight* which is determined by the current model performance. In this way, we not only simplify the heuristic choosing procedure but also associate the heuristic weights with the model performance.

6.2.3 Active Region Selection And Split

Building one HALOR hierarchy is similar to growing a decision tree. Recall that when we build a decision tree we iteratively split one leaf region using one the input dimensions and its value. By comparing all eligible splits for that region, the split that leads to the maximum information gain, or equivalently, maximum impurity reduction, is identified as the best region split. The impurity is measured by using labeled instances that fall in the region. In HALOR we still use *Gini-Index* as the impurity measurement, i.e. $I(\mu) = 2\mu(1-\mu)$. More formally, suppose one eligible 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:

$$G(d, v, (R_i, \mu_i)) = Gain(d, v, (R_i, \mu_i)) = I(\mu_i) - \frac{n^l}{n_i} I(\mu^l) - \frac{n^r}{n_i} I(\mu^r) \tag{6.1}$$

where $I(\mu)$ is the impurity measure for a region with a class proportion μ; μ_i, μ^l, μ^r are the class proportions calculated from data instances in regions R_i, R^l, R^r; n_i, n^l, n^r are the number of instances.

Back to our HALOR framework, unfortunately, such region splitting process cannot be replicated exactly as we do not know any instance labels. Therefore, in order to implement decision tree like splits in HALOR, we still leverage the two heuristics used in HALR (Section § 5.3).

6.2.3.1 Supervised Heuristic Suppose the base model is learned as $P(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}}^{(t)})$ from $F^{(t)}$, then the class proportion of any sub-region R^s ($s = l$ or r) can be inferred as:

$$\hat{\mu}^s = \frac{1}{n^s} \sum_{j=1}^{n^s} P(y_j = 1|\boldsymbol{x}_j; \hat{\boldsymbol{\theta}}^{(t)}) \tag{6.2}$$

With such approximation we can calculate the supervised information gain of a split (d, v) made on region R_i:

$$G_s(d, v, (R_i, \mu_i)) = I(\mu_i) - \frac{n^l}{n_i} I(\hat{\mu}^l) - \frac{n^r}{n_i} I(\hat{\mu}^r) \tag{6.3}$$

where μ_i is the true proportion label of R_i, and $\hat{\mu}^l$ and $\hat{\mu}^r$ are the estimated class proportions of sub-regions R^l and R^r.

6.2.3.2 Unsupervised Heuristic We implement the unsupervised heuristic based on 2-means clustering. Consider splitting a region R_i in the current fringe of leaf regions $F^{(t)}$. We perform a 2-means probabilistic clustering on the instances $D_i = \{\boldsymbol{x}_{ij}\}_{j=1}^{n_i}$ in R_i. After clustering, each instance $\boldsymbol{x}_{ij}$ will have an unsupervised probabilistic label p_{ij}^u indicating the chance of belonging to one of the two clusters. Given these instance-level labels, we use the following equation to measure the information gain (w.r.t. separating clusters) of a split (d, v) that is made on region R_i along feature dimension d at value v:

$$G_u(d, v, R_i) = I(\tilde{\mu}_i) - \frac{n^l}{n_i} I(\tilde{\mu}^l) - \frac{n^r}{n_i} I(\tilde{\mu}^r) \tag{6.4}$$

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 R^l, R^r.

6.2.3.3 Combination Of The Two Heuristics Different from HALR which treats the two heuristics as two different actions, in HALOR we combine their gains as one formula. That is, we define the final gain of splitting region R_i at (d, v) as a weighted sum of the supervised gain (G_s) and the unsupervised gain (G_u) defined above:

$$G_{s+u}(d, v, (R_i, \mu_i)) = \alpha_s^{(t)} \cdot G_s(d, v, (R_i, \mu_i)) + (1 - \alpha_s^{(t)}) \cdot G_u(d, v, R_i) \tag{6.5}$$

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 as the model becomes more accurate. To realize this intuition, we set $\alpha_s^{(t)} \propto [1/\epsilon^{(t)}]^2$ where $\epsilon^{(t)}$ is the training error of fitting the model to the labeled regions in $F^{(t)}\{(R_i, \mu_i)|1 \leq i \leq N^{(t)}\}$:

$$\epsilon^{(t)} = \epsilon(F^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}) = \sum_{i=1}^{N^{(t)}} \frac{n_i}{n} |\hat{\mu}_i - \mu_i| \tag{6.6}$$

where

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} P(y_j = 1|\boldsymbol{x}_j; \hat{\boldsymbol{\theta}}^{(t)}) \tag{6.7}$$

6.2.3.4 Selection And Split Of The Most Influential Region Finally, with Equation (6.5) defined above, we can iterate each possible split that can be made on each region (R_i, μ_i) in the fringe $F^{(t)}$, and identify the most informative split (d_*, v_*) made on a region R_* that would lead to the maximum reduction in the impurity measure. That is:

$$(d_*, v_*, R_*) = \arg \max_{(d,v)\ splits\ (R_i,\mu_i) \in F^{(t)}} \left[\frac{n_i}{n} \cdot G_{s+u}(d, v, (R_i, \mu_i)) \right] \tag{6.8}$$

6.3 Building Of Multiple Hierarchies

6.3.1 Initialization Of Multiple Hierarchies

Building multiple region hierarchies extends from the process of building one HALOR hierarchy. We initialize an unbounded root region $\mathcal{R}$ that is equivalent to the entire input space $\mathcal{X}$. It covers all data instances $\mathcal{U}$. The root region is queried with a class proportion μ. After that, we split the root region K times to create K different trees where each tree has only one split and the split dimension is different from the other $K-1$ trees. As initially we do not have enough supervision, we do the K splits using *unsupervised* heuristic. More specifically, we first perform a 2-means clustering on the instances $\mathcal{U}$ and then choose the top-K features (associated with their best split values) that can best separate the two clusters. Again, the best split is the one that gives rise to the maximum unsupervised information gain (see the notation used in Equation (6.4)):

$$G_u(d, v, R) = I(\tilde{\mu}) - \frac{n^l}{n} I(\tilde{\mu}^l) - \frac{n^r}{n} I(\tilde{\mu}^r) \tag{6.9}$$

Unsupervised splits, however, may find features that are not very relevant to the classification task we want to solve. This bias issue, in general, is unavoidable in weakly supervised learning, but can be mitigated by multiple different trees. We will see in the experiments that even a small $K \approx 2, 3$ can effectively alleviate such issue. After each of the K splits, we query one sub-region and then infer the label of the the other. In the end, the initialization phase outputs K one-split trees, together $2K$ labeled regions in total. We put them into the fringe $F^{(K+1)}$.

6.3.2 Active Region Selection And Split

The active region selection and split procedure remains the same as in Section 6.2.3 (*except* one modification that will be discussed later). That is, given the fringe $F^{(t)}$ of labeled leaf regions at time t, we split the most influential region that leads to the maximum combined information gain:

$$(d_*, v_*, R_*) = \arg \max_{(d,v) \ splits \ (R_i,\mu_i) \in F^{(t)}} \left[\frac{n_i}{n} \cdot G_{s+u}(d, v, (R_i, \mu_i)) \right] \tag{6.10}$$

where $G_{s+u}(d, v, (R_i, \mu_i))$ is the combination of the unsupervised and supervised gains:

$$G_{s+u}(d, v, (R_i, \mu_i)) = \alpha_s^{(t)} \cdot G_s(d, v, (R_i, \mu_i)) + (1 - \alpha_s^{(t)}) \cdot G_u(d, v, R_i) \tag{6.11}$$

6.3.2.1 Removal Of Region Duplicates One key point of HALOR is to diversify regions it generates. However, it is still possible that the regions generated from two different trees become identical or very close to each other. To prevent this we define a *region removal procedure* that executes before Formula (6.10) and removes the region duplicates from consideration. We note that this procedure is also applied for the initial K splits that seed K different hierarchies (trees). We define the region duplicates as follows:

Definition 4. *Given two regions $R_1 = (C_1, D_1)$ and $R_2 = (C_2, D_2)$. Let S_1, S_2 denote a set of unique features that are used in conjunctive patterns C_1, C_2. Let $D = D_1 \cap D_2$ denotes the intersection of data instances $D1$, $D2$. If (1) $S_1 \equiv S_2$ 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 duplicates.*

The first condition requires that the duplicate regions must be described by the same set of features. Even though a region is a super set of another, if they are described by different features they are not considered duplicate in that the smaller region is described more specifically and can acquire more accurate label information. The second condition ensures that two duplicate regions have sufficient overlapping with each other. In our experiments we set $\gamma = 0.1$ to perform a strict deduplication. With the definition above, when we traverse all eligible splits for each of the leaf regions in Formula 6.10, we first apply a deduplication procedure that disregards any invalid split (d, v) if it generates at least one sub-region that is duplicate to any other regions in $F^{(t)}$. Formally, consider a split (d, v) that splits a region $R_i \in F^{(t)}$ into two sub-regions R^l and R^r. If R^l or R^r is duplicate to any region $R' \in F^{(t)} - \{R_i\}$, then the split (d, v, R_i) is invalid and will be disregarded. An efficient implementation of this procedure is to first select out valid features as well as valid values and then only compute gains for these valid splits.

6.4 Learning From Overlapping Regions

After acquiring labels for the new sub-regions, the new fringe regions can be calculated as:

$$F^{(t+1)} = \{F^{(t)} - (R_*, \mu_*)\} \cup \{(R^l, \mu^l), (R^r, \mu^r)\} \tag{6.12}$$

To learn a classification model from the labeled regions (some of them overlap), we can then adopt the general LLP learning algorithm presented in Section 3.3. We design a loss function in spirit of the weighted lease squares:

$$\mathcal{L}(F^{(t)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{n_i}{n} (\hat{\mu}_i - \mu_i)^2 + \lambda \mathcal{C}(\boldsymbol{\theta}) \tag{6.13}$$

where $\hat{\mu}_i$ is the estimated class proportion by the base model (Equation 6.7) and $\lambda \mathcal{C}(\boldsymbol{\theta})$ is the regularization term penalizing model complexity. In our experiments we use L_2 penalty: $\mathcal{C}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$.

Finding of the best parameter $\hat{\boldsymbol{\theta}}$ can be worked out by a standard optimization program [Givens and Hoeting, 2012]. We use a gradient-based optimization approach. The loss function defined above is trained by minimizing squared loss via L-BFGS that was introduced in [Nocedal and Wright, 2006]. The gradient is computer as:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L} = \sum_{i=1}^{N} \frac{n_i}{n} (\hat{\mu}_i - \mu_i) \nabla_{\boldsymbol{\theta}} \hat{\mu}_i + \lambda \nabla_{\boldsymbol{\theta}} \mathcal{C} \tag{6.14}$$

where $\nabla_{\boldsymbol{\theta}} \hat{\mu}_i$ is subject to specific model choice. For larger data sets or more complex base models, stochastic gradient descent (SGD) can be applied by decomposing the summation in Eq. (6.14) and computing just one region's gradient $\frac{n_i}{n} (\hat{\mu}_i - \mu_i) \nabla_{\boldsymbol{\theta}} \hat{\mu}_i$ to perform gradient update.

6.5 Methodology Summary

We have presented HALOR, a third framework that actively builds one or more region hierarchies that are annotated by humans. The framework defines regions in a top-down (general-to-specific) manner. The selection of the regions is driven by the maximum impurity reduction principle in order to identify pure leaf regions as quickly as possible. This is because the purity of regions is the key to LLP learning success. To implement it, we leverage clustering, an unsupervised but usually class-relevant heuristic, and incorporate it into our active learning strategy.

The motivation of proposing HALOR is to remedy (at least partly) the limitations of HALR. A hierarchy built by HALR may be overly dominated by unsupervised heuristic that can be irrelevant to the actual class distribution. We thereby propose HALOR that grows multiple trees in parallel and helps one to recover from bad region selection.

6.6 Experiments

As the last set of experiments, we conduct a very comprehensive study to empirically evaluate all of our approaches (HALG, HALR, HALOR) on 18 data sets that are collected from `OpenML` machine learning repository [Vanschoren et al., 2013]. The purpose is to see how efficiently (in terms of the number of queries) our hierarchical active learning approaches can learn classification models when labelling of examples is associated with a human annotation cost.

6.6.1 Data Sets

In general, data sets used in empirical studies should be *unbiased* (i.e. not in favor of any method), *general* and *influential*. In order to do so, we did the following steps to collect the 18 data sets:

1. Go to "Data Sets" page on `OpenML`;

2. Click "Filters" to select data sets that are "Active, Binary Classification, and No Miss Values";
3. Sort the filtered data sets by "Highest Impact";
4. Skip data sets that have too few (< 10) or too many (> 500) features;
5. Finally, select the top 18 data sets[1].

