NEAR-OPTIMALITY AND ROBUSTNESS OF GREEDY ALGORITHMS FOR BAYESIAN POOL-BASED ACTIVE LEARNING

NGUYEN VIET CUONG

(B.Comp.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2015

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Nguyen Viet Cuong

18 May 2015

Acknowledgements

I would like to thank my advisor, Professor Lee Wee Sun, for his support and guidance throughout my study. The insights from my discussions with him and his suggestions have significantly improved my works. I also would like to thank Dr Chai Kian Ming Adam, Dr Chieu Hai Leong, and Dr Ye Nan for their helpful discussions about active learning and other related topics during our weekly meetings.

In addition, I would like to thank my collaborators: A/Professor Kan Min Yen, Dr Lu Wei, Dr Lam Si Tung Ho, Dr Vu Dinh, Dr Binh Thanh Nguyen, Duy Duc Nguyen, Doan Thanh Nam, Quang Pham, and Muthu Kumar Chandrasekaran, for their stimulating discussions and collaborations with me on various topics in machine learning as well as its applications in natural language processing and digital libraries. I also would like to thank Sumit Bhagwani for his help with the HOSemiCRF package, which is used in some experiments in this thesis.

Last but not least, I would like to thank my friends and my family for their patience and support throughout all these years.

List of Publications

The results in this thesis have been submitted or published in the following papers:

- Nguyen Viet Cuong, Wee Sun Lee, Nan Ye, Kian Ming A. Chai, Hai Leong Chieu. *Active Learning for Probabilistic Hypotheses Using the Maximum Gibbs Error Criterion*. In NIPS 2013.
- Nguyen Viet Cuong, Wee Sun Lee, Nan Ye. *Near-optimal Adaptive Pool-based Active Learning with General Loss*. In UAI 2014, best student paper award.
- Nguyen Viet Cuong, Nan Ye, Wee Sun Lee. *Robustness of Bayesian Pool-based Active Learning Against Prior Misspecification*. Under review.

The experimental results for active learning with conditional random fields in Chapter 6 of this thesis were based on the open-source HOSemiCRF package available at `https://github.com/nvcuong/HOSemiCRF`. The HOSemiCRF package was developed by myself and released with the following published papers:

- Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, Hai Leong Chieu. *Semi-Markov Conditional Random Field with High-Order Features*. In ICML 2011 Structured Sparsity: Learning and Inference Workshop.
- Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, Hai Leong Chieu. *Conditional Random Field with High-order Dependencies for Sequence Labeling and Segmentation*. In JMLR 2014.

During my study, I have published other papers whose contents were not included in this thesis. Those papers are:

- Nguyen Viet Cuong, Vu Dinh, Lam Si Tung Ho. *Mel-frequency Cepstral Coefficients for Eye Movement Identification*. In ICTAI 2012.

- Nguyen Viet Cuong, Lam Si Tung Ho, Vu Dinh. *Generalization and Robustness of Batched Weighted Average Algorithm with V-geometrically Ergodic Markov Data*. In ALT 2013.

- Quang H. Pham, Binh T. Nguyen, Nguyen Viet Cuong. *Punctuation Prediction for Vietnamese Texts Using Conditional Random Fields*. In ACML 2014 Workshop on Machine Learning and Its Applications in Vietnam.

- Vu Dinh, Lam Si Tung Ho, Nguyen Viet Cuong, Duy Duc Nguyen, Binh T. Nguyen. *Learning From Non-iid Data: Fast Rates for the One-vs-All Multiclass Plug-in Classifiers*. In TAMC 2015.

- Nguyen Viet Cuong, Muthu Kumar Chandrasekaran, Min-Yen Kan, Wee Sun Lee. *Scholarly Document Information Extraction using Extensible Features for Efficient Higher Order Semi-CRFs*. In JCDL 2015, short paper.