These data sets come from a variety of real life applications. Table 6.1 summarizes their basic information and statistics. Some data sets have high-dimensional feature space:{`musk`, `ozone`, `nomao`, `hill-valley`, `scene`}; some carry highly unbalanced class distribution: {`satellite`, `ozone`, `pc1`, `climate`}.

6.6.2 Baselines

We compare HALG, HALR (actually using A*HALR) and HALOR-K (K trees) to three other representative active learning approaches:

- **HS**[2]: **H**ierarchical **S**ampling for active instance sampling [Dasgupta and Hsu, 2008]. It leverages a pre-compiled hierarchical clustering to drive the instance-selection procedure. This strategy intends to query instances from impure clusters. It also splits clusters when it believes some of the child clusters are pure enough. In terms of model learning, not only the labeled instances but also the instances with predicted labels within those sufficiently pure clusters are used for learning. The reason for doing so is because they assume that the cluster structure is well aligned with the actual class labels.
- **DWUS**[3]: **D**ensity-**W**eighted **U**ncertainty **S**ampling [Settles, 2012]. It is an instance-based active learning approach that queries instances not only uncertain but also representative of other data. It is related to our methods as it also considers both unsupervised and supervised heuristics.
- **RIQY**[3]: the first active learning work that deals with forming and querying regions [Rashidi and Cook, 2011]. Its region-construction strategy, well, is driven by an instance-

[1]By the time of October 2019.
[2]We use the original code at http://www.cs.columbia.edu/~djhsu
[3]Because the original code is unavailable, we have implemented this approach tightly according to its original algorithm description and also fine-tuned the hyper-parameters.

Table 6.1: 18 binary classifications data sets.

Dataset	**Data#**	**Feature#**	**Class%**	**Feature Type**	**About**
musk	6598	**167**	84.59%	Num, Ord, Cat	Molecules
satellite	5100	36	**98.53**%	Num	Satellite images
bank	45211	16	88.30%	Num, Ord, Cat	Financial accounts
ozone	2534	**72**	**93.69**%	Num	Ozone levels
kc1	2109	21	84.54%	Num	NASA software
pc1	1109	21	**93.06**%	Num	NASA software
wdbc	569	30	62.74%	Num	Breast cancer images
eye	14980	14	55.12%	Num	Eye states
ilpd	583	10	71.35%	Num, Cat	Indian liver patients
biodeg	1055	41	66.26%	Num	Chemicals
phishing	11055	30	55.69%	Ord, Cat	Phishing websites
nomao	34465	**118**	71.44%	Num, Ord, Cat	Places deduplication
climate	540	20	**91.48**%	Num	Climate model
hill-valley	1212	**100**	50.00%	Num	Hill valley recognition
click	39948	9	83.16%	Num	Web ad clicks
telescope	19020	10	64.84%	Num	Gamma telescope
scene	2407	**299**	82.09%	Num, Cat	Scene recognition
steel-plates	1941	33	65.33%	Num	Steel-plate faults

selection procedure. That is, it first selects an instance $\boldsymbol{x}^*$ determined by DWUS algorithm and then forms a compact small region centered on $\boldsymbol{x}^*$. It relies on DWUS because they want $\boldsymbol{x}^*$ to be not only uncertain but also representative of other data in the region. In such a way, the region would likely to be pure, and thus they can safely assign the major class label of the region to all the instances within. On the other hand, however, if $\boldsymbol{x}^*$ poorly represents the data in the region or the region turns out to be very impure, RIQY would only assign the major class label to $\boldsymbol{x}^*$ alone; such kind of region query, we

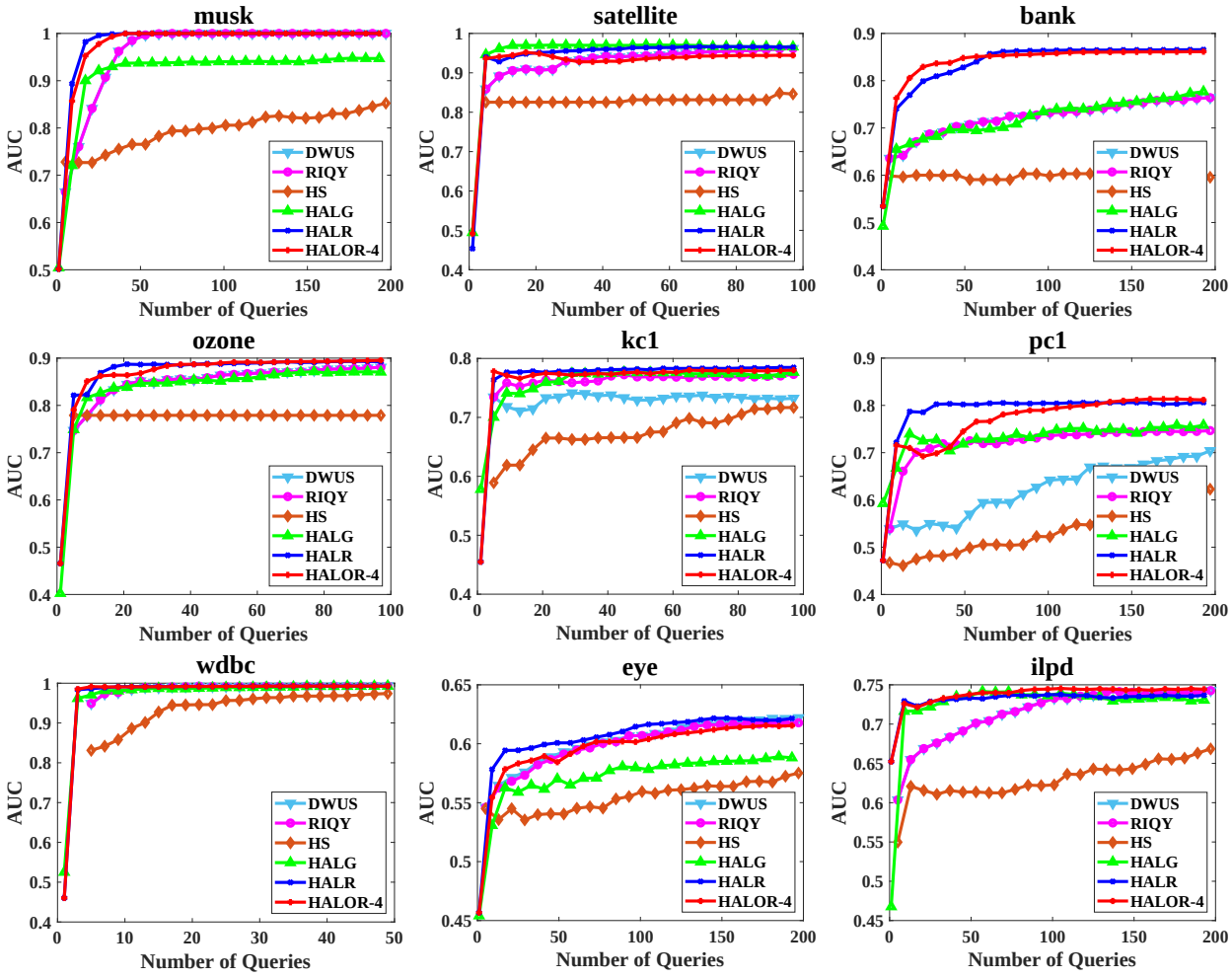

Figure 6.2: Model performance on the first 9 data sets.

say, *degenerates* to an instance-based query of $\boldsymbol{x}^*$.

6.6.3 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 RIQY and DWUS require a small portion of labeled data to start training, while others do not. To simulate region proportion feedback from human annotators, we count the empirical instances labels that fall into some region

and report the class fraction as the class proportion. Such a simulation method was originally used to test RIQY and has also been used in previous LLP studies [Quadrianto et al., 2009, Rueping, 2010, Patrini et al., 2014, Yu et al., 2013].

We evaluate all methods using two metrics. The first metric is the model performance against the number of queries. We adopt the Area Under the Receiver Operating Characteristic curve (AUC) to evaluate the generalized classification quality of Logistic Regression on the test data. Models trained by all methods use the same regularization parameter λ which has been chosen mildly between $[10^{-5}, 10^{-3}]$. The second metric is the query complexity against the number of queries. This metric reflects how complex each query could be (i.e. how many features have been used in a query).

In the following sub-sections, we will show plots on how AUC score and query complexity would change after each t queries ($t < 200$) have been posed. 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.

6.6.4 Results: Model Performance Vs. Number Of Queries

The main results are shown in Figure 6.2 and 6.3. Table 6.2 summarizes the ranking of all methods. Overall, HALR and HALOR are leading the best performance; RIQY, DWUS and HALG follow; HS ranks at the last place. There are a couple of general observations we can make:

- First, comparing region-based active learning methods (HALs, RIQY) to instance-based ones (DWUS, HS), we can see 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, a few labeled instances are usually insufficient to determine a good decision boundary. In contrast to this, generic region queries cover more instance sub-populations as well as larger input sub-space. Consequently, one region query can provide much wider class information than one instance query to drive model learning. Later on, as more queries are queried, regions are refined to be purer and they in turn accelerate

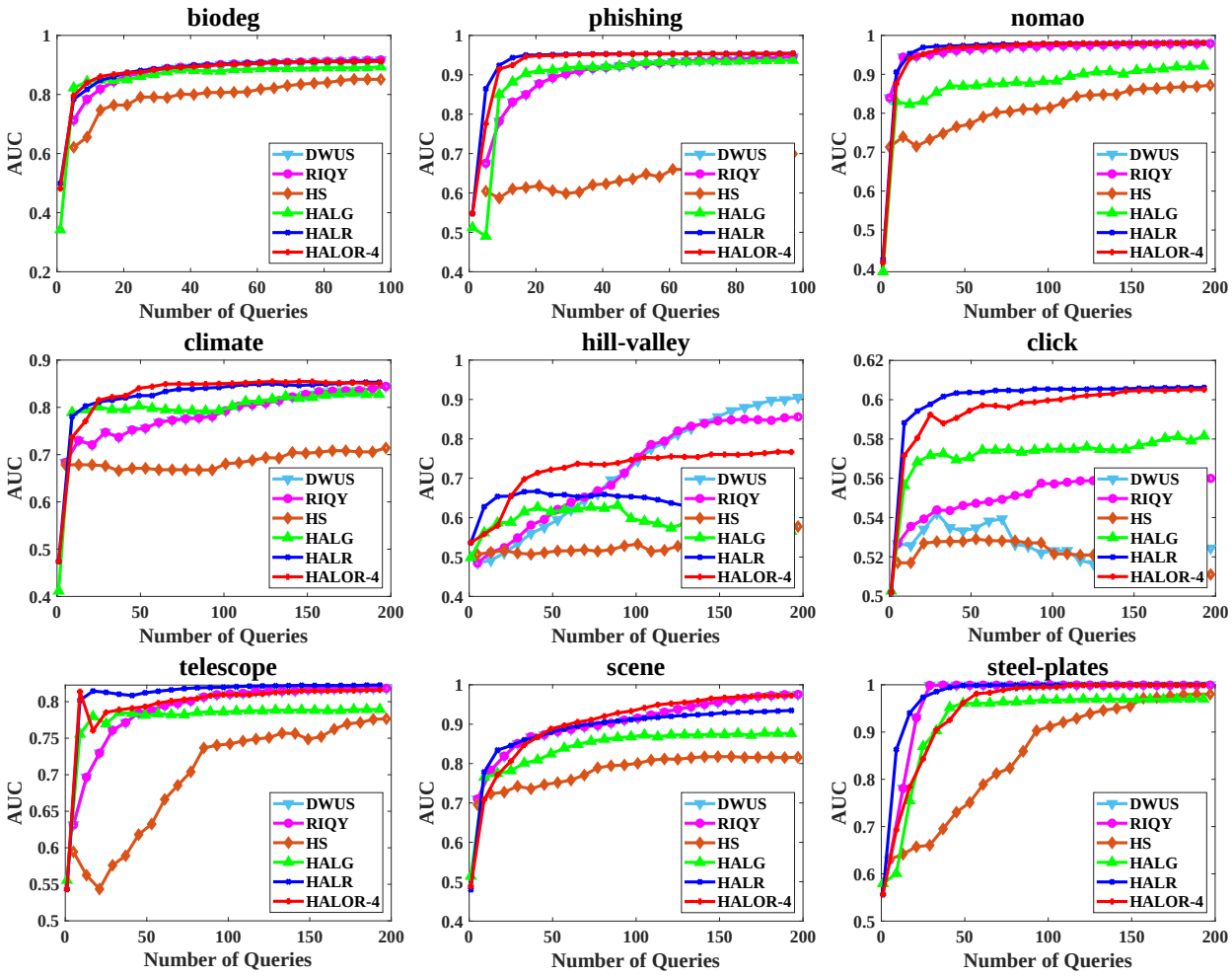

Figure 6.3: Model performance on the second 9 data sets.

the model convergence. Such observations demonstrate the advantages of active querying and learning with regions.

- Second, comparing our hierarchical active learning framework (HALs) with RIQY, we see that HAL frameworks, especially HALR and HALOR, are able to find better regions for labeling and learning. Briefly, RIQY is a very conservative method for finding regions, as it is in fact an instance-driven approach. In contrast to it, HALs construct regions in a systematic way and also learn models via LLP algorithms. The results show that HALs can learn better instance-based models from actively labeled regions.
- Finally, among the three HAL implementations, HALR shows the best performance.

In general, dynamically building a decision tree like hierarchy (HALR and HALOR) is able to identify more meaningful regions than pre-computing a region hierarchy which is done in unsupervised manner by a hierarchical clustering approach in HALG. Comparing HALR and HALOR, we see that HALR performs slightly better. The likely reason for this is that clustering heuristics was likely able to identify and match closely the class structure in the dataset by following the greedy selection on just one tree. However, we also notice that HALR fails on data set `hill-valley`. This is the only problem out of 18 problems where clustering is irreverent to classification; consequently, the single-tree implementation HALR is not able to identify well pure regions and thus the learning process turns out to be slow. Please note that multiple-tree approach (HALOR) has mitigated this problem by growing multiple trees. However, we note the initial performance of HALOR slightly drops since it needs more queries initially to grow multiple trees.

Now let us analyze each of the methods (and its performance) in detail.

6.6.4.1 HS As discussed, HS is an instance-based active learning method and its selection strategy is driven by the underlying hierarchical structure of data. Its actual performance shown here, unfortunately, is not as good as one would hope for. There are two possible reasons for it. First, as we pointed out for HALG, hierarchical clustering is an unsupervised heuristic which may not form good clusters at the very beginning. Hence, HS may only be applied to applications where *the cluster structure is well aligned with the actual class distribution.* Based on our results, such an assumption may not hold in general as almost all results show it to be inferior to other methods. The second limitation of HS is that it trains models also with *un-queried* examples that are yet to be labeled with the majority label in those *sufficiently pure* clusters. However, such predicted labels are risky and they can mislead model learning, especially when the underlying class distribution is severely unbalanced. For example, in highly unbalanced data, HS may take numerous queries before the first instance from the rare class is sampled, and thus the whole active learning efficiency may not be very good. Data sets `satellie`, `ozone`, `pc1` and `climate` carry highly unbalanced class distributions and they show an even slower learning rate of HS. Finally, we

Table 6.2: 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
scene	4	1	1	3	2	1
steel-plates	3	1	1	2	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**

note that HS is unable to reach very good AUC score on these data sets for the number of queries considered in the experiments.

6.6.4.2 DWUS Density-weighted uncertainty sampling is a good representative of the classic instance-based active learning approach. Besides using the basic uncertainty mea-

surement of data, it also considers the underlying structure of data in order to avoid sample outliers. Overall, DWUS appears to work very well on the majority of the data sets[4] except `pc1` and `click`. But surprisingly, DWUS shows its best model convergence on `hill-valley`. It is because DWUS is the only method that directly uses queried labels to train models. That is, it does not use predicted labels on unlabeled data for model learning (while HS and RIQY do), nor does it use LLP algorithm (like HALs) which may not recover a good instance-based model from not-pure-enough regions.

6.6.4.3 RIQY RIQY is one of first active learning frameworks that works with generic region-based queries. But based on our results it is also very conservative. It is conservative because it tends to assign the majority class label obtained for a region to all instances within it. It is also the reason why RIQY tries to form a relatively small compact region around $\boldsymbol{x}^*$. However, if the region defined around $\boldsymbol{x}^*$ does not cover other instances the region query would degenerate to the query of $\boldsymbol{x}^*$ only. Moreover, if the region queried is impure, $\boldsymbol{x}^*$ may be assigned a wrong label. Unfortunately, according to our results, most of RIQY's region queries cover single instances. This is why we usually see RIQY and DWUS overlap with each other on the plots. Nevertheless, when region queries manage to cover more instances, we can see that RIQY outperforms DWUS. This is visible for `kc1`, `pc1` and `click` datasets.

6.6.4.4 HALG HALG is the first implementation of our hierarchical region-based active learning framework. Overall, it shows reasonably good and stable performance. Compared to RIQY or DWUS, it has 6 wins, 4 draws and 8 losses. However, there are two major drawbacks of HALG. First, hierarchical clustering can slow down the process of finding pure regions. We can see this fact on data sets `bank`, `hill-valley`, `click`, `scene` and `steel-plates`, where HALG has a slower learning rate than region-based HALR and HALOR methods. The second limitation which appears to be more serious is that HALG is unable to converge to a very good model on many problems, such as, `musk`, `bank`, `pc1`, `eye`, `nomao`, `click`, `telescope` and `scene`. The reason is the gap between the actual group formation and the corresponding region that is used for querying. Recall that in HALG, each group is described

[4]On most of the plots DWUS overlaps with RIQY so it may be hard to see the light blue curve of DWUS.

by an *approximate* region which, however, may represent a different set of instances. As a result, the returned label proportion of the region does not exactly reflect the proportion of the group. Such a gap turns out to be harmful to model learning.

6.6.4.5 HALR HALR is the second implementation of the HAL framework that directly queries and learns with regions. Its overall performance has shown to be the best among all methods on all data sets except `hill-valley`. This evidence shows that HALR works very well for general classification tasks. Its advantage is that it does not assume a strong correlation between the structure of data and the classification problem we want to solve (unlike HS which only works when clustering is closely aligned with the actual labels). Under such a condition, the two splitting heuristics used in HALR - unsupervised and supervised heuristics - can work together to build a concise hierarchy of regions that can quickly find out pure leaf regions. On the other hand, however, we see it fail on `hill-valley` data. If clustering and classification are close to independent, HALR would become inefficient in finding pure regions.

6.6.4.6 HALOR The third hierarchical active learning approach we have developed is HALOR. It grows multiple HALR trees together and each tree is built from different feature combinations. In the experiments we grow 4 trees for HALOR, hence we refer to it as to HALOR-4. More analysis on the number of trees will follow. As expected, on `hill-valley` HALOR dramatically outperforms HALR and shows more solid performance. The reason why HALOR improves over HALR is that it can explore more class-relevant feature combinations using different trees and then automatically grow the trees that have high potential for reducing class entropy. On the other hand, since initially more queries are spent on building features for multiple trees, its initial performance may drops compared to HALR. Data sets reflecting this are `pc1`, `hill-valley`, `click`, `telescope` and `steel-plates`. However, we can also see that after this initial phase HALOR can rapidly improve and even outperform HALR.

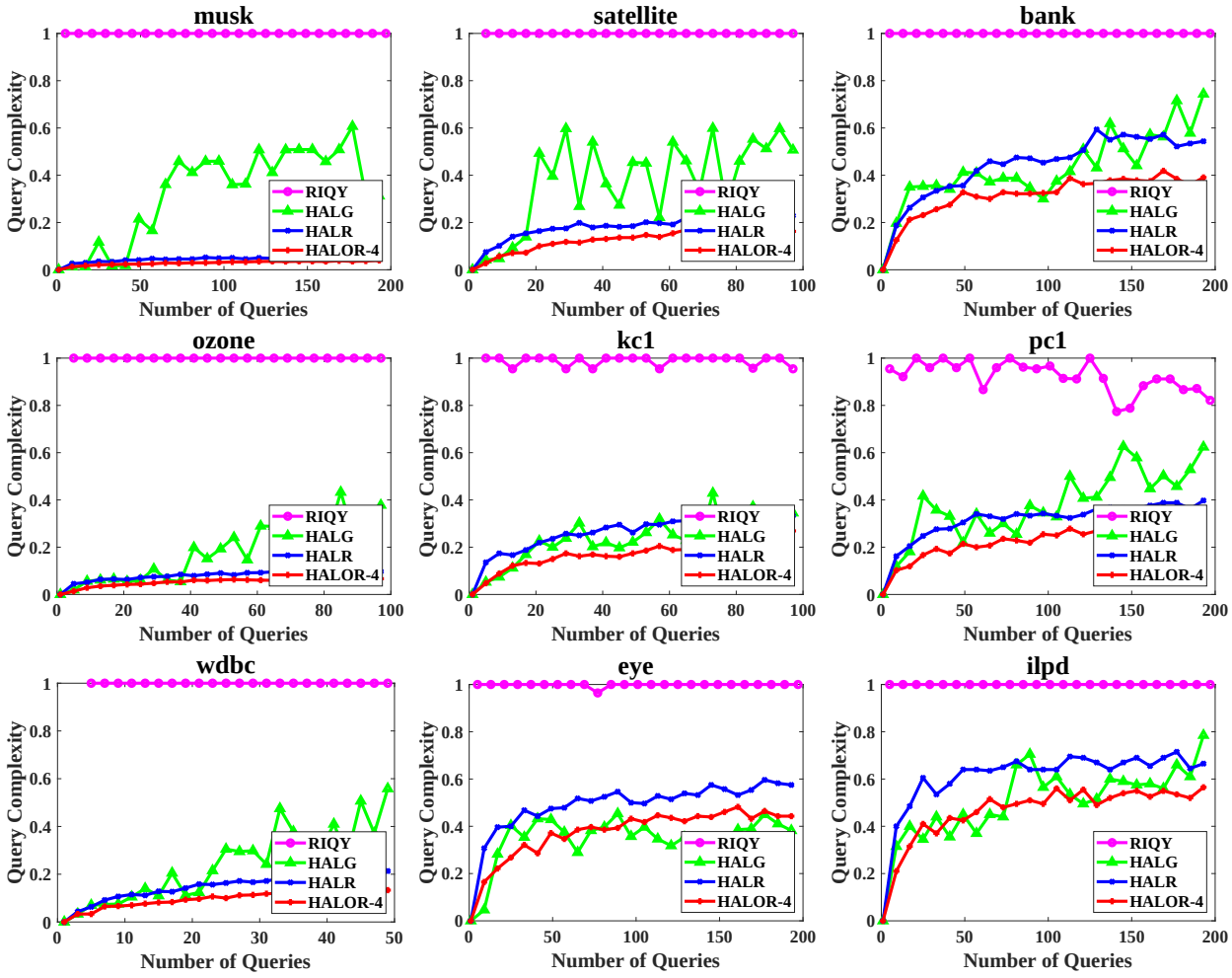

Figure 6.4: Query complexity of regions on the first 9 data sets.