Contents

Declaration ii

Acknowledgements iii

List of Publications v

Abstract ix

List of Tables x

List of Figures xi

List of Algorithms xii

1 Introduction **1**

1.1 Motivations 1

1.2 Contributions 3

1.2.1 Theoretical Tools for Analyzing Active Learning Algorithms 4

1.2.2 Analyses of Maximum Entropy and Least Confidence Algorithms 5

1.2.3 The Maximum Gibbs Error Algorithm and Its Variants 6

1.2.4 Robustness of Bayesian Active Learning Algorithms 8

1.3 Thesis Outline 9

2 Background on Active Learning **11**

2.1 Settings for Active Learning 12

2.2 Approaches to Querying for Pool-based Active Learning 15

2.2.1 Uncertainty Sampling 15

2.2.2 Expected Error Minimization 17

2.2.3 Query-By-Committee 17

2.2.4 Other Approaches 19

2.3 Theoretical Analyses of Active Learning 19

3 Preliminaries **23**

3.1 Notations and Settings 24

3.2 Equivalence of Probabilistic Hypothesis Spaces and Deterministic Hypothesis Spaces 26

4 Submodular Function Maximization **30**

4.1 Near-optimality of Submodular Function Maximization 30

4.2 Near-optimality of Adaptive Submodular Function Maximization 31

4.3 Near-optimality of Pointwise Submodular Function Maximization 33

5 Properties of Maximum Entropy and Least Confidence Active Learning Algorithms **38**

5.1 The Maximum Entropy Criterion 38

5.2 The Least Confidence Criterion 43

6 The Maximum Gibbs Error Criterion **46**

6.1 Near-optimal Greedy Algorithms for Maximizing Policy Gibbs Error 48

6.1.1 The Non-adaptive Setting 48

6.1.2 The Adaptive Setting 49

6.1.3 The Batch Mode Setting 53

6.2 Computing maxGEC 56

6.2.1 MaxGEC for Bayesian Conditional Exponential Models 56

6.2.2 Batch maxGEC for Bayesian Transductive Naive Bayes 58

6.3 Experiments 60

6.3.1 Named Entity Recognition (NER) with Bayesian CRF 60

6.3.2 Text Classification with Bayesian Transductive Naive Bayes 62

7 The Generalized Maximum Gibbs Error Criterion **65**

7.1 The Average-case Criterion 66

7.2 The Worst-case Criterion 69

7.3 Computing the Criteria 72

7.4 Experiments 75

7.4.1 Experiments with Weighted Error Types Loss 77

7.4.2 Experiments with F-scores 78

7.4.3 Experiments with Weighted Test Examples Loss 79

8 Robustness of Bayesian Pool-based Active Learning 81

8.1 Robustness of Active Learning Algorithms 81

8.1.1 The Average Case 82

8.1.2 The Worst Case 87

8.1.3 Remarks 92

8.2 Robust Active Learning Using Mixture Prior 92

8.3 Experiments 94

8.3.1 Experiments With Different Priors 95

8.3.2 Experiments With Mixture Prior 96

9 Conclusion and Future Works 98

9.1 Conclusion 98

9.2 Future Works 99

References 102

Abstract

We study pool-based active learning in the Bayesian setting. To facilitate the analyses of active learning algorithms in this setting, we develop two powerful theoretical tools: (1) an equivalence between probabilistic hypothesis spaces and deterministic hypothesis spaces, and (2) a near-optimality guarantee for greedy algorithms when maximizing pointwise monotone submodular functions. Using these tools, we analyze and prove novel theoretical properties of two commonly used greedy algorithms for active learning: the maximum entropy and the least confidence algorithms. Then we propose a new greedy criterion called the maximum Gibbs error criterion, which can be proven to have near-optimality guarantees in the average case. The criterion can be approximated more easily even for complex structured models like the Bayesian conditional random fields, and it can be shown to perform well in practice. We also generalize the maximum Gibbs error criterion to include general loss functions into the criteria. We prove near-optimality guarantees for these new criteria and show that they also perform well in our experiments. Finally, we analyze the robustness of active learning algorithms against prior misspecification in both the average case and the worst case. We propose the use of mixture prior for more robust active learning and show in our experiments that it can achieve good performance even when the correct prior is unknown.