6.6.5 Results: Query Complexity Vs. Number Of Queries

The analysis of classifier model performance vs. the number of queries is just one way to compare the quality of the active learning algorithms. Another important measure closely related to annotation cost is the complexity of the queries. In general, simpler queries are easier for human annotators to review and assess, while more complex queries may take extra time to review. Along the same lines (as pointed out in the introduction section), labeling of data instances can be very intricate for human annotators to do if instances have many features and when features are expressed in terms of high precision numbers. In contrast to this, well-built region-based queries may be much simpler, to understand and annotate for

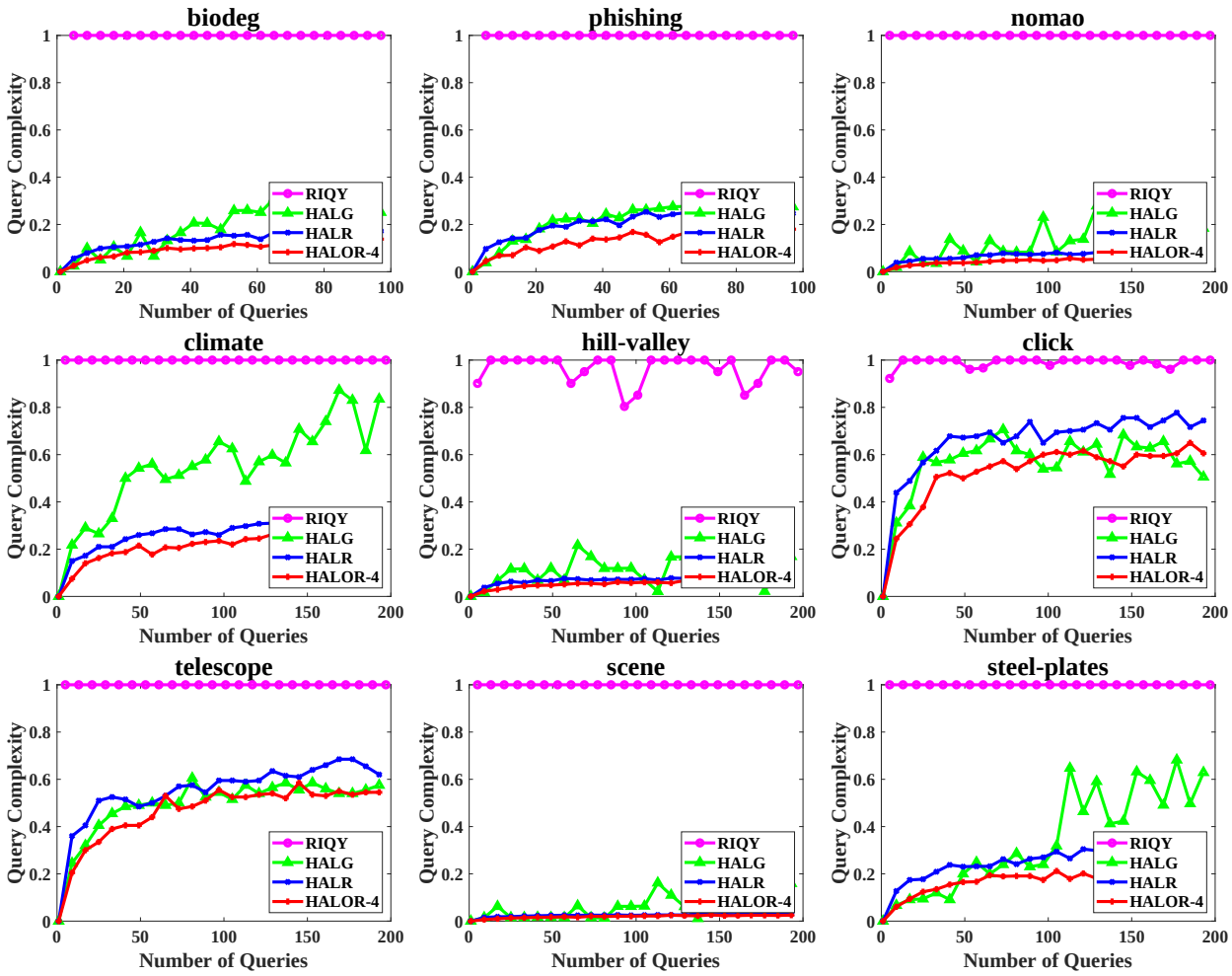

Figure 6.5: Query complexity of regions on the second 9 data sets.

human as each region can be described using a limited number of features.

To measure this aspect of queries, we define the complexity of a region query as:

$$\text{Complexity}(query) = \frac{\#\text{ of features used in the } query}{\text{total }\#\text{ of all features}} \tag{6.15}$$

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 6.4 and 6.5 shows and compares the query complexities of all region-based methods. Here are some of the findings:

- First of all, we note that instance-based queries use all features so their query complexity is 1. As opposed to this, region-based queries often rely only on a small subset of features. For example, it could be extremely time-consuming and difficult for human annotators to review a record in high-dimensional data sets `musk` (167 features), `nomao` (118), `hill-valley` (100) and `scene` (299). In contrast to this, our methods (HALR and HALOR), rely only on a few features ($< 5\%$) to describe generic regions. Such a property makes the labeling process a lot easier for a human to perform.
- The query complexity of our hierarchical active learning methods (HALG, HALR and HALOR) increases at best logarithmically and at most linearly with the total number of queries. It is because our hierarchical methods split regions and each split increases the complexity of the region by one condition. 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. In contrast, HALR and HALOR increase more stably because they build decision tree like hierarchies that add new feature dimension to the queries incrementally. Please also note that the query complexity of HALOR that uses multiple trees in parallel typically grow slower that HALR that relies on just one tree. The reason is that HALOR promotes simpler region descriptions.
- Lastly, we see RIQY's curves are very close to 1 which means it often uses almost all the features to describe regions. There are two reasons for this. First, because RIQY tends to construct very small and compact regions, it leads to induce regions of high complexity defined by many conditions (features). Second, as we mentioned earlier, RIQY has the query degeneration problem which means it fails to construct a pure enough region so it compromises to building an instance-based query instead. This problem directly results in the query complexity to be 1.

6.6.6 Analysis Of HALOR Performance With Different Number Of Trees

The hyper-parameter K in HALOR-K approach determines how many trees are grown in parallel. Figures 6.6 and 6.7 show how the model performance and query complexity vary with the number of trees K. To save space, we only plot the results for the top 9 data sets

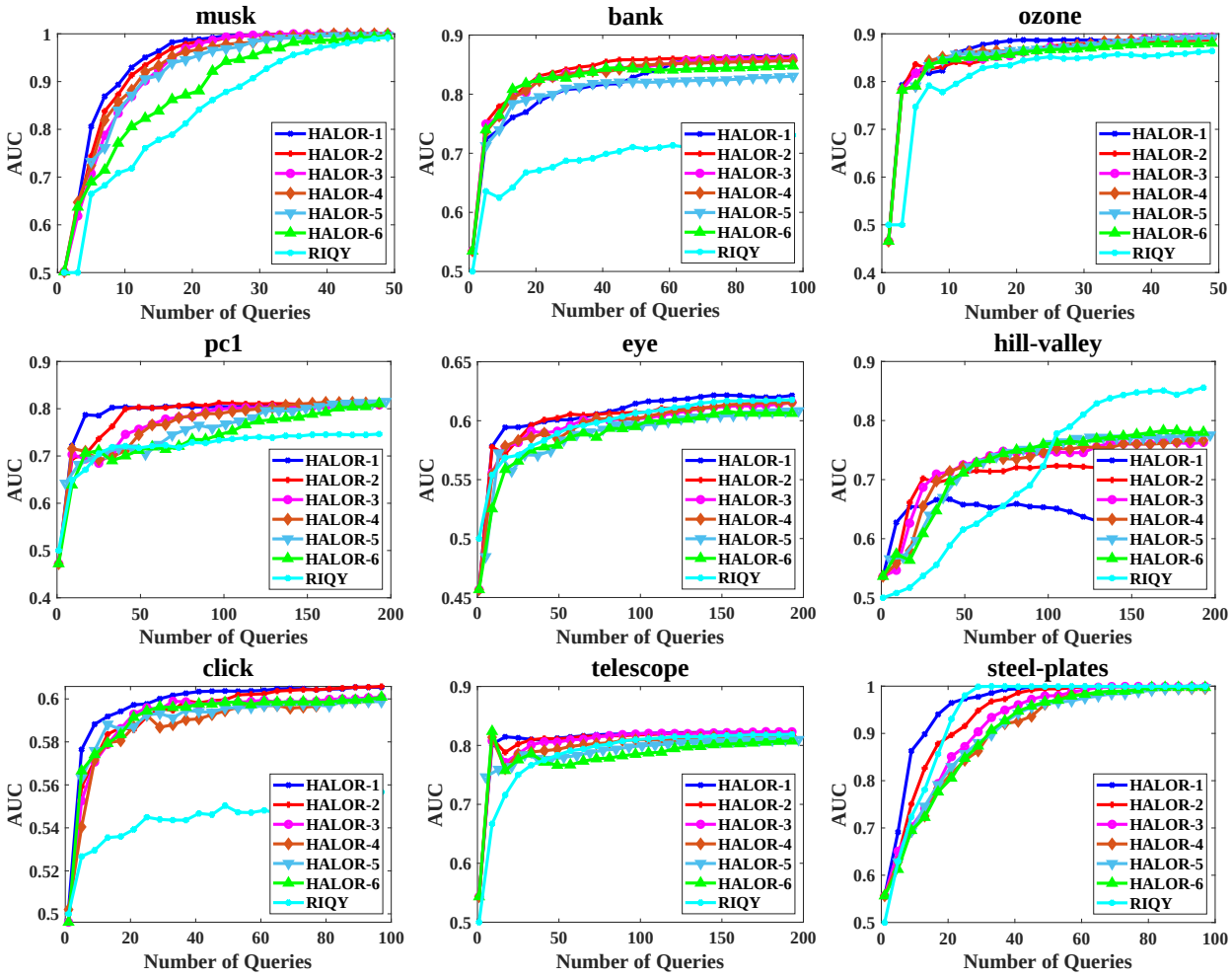

Figure 6.6: Model performance for various number of trees in HALOR.

for which the performance of different versions of HALORs differ the most (the remaining nine datasets show only little differences for different values of K). The figures and results therein illustrate a couple of trade-offs between HALR and HALOR:

- In terms of query complexity, apparently, growing more trees in HALOR translates into simpler region and hence simpler queries. This fact is clearly demonstrated by Figure 6.7.
- In terms of model performance, it initially takes more queries for HALOR to explore the feature information in the different trees. This results in slower performance improvement in the initial (exploration) phase. However, once HALOR identifies trees with class-

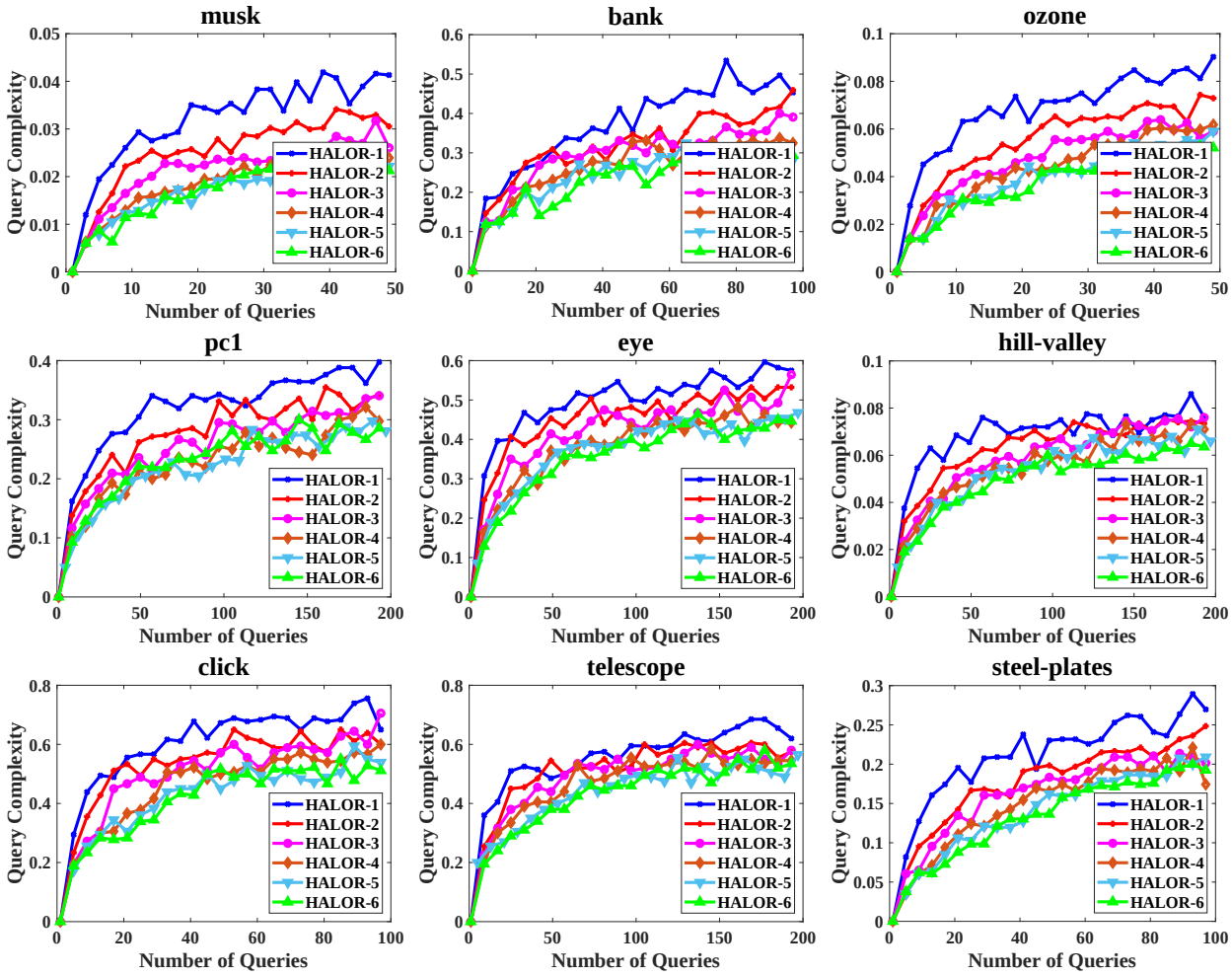

Figure 6.7: Query complexity for various number of trees in HALOR.

relevant features, it has a higher potential of further model performance improvement. As a result, the HALOR improvement accelerates. The results further suggest that usually 2 or 3 trees in HALOR would be good enough to learn models well from the tree. See the performance boosting from HALOR-1 to HALOR-2 on `hill-valley`.

6.7 Chapter Summary

In this chapter, we presented HALOR, a third implementation of the HAL framework that can actively learn instance-based classification models from region proportion feedback. The essential idea of HALOR is to build multiple region hierarchies simultaneously which provide more flexibility for finding informative and simple regions. When building only one hierarchy (i.e. HALOR-1), we adopt *maximum information gain* or *maximum impurity reduction* principle that is used to grow decision trees. In particular, when determining each split we consider all possible splits that can be made on all the leaf regions and adopt a combined information gain which is a weighted sum of a supervised gain and an unsupervised gain to evaluate each candidate split. When building multiple hierarchies (i.e. HALOR-K with $K > 1$) we diversify the feature combinations that construct different hierarchies. Finally, we adopt a general LLP learning algorithm that can learn instance-based models from a set of overlapping regions.

Compared to HALR, although being slightly less query-efficient than HALR, HALOR is shown to be a more robust approach. According to the experiment results, we see that when the unsupervised heuristic fails to suggest a good region hierarchy, HALOR can quickly build different hierarchies that are more class-relevant. Usually building two or three hierarchies appears sufficient. Besides, building multiple shallower hierarchies also reduces the query complexity of regions and thus helps avoid asking too complex queries to human annotators.

Through experiments on a much larger number of datasets, we have demonstrated that HALOR can learn good models from a very few and simple region queries. Compared to two instance-based active learning methods (DWUS and HS) and one group-based active learning approach (RIQY), HALOR is demonstrated more query-efficient. Therefore, the results again support the two hypothesis **H1** and **H2** (Section § 1.4).

7.0 Conclusions And Future Work

7.1 Thesis Summary

This thesis has developed and explored a region-based active learning framework that efficiently learns classification models from human feedback. A region is defined as a hypercubic subspace of the input space and represents a subpopulation of instances. Regions are described by conjunctive patterns that are easy to understand for humans. Annotators label a region with an estimate of the proportion of the instance subpopulation that belongs to one of the classes. By leveraging learning from label proportions (LLP) algorithms, the framework can learn a variety of instance-based classification models from labeled regions.

The motivation for studying region-based active learning methods is that learning with regions can be annotation-efficient. In many applications, it may be easier for human annotators to label regions than to label instances. Conventional data instances are usually recorded by numerous and detailed feature values while regions can be described by a concise feature set. Moreover, when regions are actively formed, the number of queries made for regions can be reduced, and thus the human annotation effort can be further saved.

To identify meaningful regions for labeling and learning, we have developed a novel hierarchical active learning framework named **HAL**. It actively builds one or multiple hierarchies of regions and queries regions in a top-down, general-to-specific fashion. The key is to quickly identify pure leaf regions in the hierarchy, which ensures the LLP learning success. We have implemented three versions of the HAL framework: (1) HALG that pre-compiles a hierarchy of clusters and then selects groups (hypercubic clusters) within the hierarchy for labeling and learning, (2) HALR that dynamically builds a decision tree like region hierarchy and employs two heuristics for region splits, and (3) HALOR that builds multiple different hierarchies simultaneously and provides additional flexibility and robustness for finding more informative and simpler regions.

Among these three implementations, the major difference is how much they rely on the unsupervised heuristic (clustering). We have seen that clustering overly dominates HALG,

and HALR reduces the reliance of clustering heuristic by dynamically controlling it; HALOR relies on clustering least since it has an initialization phase to explore class-relevant features that are used to construct informative hierarchies. Another difference lies in the query complexity of regions. We have seen that HALR and HALOR can learn models from very simple regions that only need a few features (<10 on average). This property greatly simplifies the labeling process when the input data instances are recorded by a large number of features.

Through comprehensive experiments on a number of datasets, we have demonstrated that our HAL framework is able to learn high-quality models from a very few and simple region-based queries. Also, it is more query-efficient than instance-based active learning methods, thereby providing empirical support for the first hypothesis **H1**.

H1: *Active learning of binary classification models from group-based feedback can be more query-efficient than learning from instance-based feedback.*

Compared to another region-based active learning method RIQY, our solutions demonstrate higher stability and faster model convergence while using simpler queries. Therefore, the second hypothesis **H2** is also supported empirically.

H2: *Our hierarchical approach can discover more informative and simpler groups than existing methods that identify groups based upon instances, and therefore our solution is more query-efficient.*

7.2 Assumptions Of The Current Work

While the HAL framework may successfully solve many classification learning problems, it is important to note the key assumptions and possible limitations of our studies.

- **Simulated region annotations.** All experiments in this thesis so far were conducted using simulated region annotations that are derived from available instance labels and their class proportions. A natural open question that arises is how these methods work when regions are labeled by humans. Of particular concern here are labeling noise, labeling bias, and inability of humans to provide accurate proportion labels to region-based queries. We will present a small pilot study addressing this question in the next

section.

- **Clustering heuristic.** One assumption the HAL relies on when it identifies regions is that the structure of data mimics their class distribution. This assumption has been used frequently in semi-supervised learning [Zhu et al., 2003], and the logic is that similar data instances tend to carry similar class labels. This is an important assumption in our HAL framework as we have frequently used the clustering heuristic to help build region hierarchies. If it does not hold then the single-tree solutions HALG and HALR can fail. The multi-tree solution (HALOR) is also affected but to a lesser degree. Overall, according to our last set of experiments, we have seen positive support for this assumption.
- **Supervised heuristic.** The second heuristic that HAL relies on is supervised heuristic. It uses a specific family of classification models to split regions. In all our experiments, we used linear classification models. However, we note this model may not be the best choice for all the problems explored. Therefore, using different families of models may lead to different performances of HAL, as well as, other alternative methods.
- **Annotation cost.** Throughout this work, we assumed that the cost of answering a region-based query by human annotators is comparable to answering an instance-based query. This is the assumption we use when comparing the query-efficiencies of different active learning methods in terms of the number of queries. Moreover, we assume that the cost of answering region-based queries is proportional to their query complexities (i.e. how many features are used in their conjunctive patterns.) These assumptions may not hold when a region-based query is simplistic, non-informative, or overly complex to assess and review. In such cases, the cost of answering a query is nonlinear with respect to the number of features. Finally, we note that annotation costs may differ for the different data types (say text, images), from application to application. Also, data objects with high-dimensional nature can make them intrinsically hard to structure and annotate. We will address some of these issues later in this chapter.
- **Binary classification learning.** Lastly, the current work only deals with binary classification problems. It is not immediately clear how annotators can provide similar multi-class proportion feedback. Later in this chapter, we will discuss some new types of feedback that can be potentially used for these problems.

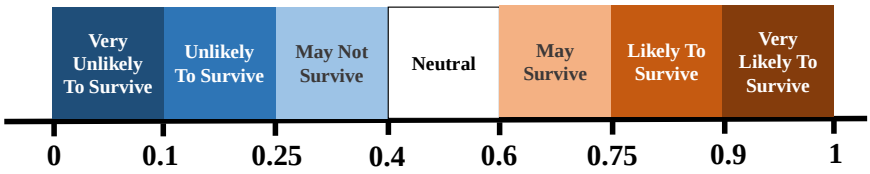

Figure 7.1: We use ordinal proportion categories as the human feedback. The seven categories partition the whole range of the positive class proportion values.

In the next section, we present a preliminary study showing that the HAL framework may indeed learn from human expert feedback. After that, we list open issues and future research directions that would allow us to relax some of the assumptions of the framework and make it applicable to a broader set of classification problems.

7.3 A Pilot Study: Human Annotation Of Regions

In this section, we present a pilot study to evaluate our HAL framework by interfacing it with a human annotator. We interact with an oncologist who provides feedback on whether a set of patients with colorectal cancer could survive three years or not. The data are publicly accessible and are collected from the U.S. NIH cancer database [NIH-SEER, 2019].

7.3.1 Methodology

For the purpose of this study, we have built a new implementation of our HAL framework named **HALQ** (**HAL** with **Q**ualitative feedback). It is similar to HALR, in that it dynamically builds a hierarchy of regions. However, instead of using numerical proportion feedback it relies on ordinal proportion categories (Figure 7.1). An example of one hierarchy built by HALQ is shown in Figure 7.2.

The HALQ has been modified to address some practical issues when compared to the previous implementations. There are three such modifications. The first one is to address the labeling noise issue that is concerned with providing exact class proportions to regions.

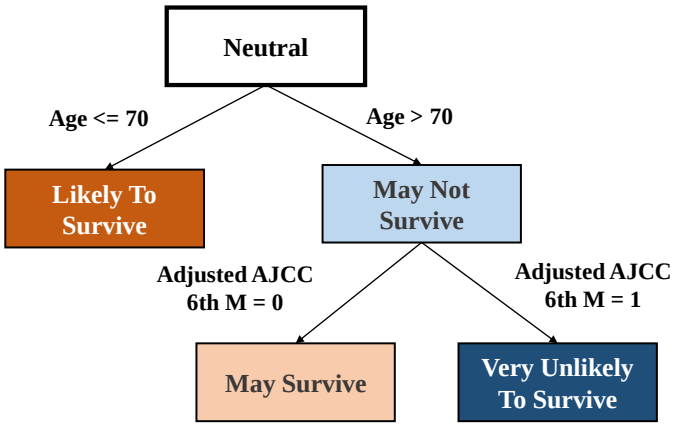

Figure 7.2: An example of a hierarchy of regions built by HALQ for survival analysis of colorectal cancer patients. The hierarchy has made two splits: the first one uses attribute *Age* (at value *70*); the second one uses attribute *Adjusted AJCC 6th M* which is a key factor for determining the cancer patient survival [NIH-SEER, 2019]. Instead of exact proportion estimates, we assume regions are assigned to one of the ordinal proportion categories. The framework repeatedly selects the most uncertain leaf region, splits it, and asks for the oncologist to assess the new leaf regions.

We mitigate it by designing a small set of ordinal categories as the region feedback. These categories are formed by coarsely-defined ranges of proportions as shown in Figure 7.1. Such qualitative feedback has been shown precise in assessing human attitudes in psychological studies [Likert, 1932, Ekman, 1978].

As qualitative feedback loses labeling precision, the second modification is to additionally solicit from annotators *discriminative feature feedback* that is used to split regions. Such feature feedback often comes with no extra cost [Druck et al., 2009, Dasgupta et al., 2018] but can substantially help construct discriminative regions. More concretely, when determining how to split a most uncertain region we ask the annotator to provide a splitting suggestion using any of the following granularity:

1. Provide both the splitting dimension d and value v;
2. Provide only the splitting dimension d;

3. Provide neither d nor v.

The first option is applicable when d is a categorical or a binary feature. For example, *gender* may be a key factor for diagnosis. The second option may be used most often since usually people may know what features are discriminative but are not exactly sure where the split values are. The last option serves as a backup. In either situation 2 or 3, HALQ needs to apply the two splitting heuristics as before to complement the split.

The last modification is to change the LLP learning algorithm to adapt to the qualitative feedback. We simply change the loss function in Formula 6.13. Recall that the loss minimizes a sum of region-level proportions $(\hat{\mu}_i - \mu_i)^2$ where $\hat{\mu}_i$ is a model estimate of the class proportion for region R_i, and μ_i the true proportion. Here we modify it as $[I(\hat{\mu}_i, l_i)]^2$ that is defined as:

$$I(\hat{\mu}_i, l_i) = \begin{cases} |\hat{\mu}_i - ub(l_i)|, & \text{if } \hat{\mu}_i > ub(l_i) \\ |\hat{\mu}_i - lb(l_i)|, & \text{else if } \hat{\mu}_i < lb(l_i) \\ 0, & \text{otherwise} \end{cases}$$

In above, l_i is one of the ordinal categories defined in Figure 7.1 and annotated to region R_i, and $lb(l), ub(l)$ are the lower, upper range boundaries of a label l. After the substitution, a new loss function is defined similarly as follows:

$$\mathcal{L}(F^{(t)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{n_i}{n} [I(\hat{\mu}_i, l_i)]^2 + \lambda \mathcal{C}(\boldsymbol{\theta}) \tag{7.1}$$

The optimization procedure remains the same and an instance-based model can be learned when the new loss function is minimized.