List of Tables

1.1 Theoretical properties of greedy criteria for adaptive active learning. 4

6.1 AUCs (%) of different learning algorithms with batch size $s = 10$. 64

6.2 AUCs (%) of different learning algorithms with batch size $s = 20$. 64

6.3 AUCs (%) of different learning algorithms with batch size $s = 30$. 64

7.1 AUCs (%) for loss with weighted error types on the 20 Newsgroups data set. 77

7.2 AUCs (%) for loss with weighted error types on UCI data sets. 77

7.3 AUCs (%) for F_2 and $F_{0.5}$ on the 20 Newsgroups data set. 78

7.4 AUCs (%) for F_2 and $F_{0.5}$ on UCI data sets. 78

7.5 AUCs (%) for loss with weighted test examples on the 20 Newsgroups data set. 80

7.6 AUCs (%) for loss with weighted test examples on UCI data sets. 80

8.1 AUCs of the maximum Gibbs error algorithm with $1/\sigma^2 =$ 0.01, 0.1, 1, 10 and the mixture prior model on the 20 Newsgroups data set. 97

8.2 AUCs of the maximum Gibbs error algorithm with $1/\sigma^2 =$ 0.01, 0.1, 1, 10 and the mixture prior model on the UCI data set. 97

List of Figures

2.1 The active learning process. 12

3.1 Examples of a non-adaptive policy tree (left) and an adaptive policy tree (right). 25

3.2 An example of a path ρ in an adaptive policy tree. In this example, the probability of ρ is $p_0^\pi[\rho] = p_0[(y_1 = 1, y_2 = 1); (x_1, x_2)]$. 26

6.1 Absolute AUC scores under the F_1 curves on the CoNLL 2003 data set. 62

8.1 The average AUC scores for passive learning and the maximum Gibbs error active learning algorithm with $1/\sigma^2 =$ 0.01, 0.1, 0.2, 1, and 10 on the 20 Newsgroups data set (a) and the UCI data set (b). 95

List of Algorithms

6.1 Batch maxGEC for Bayesian Batch Mode Active Learning 54

6.2 Approximation for Equation (6.5) in Bayesian transductive Naive Bayes model. 59

8.1 Active learning for the mixture prior model. 93

CHAPTER 1

Introduction

1.1 Motivations

A popular framework in supervised machine learning is passive learning, where a large amount of training data are randomly gathered and labeled by human annotators and then passed to a learning algorithm to train a model (e.g., a classifier). A good model can usually be obtained when there are large enough labeled training data. However, in practice, there are many cases where labeled training data are expensive to obtain (for example, text, audio, or visual data), and thus we need frameworks that enable learning algorithms to learn a good model with as few labeled training data as possible.

Active learning is one such framework in which learning algorithms are allowed to actively choose examples to learn (Angluin, 1988; Atlas et al., 1990; Lewis and Gale, 1994). This is an important framework as it helps to significantly reduce the number of labeled examples required for learning algorithms to train a good model. There are many different settings for active learning such as the membership query setting (Angluin, 1988), the stream-based setting (Atlas et al., 1990; Cohn et al., 1994), and the pool-based setting (Lewis and Gale, 1994). Among these settings, the pool-based setting where training examples are selected from a finite pool of unlabeled examples is very common in practice due to the availability of large amounts of unlabeled data for many real-world problems (Lewis and Gale, 1994; McCallum and Nigam, 1998; Tong and Chang, 2001; Hoi et al., 2006a). Various algorithms have been proposed and applied successfully for pool-based active learning. Perhaps the most famous among them is the family of uncertainty sampling algorithms which query the examples whose label is the least certain (Lewis and Gale, 1994; Culotta and McCallum, 2005; Settles and Craven, 2008). Although these algorithms perform well in practice, they still lack most of the theoretical understandings and guarantees, except in some very limited settings where the underlying model is assumed to be noiseless (Golovin and Krause, 2011).

The lack of theoretical properties for uncertainty sampling algorithms motivates us to study them in a more general and noisy setting. This setting is particularly useful in practice since real labels tend to be noisy (Angluin and Laird, 1988; Cuong et al., 2013a). Furthermore, considering the noisy setting allows us to develop new active learning algorithms that can be used with many readily available probabilistic models such as naive Bayes, logistic regression, conditional random field, etc. while at the same time enjoying the theoretical guarantees that come with these algorithms.

On our trek to investigating active learning algorithms in the noisy setting, various new active learning algorithms have been developed to fit different purposes. The maximum Gibbs error algorithm allows queries to be made based on the probabilities of all labels instead of just one or two most dominant labels as in the least confidence algorithm (Culotta and McCallum, 2005; Settles and Craven, 2008) or the margin sampling algorithm (Scheffer et al., 2001). The maximum Gibbs error algorithm also allows full Bayesian inference to be used to approximate the uncertainty criterion in large, complex structured models such as conditional random fields. Using full Bayesian inference to compute other active learning criteria such as maximum entropy or least confidence is usually not easy in such models due to the exponentially large number of structured labels. Furthermore, to combine active learning with the power of loss functions, we develop the generalized maximum Gibbs error algorithms that not only allow users to incorporate a general loss function into the active learning criterion but also maintain good theoretical guarantees. The use of loss functions is important in machine learning (Gneiting and Raftery, 2007; Masnadi-Shirazi and Vasconcelos, 2009) and is thus important for active learning as well.

For active learning algorithms in the Bayesian setting such as the maximum Gibbs error algorithm and its variants, the choice of prior is very important as it determines the uncertainty of each example and consequently the examples that will be queried. However, in real-world applications, the true prior that generates the data is usually unknown, and we have to use a prior close to the true prior in order to achieve a good performance. An analysis for such scenario, where a perturbed prior is used instead of the true prior, is still lacking for active learning. This motivates us to study the robustness of active learning algorithms under prior misspecification. Such study is useful as it helps us gain a deeper understanding of these algorithms and thus choose the robust ones to use. In addition, studying the robustness of active learning algorithms also motivates the use of mixture prior models for active learning that can achieve good performance when the true prior is unknown.

1.2 Contributions

This thesis makes the following four contributions to the study of active learning:

1. It develops two general and powerful tools for analyzing the theoretical properties of active learning algorithms in the noisy setting.
2. It applies the tools above to prove novel theoretical properties of two well-known active learning algorithms: the maximum entropy algorithm and the least confidence algorithm.
3. It develops a family of novel active learning algorithms called the maximum Gibbs error algorithms that are useful in practice and have good theoretical properties.
4. It analyzes the robustness of the maximum Gibbs error algorithms and the least confidence algorithm against prior misspecification. Then it proposes the use of a mixture prior model for more robust active learning.

Criterion	Objective	Near-optimality	Property	Robustness
Maximum entropy	Policy entropy	No constant factor approximation		
Least confidence	Worst-case version space reduction	$(1 - 1/e)$ factor approximation	Pointwise monotone submodular	Robust
Maximum Gibbs error	Policy Gibbs error (expected version space reduction)	$(1 - 1/e)$ factor approximation	Adaptive monotone submodular	Robust
Average generalized Gibbs error	Generalized policy Gibbs error (expected generalized version space reduction)	Loss-dependent		
Worst-case generalized Gibbs error	Total generalized version space reduction	$(1 - 1/e)$ factor approximation	Pointwise monotone submodular	Robust

Table 1.1: Theoretical properties of greedy criteria for adaptive active learning.

We briefly explain these contributions below. In Table 1.1, we summarize the theoretical findings in this thesis.

1.2.1 Theoretical Tools for Analyzing Active Learning Algorithms

We develop two general and powerful tools for analyzing theoretical properties of active learning algorithms in the noisy setting. The first tool is an equivalence result between the probabilistic model and the deterministic model that allows algorithms in the noisy setting to be analyzed in the noiseless setting. More specifically, to deal with probabilistic hypotheses, we construct an equivalent deterministic hypothesis space which contains all the possible labelings of the pool. Then a new prior on the deterministic hypothesis space is derived from the original prior such that the probability of any event is always the same with respect to both priors. Using this new model, we can prove the theoretical properties of various Bayesian pool-based active learning algorithms.