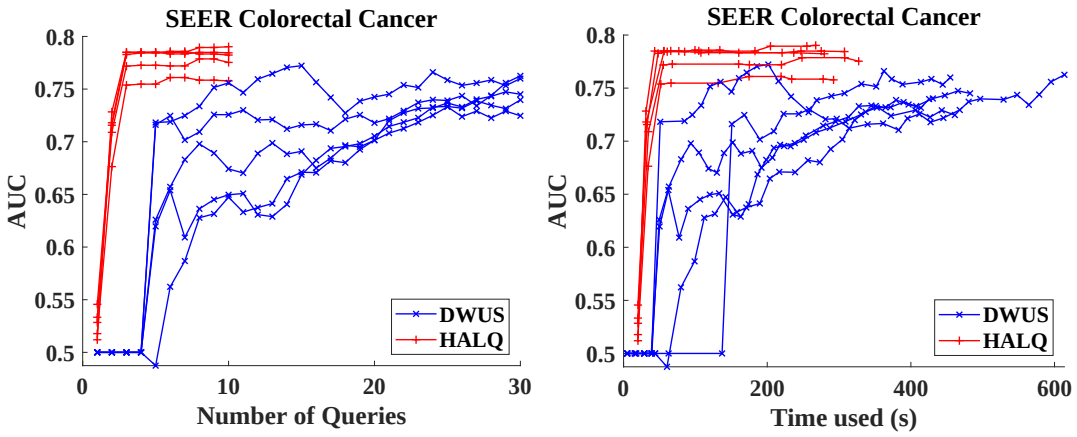

Figure 7.3: Performance of different methods in the real study.

7.3.2 Experiments

7.3.2.1 Data We randomly select 5000 colorectal cancer patient cases covering years 1973 to 2015. We also pre-select 10 features as the predictors and assign binary survival labels according to the mortality record. The labels indicate whether patients can survive the following three years or not upon diagnosis. Overall, there are 68.13% patients labeled positive (survived). Note that we only use the survival labels for test data; for training data, the oncologist provides the labels. The 10 features are:

- **Numeric**: *Age, CS tumor size*
- **Ordinal**: *Cancer grade, Adjusted AJCC 6th T, Adjusted AJCC 6th N, Total number of situ/malignant tumors*
- **Categorical**: *Marital status, Sex, Adjusted AJCC 6th M*

7.3.2.2 Experimental Settings We compare our HALQ to DWUS that represents instance-based active learning methods. The base classification model is still Logistic Regression. To reduce the randomness in experiments we perform 5 runs of different training/test data split. The split ratio is still 3:1. We plot AUC score on test data vs. the number of queries or the time used (in second). We terminate the running of each method whenever the test AUC score becomes stable.

7.3.2.3 Results And Analysis

Before we present the results, there are some facts about the annotation process. When the oncologist suggests features for splitting regions, 84% of the time he provides only the splitting features; 15% of the time he provides both the splitting feature and the value; <1% he provides nothing. The most important features he provides are ranked as *Age, Adjusted AJCC 6th M, Adjusted AJCC 6th N* and *Cancer grade*. The results are shown in Figure 7.3. We plot all the 5 runs. They demonstrate strong evidence that HALQ is both more label- and time-efficient than DWUS. Furthermore, we can see that HALQ, in general, has a more stable performance. Again, because regions can better cover the input space and present larger populations of instances, learning with regions yields steady learning curves. By comparison, learning with a limited number of instances may be sensitive to newly labeled ones. This is a typical problem in instance-based active learning methods.

7.3.3 Discussion

In this pilot study, we have presented HALQ, a more practical implementation of the HAL framework. Compared to other HAL implementations, HALQ overcomes the limitation of using exact proportion labels by adopting qualitative feedback for region assessment. Moreover, by allowing humans to provide feature feedback on constructing regions, HALQ is shown also efficient in constructing informative region hierarchies. The experiment results support that HALQ is both label- and time-efficient to build binary classification models.

There is one potential issue of HALQ concerned with its solicitation of feature feedback from human annotators. As humans can be subjective, their suggestions of discriminative features can be very different and thus result in constructing different hierarchies. One alleviation to the annotation bias issue is to have multiple annotators build different hierarchies and grow those with more informative regions. This is in spirit to the HALOR implementation.

7.4 Open Questions And Future Directions

The region-based active learning methods proposed in this thesis form a new direction of learning classification models from an alternative human feedback. Its novelty creates many opportunities for further refinement and its application to new problems. It also leaves some interesting open issues and problems to address. In the following, we list some of these new future directions and open problems.

- **Query complexity and the cost of human annotation.** One important open question regards the actual difficulty of understanding and answering region-based queries by human annotators. We assumed that in general, simpler queries defined by shorter conjunctive patterns are easier to review and assess. Also, we assumed the cost of answering more complex queries is proportional to the query complexity defined in terms of the number of features used in the query. However, these assumptions may not hold in practice. For example, if some queries are very short they may not be very informative, especially when they contain features that are irrelevant to the classification problem, then proportion labels may be hard to provide for humans. We expect that less informative queries increase the time of query reviewing and assessment. At the other extreme, if some queries are too complex and contain complicated feature combinations then understanding such queries could be mentally fatiguing; thus, the time and cost needed to review them could become a non-linear function of their complexity. Therefore, finding concise and relevant feature combinations for region description is the key to easing the human annotation process. A better understanding of all these annotation trade-offs could lead to better and more cost-effective querying strategies.
- **Noisy proportion label estimates.** Another important question is related to the quality of class proportion estimates that are used to assess regions. In general, humans are not very accurate in estimating class proportions so the feedback received from human annotators could come with various biases and noise. An open question is how such a labeling noise could affect the construction of hierarchies and the learned models. This aspect of the problem prompts a more rigorous exploration of the group learning methods and their robustness.

- **Multi-class classification problems.** Another set of open questions is related to the extension of the HAL framework to multi-class and multi-label classification problems. For multi-class classification problems, the main obstacle lies in whether human annotators can provide accurate class proportions to regions for all the classes. If this is feasible for humans to do (e.g. when the number of classes is small), then our HAL framework can be directly applied (note that LLP algorithms can learn multi-class classifiers). However, if such an annotation process is costly or infeasible then we need to solicit different feedback and modify our framework. Xue and Hauskrecht [Xue and Hauskrecht, 2018] have dealt with a similar situation for annotating instances (not regions) for multi-class classification problems. One of their solutions [Xue and Hauskrecht, 2018] was to ask the user to only identify the most likely class and the soft label was provided only for that class. The model learning algorithm was also changed accordingly so as to adapt to such feedback.
- **Multi-label classification problems.** Multi-label classification problem is different. In general, labels assigned to instances are not independent of each other, so many state-of-the-art multi-label classification algorithms such as [Batal et al., 2013, Pakdaman et al., 2014, Hong et al., 2014, Hong et al., 2015] as well as
  [Hong and Hauskrecht, 2015b, Hong and Hauskrecht, 2015a] are trying to assure that the dependencies among different labels are modeled properly. One possible direction to adapt our region-based algorithms to multi-label settings is to rely on chain classification approach that decomposes the multi-label classification problem into a chain of single-label classification sub-problems:

$$P(y_1, y_2, \cdots, y_K|\boldsymbol{x}) = P(y_1|\boldsymbol{x})P(y_2|y_1, \boldsymbol{x}) \cdots P(y_K|y_1, y_2, \cdots, y_{K-1}, \boldsymbol{x})$$

  where the learning of each single-label classifier is solved by our group-based model. In terms of how human annotators assign different class labels to groups, if each y_i is a binary class label then providing class proportions is still possible. However, if there are labels that are multi-classed then the annotation process could become much harder. In light of this, [Xue and Hauskrecht, 2019] has proposed a efficient protocol for instance-based annotation. They ask for an ordered set of classes without soliciting exact proportions

and then learn relative ranking functions that are further used to build classification models.

- **Time-series classification.** One of our current assumptions is that instances are defined by a fixed set of features and their values. Regions are hypercubic subspaces that are defined by this feature space. However, the data objects we often need to classify can be more complex and their featurization is not always straightforward or human-friendly. One such data type is time series. Briefly, time series are formed by sequences of observations. Two classification tasks are typically defined upon time series [Batal et al., 2016]: time-series classification and time-series or event prediction. Both of these tasks typically transform time series to a different feature space for problem-solving. Examples of such transformations for time series classification problems are discrete Fourier or discrete wavelet transform [Batal and Hauskrecht, 2009]. On the other hand, time series prediction tasks are most often handled by using various kinds of latent models and latent representations based on models, such as Hidden Markov Models, Linear dynamic models [Liu and Hauskrecht, 2015b, Liu et al., 2013, Liu and Hauskrecht, 2015a, Liu and Hauskrecht, 2016a, Liu and Hauskrecht, 2016b], point processes
  [Liu and Hauskrecht, 2019, Liu et al., 2017a, Liu et al., 2017b, Liu et al., 2018], or more recently, models based on recurrent neural networks (RNNs) [Lee and Hauskrecht, 2019, Lee and Hauskrecht, 2020]. Unfortunately, these latent feature representations are typically not very friendly for a human to review and assess. One possible direction to extend our work to time-series problems is to build upon predictive temporal pattern mining solutions. Briefly, predictive temporal pattern mining methods [Batal et al., 2012, Batal et al., 2016, Batal et al., 2009b, Batal et al., 2009a] use time series and their value and trend abstractions to define temporal patterns that can characterize and describe the time series in terms of the original variables and their temporal behaviors. The basic temporal patterns can be combined with more complex temporal patterns with temporal logic operations and by making conjunctions of such patterns. The advantage of such representations is that they are friendly for humans to read and interpret [Batal et al., 2012, Batal et al., 2016]. In addition, more complex temporal patterns are typically built hierarchically via apriori-style algorithms. It is similar to our hierarchical

decision tree group refinement strategies which makes it a viable solution for defining region hierarchies.

- **Classification of images and text.** Apart from time series, other data objects and data types need to be often classified. Take images or texts for an example, the data instances and their representations for these two data types are typically high-dimensional. Hence, using conjunctive patterns to summarize groups of images or documents from these representations could be very hard. However, we can abstract the idea of HAL and adapt to these data types. The essence of the group-based labeling technique is summarizing groups in a human-understandable way and soliciting a meta-label for the group. Using conjunctive patterns and class proportions to describe and label groups is just one specific way. Therefore, if images or texts can be summarized and labeled by humans, then the HAL framework can be potentially applied. Currently, there are many automatic summarization techniques for images [Tschiatschek et al., 2014, Yang et al., 2013] and texts [Gambhir and Gupta, 2017]. How to integrate such techniques into a more general version of HAL is an interesting direction.
- **Multi-task learning.** Multi-task learning refers to a category of machine learning methods that learn multiple related tasks simultaneously. The motivation is to exploit task relationships, commonalities, and differences. Multi-task learning methods can take advantage of available data for similar learning tasks by imposing similarities between the behavior of related models. Multi-task learning has shown promising results in improving individual tasks compared to the standard solution of learning each task independently [Malakouti and Hauskrecht, 2019]. Our current work can be extended to learning multiple tasks that are associated with each other. For example, in the work of [Malakouti and Hauskrecht, 2020], the tasks to learn are hierarchically structured and thus the information from the parent tasks can be inherited to the child tasks. In our work, if the learned region hierarchy of one task can be used to initialize the hierarchy of another one, then the active learning efficiency can be further improved.

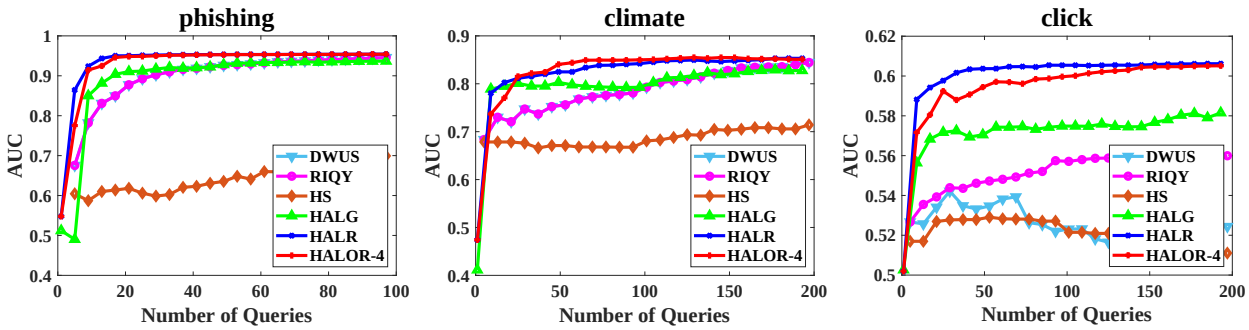

Figure 7.4: Model performance of different methods on 3 data sets.