The second tool is a general theoretical guarantee for a greedy algorithm when maximizing pointwise monotone submodular functions. In particular, we prove that if a utility function satisfies *pointwise monotonicity*, *pointwise submodularity* and *minimal dependency*, then a simple greedy algorithm can achieve a constant factor approximation to the worst-case utility. This result is very useful for deriving greedy algorithms to maximize various utility functions with a theoretical guarantee.

1.2.2 Analyses of Maximum Entropy and Least Confidence Algorithms

The maximum entropy and the least confidence algorithms are two well-known uncertainty sampling criteria for pool-based active learning. The *maximum entropy criterion* greedily selects the example with maximum label entropy given the observed labels (Settles, 2010). In the non-adaptive case, where examples are selected before any label is observed, this criterion selects the example that maximally increases the label entropy of the selected set. The greedy criterion in this non-adaptive case is well-known to be near-optimal due to the submodularity of the entropy function: the label entropy of the selected examples is at least $(1 - 1/e)$ of the optimal set. Selecting a set with large label entropy is desirable, as the chain rule of entropy implies that maximizing the label entropy of the selected set will minimize the conditional label entropy of the remaining examples. It would be desirable to have a similar near-optimal performance guarantee for the adaptive case where the label is provided after every example is selected.

Thus, we formulate a similar objective for the maximum entropy criterion in the adaptive case which is called the *policy entropy*. Policy entropy is a generalization of Shannon entropy to general (possibly adaptive) policies. For non-adaptive policies, it reduces to Shannon entropy for sets. We prove that maximizing policy entropy is desirable since that will minimize the posterior label entropy of the remaining unlabeled examples. However, we also show that the maximum entropy algorithm does not provide a constant factor approximation to the optimal policy entropy in the adaptive case.

The *least confidence criterion* is another commonly used greedy criterion. This criterion selects the example whose most likely label has the smallest probability (Lewis and Gale, 1994; Culotta and McCallum, 2005). In this thesis, we show that this criterion provides a near-optimal adaptive algorithm for maximizing the worst-case version space reduction, where the version space is the probability of labelings that are consistent with the observed labels. This will be derived using the developed result above for pointwise monotone submodular functions.

1.2.3 The Maximum Gibbs Error Algorithm and Its Variants

We investigate an alternative objective function suitable for active learning called the *policy Gibbs error*. This is the expected error rate of a Gibbs classifier[1] on the set adaptively selected by the policy. It is a lower bound of the policy entropy; thus, by maximizing policy Gibbs error, we hope to maximize the policy entropy, whose maximality implies the minimality of the posterior label entropy of the remaining unlabeled examples in the pool. Besides, by maximizing policy Gibbs error, we also aim to obtain a small expected error of a posterior Gibbs classifier.[2] Small expected error of the posterior Gibbs classifier is desirable as it upper bounds the Bayes error but is at most twice of it.

Maximizing policy Gibbs error is hard, and we propose a greedy criterion, the *maximum Gibbs error criterion* (maxGEC), to solve it. This criterion queries the candidate that has maximum Gibbs error, the probability that a randomly sampled labeling does not match the actual labeling. We investigate the criterion in three settings: non-adaptive, adaptive, and batch mode settings (Hoi et al., 2006b). In all these settings, we prove that maxGEC is near-optimal compared to the best policy in the setting. We then examine how to compute maxGEC, particularly for large structured probabilistic models such as the conditional random fields (Lafferty et al., 2001). When inference in these models can be done efficiently, we show how to compute an approximation to the Gibbs error by sampling and efficient inference. We also provide an approximation for maxGEC in the non-adaptive and batch mode settings with the Bayesian transductive Naive Bayes

[1] A Gibbs classifier samples a hypothesis from the prior for labeling.

[2] A posterior Gibbs classifier samples a hypothesis from the posterior (instead of the prior) for labeling.

model. Experiments with maxGEC on named entity recognition and text classification tasks show its good performance in terms of the area under the curve.

MaxGEC can be seen as a greedy algorithm for sequentially maximizing the Gibbs error over the dataset. The Gibbs error of the dataset is the expected error of a Gibbs classifier that predicts using an entire labeling sampled from the prior label distribution of the whole dataset. Here, a labeling is considered incorrect if any example is incorrectly labeled by the Gibbs classifier. Predicting an entirely correct labeling of all examples is often unrealistic in practice, particularly after only a few examples are labeled. This motivates us to generalize the Gibbs error to handle different loss functions between labelings, e.g., Hamming loss which measures the Hamming distance between two labelings. We call the greedy criterion that uses general loss functions the *average generalized Gibbs error* criterion.

The corresponding performance measure for the average generalized Gibbs error criterion is the generalized policy Gibbs error, which is the expected value of the generalized version space reduction. The generalized version space reduction function is an extension of the version space reduction function with a general loss. We investigate whether this new objective is adaptive submodular, as this property would provide a constant factor approximation for the average generalized Gibbs error criterion. Unfortunately, we can show that this function is not necessarily adaptive submodular, although it is adaptive submodular for the special case of the version space reduction. Despite that, our experiments show that the average generalized Gibbs error criterion can perform reasonably well in practice, even when we do not know whether the corresponding utility function is adaptive submodular.

We also consider a worst-case setting for the generalized Gibbs error. The worst case against a target labeling can be severe, so we consider a variant: the total generalized version space reduction. This function targets the sum of the remaining losses over all the remaining labelings, rather than against a single worst-case labeling. We call the corresponding greedy criterion the *worst-case generalized Gibbs error* criterion. It selects the example with maximum worst-case total generalized version space reduction as the next query. We show that the total generalized version space reduction function is pointwise monotone submodular and satisfies the minimal dependency property; thus, the method is guaranteed to be near-optimal. Our experiments show that the worst-case generalized Gibbs error criterion performs well in practice. For binary problems, the maximum entropy, least confidence, and Gibbs error criteria are all equivalent, and the worst-case generalized Gibbs error criterion outperforms them for most problems in our experiments.

1.2.4 Robustness of Bayesian Active Learning Algorithms

Bayesian pool-based active learning assumes the labeling of data is generated from a prior distribution, which is generally assumed to be known in theory (Golovin and Krause, 2011; Chen and Krause, 2013). In practice, the prior is often unknown, and we choose a prior that is considered to be close to the true prior. However, to the best of our knowledge, there is no analysis on the effect of a perturbed prior on the performance of active learning algorithms. Thus, we investigate the *robustness* of active learning algorithms against prior misspecification – that is, whether an algorithm achieves similar performance using a perturbed prior as compared to using a true prior.

Our main result is that if the utility function is continuous in the prior, for both the average case and the worst case, an α-approximate algorithm is robust; that is, when using a perturbed prior, the algorithm is near α-approximate. In particular, if the utility function satisfies a Lipschitz continuity condition in the prior, then the performance guarantee on the expected utility or the worst-case utility degrades by at most a constant multiple of the ℓ_1 distance between the perturbed prior and the true prior. Combining with the $(1 - 1/e)$-approximation results for the maximum Gibbs error algorithm and the least confidence algorithm, it follows that they are near $(1 - 1/e)$-approximate. In addition, if an algorithm achieves the optimal approximation ratio using the true prior, it is a near-optimal approximation algorithm using a perturbed prior.

In practical usage of the active learning algorithms in the Bayesian setting, it is usually difficult to determine a good prior to use. However, it is often reasonable to select a set of priors with the expectation that one member of the set would be close to a good prior. For example, in passive supervised learning, it is a common practice to use a validation set to select a regularization parameter from among a small set of reasonable parameters in methods such as regularized logistic regression. Given a finite set of candidate priors, our theoretical analysis suggests that we should try to design a prior that is not too far from any of the priors in the set. One simple prior that is not too far away from a set of priors is a uniform mixture of the priors in the set. We experimented with using the uniform mixture prior for active learning on some UCI data sets and a text classification data set. The experiments show that the mixture prior is indeed robust and gives performance that is reasonably close to the best prior in the set.