7.5 Sample-Efficiency Of Learning From Groups

In this section, we introduce and describe an interesting theoretical direction that studies more systematically the complexity of the learning from groups framework.

Throughout this thesis, we have seen empirical results that support our first hypothesis: **H1**: *Active learning of binary classification models from group-based feedback is more query-efficient than learning from instance-based feedback.*

Two possible reasons that could explain this hypothesis are:

1. Our active learning strategies are more effective;
2. Learning from group proportion feedback itself (without the active learning component) is more sample-efficient than learning from instance-based feedback.

But which of the two reasons is more important? Our conjecture is that the second one. We have observed empirical support for this conjecture when the number of group queries was very small (for example, when we had less than 10 queries). In this case, learning from labeled groups was able to output a decent model while learning from labeled instances typically failed. This fact is supported by three example data sets illustrated in Figure 7.4. We believe that the performance gap shown in the plot is less likely due to the active learning strategies. The reason is that initially there is a very little supervision which is unable to produce very effective supervised strategies. Instead, the performance gap should be mostly attributed to the group proportion feedback, which is able to cover larger portions of the

input data space and thereby provides broader class information than instance labels.

An interesting research direction related to the above observation could be the study of group learning with class proportion labels when groups are i.i.d. generated and its efficiency compared to learning from labeled instances. More precisely, consider the following statement:

Under some conditions and for some family of binary classification functions, learning the true function from i.i.d. generated groups achieves lower sample complexity than learning from i.i.d. generated instances.

In other words, given the same number of i.i.d. labeled groups and instances, the question is if learning from groups can output a more accurate classification function than learning from instances. If the above statement holds, it will also benefit instance-based active learning methods. One application would be providing some general groups and their proportion feedback to initialize a classification model and then using it for the successive active learning process, either instance-based or group-based.

7.5.1 Probably Approximately Correct

Studying sample complexity (i.e. how many training data are needed to estimate a distribution or to learn a good model) has a long history in statistics and machine learning research [Vapnik, 1998, Van der Vaart, 2000, Friedman et al., 2001]. One useful tool is the *probably approximately correct* model (PAC) [Valiant, 1984]. The intent of the PAC model is that successful learning of an unknown target concept (i.e. the true classification function) should entail obtaining, with high probability, a hypothesis that is a good approximation of it. Hence the name Probably Approximately Correct. In other words, it studies the sample complexity if the hypothesis learned from the sample is ϵ-close to the target concept, with high probability $1-\delta$. To qualify as a PAC learning algorithm, it must satisfy this guarantee for all possible target concepts in a given family and under all possible data distributions. To achieve this objective, the learning algorithm is supplied with a training sample that consists of n i.i.d. labeled data instances, along with the corresponding correct classifications. One of the central questions in the study of PAC learning is determining the minimum number

$\mathcal{S}(\epsilon, \delta)$ of training instances necessary and sufficient such that there exists a PAC learning algorithm requiring at most $\mathcal{S}(\epsilon, \delta)$ labeled data (for any given ϵ and δ). This quantity $\mathcal{S}(\epsilon, \delta)$ is known as the sample complexity.

We begin by introducing some background essential to the following discussion. Let $\mathcal{X}$ be a nonempty set called the input space and $\mathcal{Y} = \{0, 1\}$ the class space. A classifier is any function of mapping $h : \mathcal{X} \to \mathcal{Y}$. Also, fix a nonempty set $\mathbb{C}$ of classifiers called the *concept space*. Conventional to PAC, we use $f^* \in \mathbb{C}$ to denote the *target concept*, and use h to denote the *hypothesis* learned from a sample of data. Note that h is not necessarily in $\mathbb{C}$.

In a learning problem, assume there is an instance distribution $\mathcal{P}$ over $\mathcal{X}$. Then we can collect one training sample $\mathcal{S}_n = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$, where each $\boldsymbol{x}_i$ is independently sampled according to $\mathcal{P}$ and y_i is the true label of $\boldsymbol{x_i}$, i.e. $y_i = f^*(\boldsymbol{x_i})$. For any probability measure $\mathcal{P}$ over $\mathcal{X}$ and any classifier h, denote by $err_{\mathcal{P}}(h, f^*) = P(\boldsymbol{x} : h(\boldsymbol{x}) \neq f^*(\boldsymbol{x}))$. Then for any $\epsilon, \delta \in (0, 1)$, the sample complexity $\mathcal{S}(\epsilon, \delta)$, is defined as the smallest positive integer for which there exists a learning algorithm that outputs h, such that for every possible data distribution $\mathcal{P}$ and all the target concept $f^* \in \mathbb{C}$, the following inequality holds:

$$P(err_{\mathcal{P}}(h, f^*) \leq \epsilon) \geq 1 - \delta$$

Determining the sample complexity of PAC learning is a long-standing open problem. There have been upper and lower bounds established for decades. Before we present the known results, we should introduce the concept of the Vapnik-Chervonenkis dimension (or VC dimension) [Vapnik, 1998]. This quantity is of fundamental importance in characterizing the sample complexity of PAC learning. Roughly speaking, the VC dimension of a concept class $\mathbb{C}$ measures how complex the classification functions in $\mathbb{C}$ could be. For binary classifications, each instance $\boldsymbol{x}$ can be mapped to two outcomes, 0 or 1. So for any vector consisting of k instances $\langle \boldsymbol{x}_1, ..., \boldsymbol{x}_k \rangle$, there could be potentially 2^k distinct classifications for the k instances. So we say a vector $\langle \boldsymbol{x}_1, ..., \boldsymbol{x}_k \rangle$ of k points in $\mathcal{X}$ is *shattered* by $\mathbb{C}$ if $\forall y_1, ..., y_k \in \mathcal{Y}$, there exists an $f \in \mathbb{C}$ such that $\forall i \in \{1, ..., k\}$, $f(x_i) = y_i$. Then the VC dimension of $\mathbb{C}$ is defined as the largest such k for which there exists a vector $\langle \boldsymbol{x}_1, ..., \boldsymbol{x}_k \rangle$ in $\mathcal{X}$ shattered by $\mathbb{C}$. For example, the VC dimension of homogeneous linear classification function class in $\mathbf{R}^m$ is m. We denote by $\mathbf{VC}(\mathbb{C})$ the VC dimension of $\mathbb{C}$.

With VC dimension, we can present the known bounds of sample complexity. A basic lower bound of $\frac{1-\epsilon}{\epsilon}\log(1/\delta)$ was established by [Blumer et al., 1989] for $0 < \epsilon < 1/2$ and $0 < \delta < 1$. A second lower bound of $\frac{\mathbf{VC}(\mathbb{C})-1}{32\epsilon}$ was supplied by [Ehrenfeucht et al., 1989] for $0 < \epsilon < 1/8$ and $0 < \delta < 1/100$. Taken together, these two bounds yield a general lower bound, that for any $0 < \epsilon < 1/8$ and $0 < \delta < 1/100$,

$$\mathcal{S}(\epsilon, \delta) \geq \max\{\frac{\mathbf{VC}(\mathbb{C}) - 1}{32\epsilon}, \frac{1-\epsilon}{\epsilon}\log(1/\delta)\} = \Omega(\frac{1}{\epsilon}(\mathbf{VC}(\mathbb{C}) + \log(\frac{1}{\delta})))$$

This lower bound is complemented by classic upper bounds on the sample complexity. [Blumer et al., 1989] also established an upper bound of

$$\mathcal{S}(\epsilon, \delta) = O(\frac{1}{\epsilon}(\mathbf{VC}(\mathbb{C})\log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})))$$

So the general lower and upper bounds differ by a logarithmic factor.

As we have seen, the sample complexity has been well studied for the instance-based learning protocol. However, to our best knowledge, there is no known work that studies the sample complexity of learning from groups methods. Hence, this future work needs to explore (1) under what conditions, (2) for what kind of function classes, (3) and with what design of groups, the learning from groups methods achieve competing or lower sample complexity than methods that learn models from instances.

Bibliography

[Amores, 2013] Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105.

[Andrews et al., 2003] Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 577–584.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

[Balcan et al., 2009] Balcan, M.-F., Beygelzimer, A., and Langford, J. (2009). Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89.

[Bartlett and Mendelson, 2002] Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

[Batal et al., 2016] Batal, I., Cooper, G. F., Fradkin, D., Harrison, J., Moerchen, F., and Hauskrecht, M. (2016). An efficient pattern mining approach for event detection in multivariate temporal data. *Knowledge and information systems*, 46(1):115–150.

[Batal and Hauskrecht, 2009] Batal, I. and Hauskrecht, M. (2009). A supervised time series feature extraction technique using dct and dwt. In *International Conference on Machine Learning and Applications (ICMLA)*.

[Batal et al., 2013] Batal, I., Hong, C., and Hauskrecht, M. (2013). An efficient probabilistic framework for multi-dimensional classification. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, pages 2417–2422. ACM.

[Batal et al., 2009a] Batal, I., Sacchi, L., Bellazzi, R., and Hauskrecht, M. (2009a). Multivariate time series classification with temporal abstractions. In *Proceedings of the Florida Artificial Intelligence Research Society (FLAIRS)*.

[Batal et al., 2009b] Batal, I., Sacchi, L., Bellazzi, R., and Hauskrecht, M. (2009b). A temporal abstraction framework for classifying clinical temporal data. In *AMIA Annual Symposium Proceedings*.

[Batal et al., 2012] Batal, I., Valizadegan, H., Cooper, G. F., and Hauskrecht, M. (2012). A Temporal Pattern Mining Approach for Classifying Electronic Health Record Data. *ACM Transaction on Intelligent Systems and Technology (ACM TIST), Special Issue on Health Informatics*.

[Besbes et al., 2014] Besbes, O., Gur, Y., and Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *NIPS*, pages 199–207.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Blumer et al., 1989] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965.

[Bose and Mahapatra, 2001] Bose, I. and Mahapatra, R. K. (2001). Business data mining—a machine learning perspective. *Information & management*, 39(3):211–225.

[Cohn et al., 1996] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

[Dasgupta, 2011] Dasgupta, S. (2011). Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781.

[Dasgupta et al., 2018] Dasgupta, S., Dey, A., Roberts, N., and Sabato, S. (2018). Learning from discriminative feature feedback. In *Advances in Neural Information Processing Systems*, pages 3955–3963.

[Dasgupta and Hsu, 2008] Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM.

[Dietterich et al., 1997] Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71.

[Druck et al., 2009] Druck, G., Settles, B., and McCallum, A. (2009). Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 81–90. Association for Computational Linguistics.

[Du and Ling, 2010] Du, J. and Ling, C. X. (2010). Asking generalized queries to domain experts to improve learning. *Knowledge and Data Engineering, IEEE Transactions*, 22(6):812–825.

[Ehrenfeucht et al., 1989] Ehrenfeucht, A., Haussler, D., Kearns, M., and Valiant, L. (1989). A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261.

[Ekman, 1978] Ekman, P. (1978). Facial action coding system. *Consulting Psychologists Palo Alto*.

[Freytag et al., 2014] Freytag, A., Rodner, E., and Denzler, J. (2014). Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision*, pages 562–577. Springer.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*. Springer series in statistics New York, NY, USA:.

[Gambhir and Gupta, 2017] Gambhir, M. and Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

[Givens and Hoeting, 2012] Givens, G. H. and Hoeting, J. A. (2012). *Computational statistics*, volume 710. John Wiley & Sons.

[Hauskrecht et al., 2016] Hauskrecht, M., Batal, I., Hong, C., Nguyen, Q., Cooper, G. F., Visweswaran, S., and Clermont, G. (2016). Outlier-based detection of unusual patient-management actions: An icu study. *Journal of Biomedical Informatics*, 64:211 – 221.

[Hauskrecht et al., 2013] Hauskrecht, M., Batal, I., Valko, M., Visweswaran, S., Cooper, G. F., and Clermont, G. (2013). Outlier detection for patient monitoring and alerting. *Journal of Biomedical Informatics*, 46(1):47 – 55.