1.3 Thesis Outline

The rest of this thesis are structured as follows.

- Chapter 2 reviews some backgrounds on active learning.
- Chapter 3 gives the notations and settings used in the thesis, then it discusses the equivalence between the probabilistic model and the deterministic model. This equivalence result was published in (Cuong et al., 2013b).
- Chapter 4 reviews some previous results on submodular and adaptive submodular function maximization, and then proves the new result on pointwise submodular function maximization. This new result was published in (Cuong et al., 2014a).
- Chapter 5 analyzes the theoretical properties of the maximum entropy criterion and the least confidence criterion. The results in this chapter were published in (Cuong et al., 2014a).
- Chapter 6 introduces the maximum Gibbs error criterion, analyzes its near-

optimality guarantees, and provides experimental results regarding the new criterion. The results in this chapter were published in (Cuong et al., 2013b).

- Chapter 7 discusses the generalizations of the maximum Gibbs error criterion with general loss functions and analyzes their near-optimality guarantees. The chapter also provides the experimental results for the new criteria. The results in this chapter were published in (Cuong et al., 2014a).
- Chapter 8 analyzes the robustness of Bayesian pool-based active learning algorithms and applies the results to the maximum Gibbs error and least confidence algorithms. The chapter also describes the mixture prior model for more robust active learning and provides experimental results for this model.
- Chapter 9 gives the conclusion and some directions for future works.

Chapter 2

Background on Active Learning

Machine learning is an interdisciplinary field that studies algorithms that can learn from past data and make predictions on new data. It has many useful applications such as in natural language processing (Manning and Schütze, 1999; Sebastiani, 2002), computer vision (Guo et al., 2000), biometrics (Huang et al., 2002; Cuong et al., 2012), and social network analysis (Al Hasan et al., 2006).

Active learning is a machine learning framework in which the learning algorithms are allowed to actively choose the examples to learn. More specifically, at each iteration, an active learning algorithm may choose an unlabeled example and query an oracle (a human annotator) for its label. After observing the label of the chosen example, the active learning algorithm can then retrain or update its model and then use the new model to select the next unlabeled example. The unlabeled examples given to the oracle may be generated by the active learning algorithm itself or may be chosen from a stream or a pool of data. A typical active learning process is shown in Figure 2.1.

The motivation for active learning is based on the assumption that unlabeled data are free or very inexpensive, while the cost of labeling them is expensive. This situation may often be encountered in real-world problems such as speech recognition (Zhu, 2005) and information extraction (Settles et al., 2008a). By allowing the algorithm to actively choose the data from which it learns, we hope that the active learning algorithm can achieve the same or better accuracy than usual passive learning algorithms while requiring a much lower cost for labeling the data. Thus, the freedom of choosing the

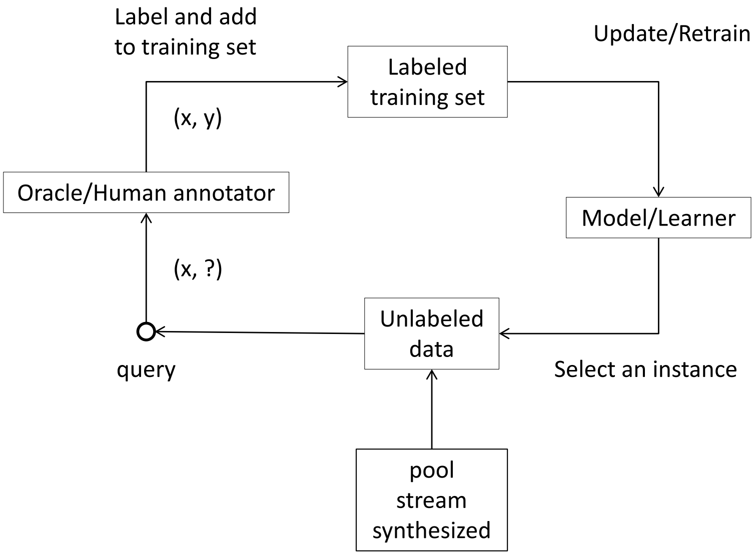

Figure 2.1: The active learning process.

data to learn is an important property of active learning since it helps reduce the number of labeled data needed to train a good model.