[Heim and Hauskrecht, 2015] Heim, E. and Hauskrecht, M. (2015). Sparse multidimensional patient modeling using auxiliary confidence labels. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 331–336. IEEE.

[Heim et al., 2014] Heim, E., Valizadegan, H., and Hauskrecht, M. (2014). Relative comparison kernel learning with auxiliary kernels. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 563–578. Springer.

[Hema et al., 2006] Hema, R., Omid, M., and Rosie, J. (2006). Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686.

[Hong et al., 2014] Hong, C., Batal, I., and Hauskrecht, M. (2014). A mixtures-of-trees framework for multi-label classification. In *Proceedings of the 23nd ACM International Conference on Information and Knowledge Management*, CIKM '14. ACM.

[Hong et al., 2015] Hong, C., Batal, I., and Hauskrecht, M. (2015). A generalized mixture framework for multi-label classification. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM.

[Hong and Hauskrecht, 2015a] Hong, C. and Hauskrecht, M. (2015a). MCODE: multivariate conditional outlier detection. *CoRR*, abs/1505.04097.

[Hong and Hauskrecht, 2015b] Hong, C. and Hauskrecht, M. (2015b). Multivariate conditional anomaly detection and its clinical application. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 4239–4240. AAAI Press.

[Hwa, 2004] Hwa, R. (2004). Sample selection for statistical parsing. *Computational linguistics*, 30(3):253–276.

[Ishibuchi et al., 1995] Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on fuzzy systems*, 3(3):260–270.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

[Jordan and Mitchell, 2015] Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[Kotzias et al., 2015] Kotzias, D., Denil, M., De Freitas, N., and Smyth, P. (2015). From group to individual labels using deep features. In *Proceedings of the 21th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 597–606. ACM.

[Kovashka et al., 2012] Kovashka, A., Parikh, D., and Grauman, K. (2012). Whittlesearch: Image search with relative attribute feedback. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2973–2980. IEEE.

[Kovashka et al., 2016] Kovashka, A., Russakovsky, O., Fei-Fei, L., Grauman, K., et al. (2016). Crowdsourcing in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 10(3):177–243.

[Kovashka et al., 2011] Kovashka, A., Vijayanarasimhan, S., and Grauman, K. (2011). Actively selecting annotations among objects and attributes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1403–1410. IEEE.

[Kück and de Freitas, 2005] Kück, H. and de Freitas, N. (2005). Learning about individuals from group statistics. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 332–339. AUAI Press.

[Lee and Hauskrecht, 2019] Lee, J. M. and Hauskrecht, M. (2019). Recent-context-aware LSTM-based Clinical Time-Series Prediction. In *In Proceedings of AI in Medicine Europe (AIME)*.

[Lee and Hauskrecht, 2020] Lee, J. M. and Hauskrecht, M. (2020). Multi-scale temporal memory for clinical event time-series prediction. In *In Proceedings of the International Conference on AI in Medicine (AIME)*.

[Lewis and Catlett, 1994] Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning Proceedings 1994*, pages 148–156. Elsevier.

[Likert, 1932] Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

[Liu and Hauskrecht, 2019] Liu, S. and Hauskrecht, M. (2019). Nonparametric regressive point processes based on conditional gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1062–1072.

[Liu et al., 2017a] Liu, S., Wright, A., and Hauskrecht, M. (2017a). Change-point detection method for clinical decision support system rule monitoring. In *International Conference on Artificial Intelligence in Medicine*, volume 10259, pages 126–135.

[Liu et al., 2018] Liu, S., Wright, A., and Hauskrecht, M. (2018). Change-point detection method for clinical decision support system rule monitoring. *Artificial Intelligence in Medicine*.

[Liu et al., 2017b] Liu, S., Wright, A., Sittig, D. F., and Hauskrecht, M. (2017b). Change-point detection for monitoring clinical decision support systems with a multi-process dynamic linear model. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 569–572. IEEE.

[Liu and Hauskrecht, 2015a] Liu, Z. and Hauskrecht, M. (2015a). Clinical time series prediction: Toward a hierarchical dynamical system framework. *Artificial Intelligence in Medicine*, 65(1):5–18.

[Liu and Hauskrecht, 2015b] Liu, Z. and Hauskrecht, M. (2015b). A regularized linear dynamical system framework for multivariate time series analysis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Liu and Hauskrecht, 2016a] Liu, Z. and Hauskrecht, M. (2016a). Learning adaptive forecasting models from irregularly sampled multivariate clinical data. In *The 30th AAAI Conference on Artificial Intelligence*, pages 1273–1279.

[Liu and Hauskrecht, 2016b] Liu, Z. and Hauskrecht, M. (2016b). Learning linear dynamical systems from multivariate time series: A matrix factorization based framework. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 810–818. SIAM.

[Liu et al., 2013] Liu, Z., Wu, L., and Hauskrecht, M. (2013). Modeling clinical time series using gaussian process sequences. In *SIAM International Conference on Data Mining*.

[Luo and Hauskrecht, 2017a] Luo, Z. and Hauskrecht, M. (2017a). Active learning of classification models from soft-labeled groups. In *Advances in Neural Information Processing Systems, Learning from Limited Data Workshop*.

[Luo and Hauskrecht, 2017b] Luo, Z. and Hauskrecht, M. (2017b). Group-based active learning of classification models. In *Proceedings of the 30th International Florida AI*

Research Society Conference. Florida AI Research Symposium, volume 2017, page 92. NIH Public Access.

[Luo and Hauskrecht, 2018a] Luo, Z. and Hauskrecht, M. (2018a). Hierarchical active learning with group proportion feedback. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI'18*, pages 2532–2538.

[Luo and Hauskrecht, 2018b] Luo, Z. and Hauskrecht, M. (2018b). Hierarchical active learning with proportion feedback on regions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 464–480. Springer.

[Luo and Hauskrecht, 2019] Luo, Z. and Hauskrecht, M. (2019). Region-based active learning with hierarchical and adaptive region construction. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 441–449. SIAM.

[Luo and Hauskrecht, 2020] Luo, Z. and Hauskrecht, M. (2020). Hierarchical active learning with overlapping regions. In *Proceedings of the 29th ACM international conference on Information and knowledge management*.

[Malakouti and Hauskrecht, 2019] Malakouti, S. and Hauskrecht, M. (2019). Hierarchical adaptive multi-task learning framework for patient diagnoses and diagnostic category classification. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*.

[Malakouti and Hauskrecht, 2020] Malakouti, S. and Hauskrecht, M. (2020). Not all samples are equal: Class dependent hierarchical multi-task learning for patient diagnosis classification. In *The Thirty-Third International Flairs Conference*. AAAI.

[Maron and Lozano-Pérez, 1998] Maron, O. and Lozano-Pérez, T. (1998). A framework for multiple-instance learning. In *Advances in neural information processing systems*, pages 570–576.

[Mjolsness and DeCoste, 2001] Mjolsness, E. and DeCoste, D. (2001). Machine learning for science: state of the art and future prospects. *science*, 293(5537):2051–2055.

[Neter et al., 1996] Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W. (1996). *Applied linear statistical models*, volume 4. Irwin Chicago.

[Nguyen and Smeulders, 2004] Nguyen, H. T. and Smeulders, A. (2004). Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79. ACM.

[Nguyen et al., 2011a] Nguyen, Q., Valizadegan, H., and Hauskrecht, M. (2011a). Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*, pages 477–486. IEEE.

[Nguyen et al., 2014] Nguyen, Q., Valizadegan, H., and Hauskrecht, M. (2014). Learning classification models with soft-label information. *Journal of the American Medical Informatics Association*, 21(3):501–508.

[Nguyen et al., 2011b] Nguyen, Q., Valizadegan, H., Seybert, A., and Hauskrecht, M. (2011b). Sample-efficient learning with auxiliary class-label information. In *AMIA Annual Symposium Proceedings*, volume 2011, page 1004. American Medical Informatics Association.

[NIH-SEER, 2019] NIH-SEER (2019). Surveillance, epedimeology, and end results (seer) program research data (1975-2016), national cancer institute. *https://seer.cancer.gov/*.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[Pakdaman et al., 2014] Pakdaman, M., Batal, I., Liu, Z., Hong, C., and Hauskrecht, M. (2014). An optimization-based framework to learn conditional random fields for multi-label classification. In *SDM*. SIAM.

[Patrini et al., 2014] Patrini, G., Nock, R., Rivera, P., and Caetano, T. (2014). (almost) no label no cry. In *Advances in Neural Information Processing Systems*, pages 190–198.

[Platt et al., 1999] Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

[Poulis and Dasgupta, 2017] Poulis, S. and Dasgupta, S. (2017). Learning with feature feedback: from theory to practice. In *Artificial Intelligence and Statistics*, pages 1104–1113.

[Qian et al., 2013] Qian, B., Wang, X., Wang, F., Li, H., Ye, J., and Davidson, I. (2013). Active learning from relative queries. In *IJCAI*. Citeseer.

[Quadrianto et al., 2009] Quadrianto, N., Smola, A. J., Caetano, T. S., and Le, Q. V. (2009). Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(Oct):2349–2374.

[Quinlan, 2014] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[Rahmani and Goldman, 2006] Rahmani, R. and Goldman, S. A. (2006). Missl: Multiple-instance semi-supervised learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 705–712. ACM.

[Rashidi and Cook, 2011] Rashidi, P. and Cook, D. J. (2011). Ask me better questions: active learning queries based on rule induction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 904–912. ACM.

[Ray and Craven, 2005] Ray, S. and Craven, M. (2005). Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd international conference on Machine learning*, pages 697–704. ACM.

[Roy and McCallum, 2001] Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

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

[Salmani and Sridharan, 2014] Salmani, K. and Sridharan, M. (2014). Multi-instance active learning with online labeling for object recognition. In *The Twenty-Seventh International Flairs Conference*, pages 406–411.

[Schultz and Joachims, 2004] Schultz, M. and Joachims, T. (2004). Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48.

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

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

[Settles et al., 2008] Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

[Seung et al., 1992] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

[Slivkins, 2018] Slivkins, A. (2018). Introduction to multi-armed bandits.

[Tschiatschek et al., 2014] Tschiatschek, S., Iyer, R. K., Wei, H., and Bilmes, J. A. (2014). Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421.

[Urner et al., 2013] Urner, R., Wulff, S., and Ben-David, S. (2013). Plal: Cluster-based active learning. In *Conference on Learning Theory*, pages 376–397.

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

[Valizadegan et al., 2013] Valizadegan, H., Nguyen, Q., and Hauskrecht, M. (2013). Learning classification models from multiple experts. *Journal of biomedical informatics*, 46(6):1125–1135.

[Van der Vaart, 2000] Van der Vaart, A. W. (2000). *Asymptotic statistics*, volume 3. Cambridge university press.

[Vanschoren et al., 2013] Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

[Vapnik, 1998] Vapnik, V. N. (1998). Statistical learning theory. *A Wiley-Interscience Publication*.

[Ward Jr, 1963] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

[Xing et al., 2003] Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. (2003). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528.

[Xue and Hauskrecht, 2017a] Xue, Y. and Hauskrecht, M. (2017a). Active learning of classification models with likert-scale feedback. In *SIAM Data Mining Conference, 2017*. SIAM.

[Xue and Hauskrecht, 2017b] Xue, Y. and Hauskrecht, M. (2017b). Efficient learning of classification models from soft-label information by binning and ranking. In *The Thirtieth International Flairs Conference*.

[Xue and Hauskrecht, 2018] Xue, Y. and Hauskrecht, M. (2018). Active learning of multiclass classifiers with auxiliary probabilistic information. In *The Thirty-First International Flairs Conference*.

[Xue and Hauskrecht, 2019] Xue, Y. and Hauskrecht, M. (2019). Active learning of multiclass classification models from ordered class sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5589–5596.

[Yang et al., 2013] Yang, C., Shen, J., Peng, J., and Fan, J. (2013). Image collection summarization via dictionary learning for sparse representation. *Pattern Recognition*, 46(3):948–961.

[Yu et al., 2013] Yu, F., Liu, D., Kumar, S., Tony, J., and Chang, S.-F. (2013). \proptosvm for learning with label proportions. In *ICML*, pages 504–512.

[Yu et al., 2014] Yu, F. X., Choromanski, K., Kumar, S., Jebara, T., and Chang, S.-F. (2014). On learning from label proportions. *arXiv preprint arXiv:1402.5902*.

[Zadeh, 1992] Zadeh, L. A. (1992). The calculus of fuzzy if-then rules. In *Fuzzy Engineering toward Human Friendly Systems—Proceedings of the International Fuzzy Engineering Symposium'91*, volume 1, pages 11–12.

[Zhu et al., 2003] Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3.