In the subsequent sections, we will introduce some main settings for active learning. Then we discuss various approaches for choosing training data in the pool-based setting, which is the focus of this thesis. Finally, we summarize some theoretical results on active learning.

2.1 Settings for Active Learning

In active learning, there are different settings in which an active learning algorithm may choose the unlabeled data and query the oracle for their labels. The three main settings that have been considered are the membership query setting (Angluin, 1988), the stream-based setting (Atlas et al., 1990; Cohn et al., 1994), and the pool-based setting (Lewis and Gale, 1994). Membership query is one of the first active learning settings to be investigated; however, there are some limitations with this setting. The stream-based and pool-based settings are proposed to overcome these limitations.

The Membership Query Setting: In this setting, the active learning algorithm can query the label of any data point in the input space, including those that the algorithm generates by itself. Although efficiently generating the queries is tractable for finite domains (Angluin, 2001), this setting may cause difficulties for the human annotator. The reason is that it is usually confusing for a human annotator to label arbitrary data instances, especially those generated by the active learning algorithms. For example, when generating handwritten characters for querying, the active learning algorithm may construct hybrid characters that have no semantic meaning to the human annotator (Baum and Lang, 1992). Despite this limitation, membership query still has some promising applications in a few real-world problems (Cohn et al., 1996; King et al., 2004, 2009).

The Stream-based Setting: In this setting, unlabeled data examples are drawn one by one from an underlying distribution. For each unlabeled example, the active learning algorithm has to decide immediately whether to query its label or not. If an example is queried, its label can be used to update the model before the next unlabeled example in the stream is considered. There are many strategies for deciding whether to query a label or not. For example, one can measure the amount of information in each data example and query the more informative examples (Dagan and Engelson, 1995). Another common approach is to compute the region of uncertainty (Cohn et al., 1994), which is a subset of the input space that is still uncertain to the active learning algorithm. If an unlabeled example is in this region, the algorithm will query its label. To compute the region of uncertainty, we need to define the version space, the set of hypotheses consistent with the current labeled training data. The region of uncertainty contains the data examples that cause disagreement among the hypotheses in the version space. Although the region of uncertainty is useful, computing it explicitly is usually intractable. Thus, we often approximate it in practice (Seung et al., 1992; Cohn et al., 1994; Dasgupta et al., 2007).

The Pool-based Setting: In this setting, we assume that the active learning algorithm is given a large finite pool of unlabeled data drawn from an underlying distribution. If a large enough pool is sampled from the true distribution, good performance of a model on the pool implies good generalization performance of the model. During the learning process, at each iteration, the active learning algorithm chooses an unlabeled example from the pool and queries the oracle for its label. The choice of the unlabeled example is usually based on some greedy criterion such as most informativeness or least confidence. The main difference between pool-based and stream-based settings is that in the pool-based setting, we can consider all the unlabeled examples in the pool and make the query, while in the stream-based setting, we can only consider the unlabeled examples one by one and make the decision locally. The pool-based setting has been used in many real-world applications of active learning such as information extraction (Thompson et al., 1999; Settles and Craven, 2008), information retrieval (Tong and Chang, 2001; Zhang and Chen, 2002), or speech recognition (Tur et al., 2005). Both stream-based and pool-based settings can overcome the limitation of membership query because the unlabeled data in the two former settings are drawn from the true data distribution, and thus rare data that may confuse the human annotators have a low probability to occur.

Other Variants: There are also other setting variants for active learning such as active learning for structured data or batch mode active learning. Many real-world applications can be modeled as a prediction task on structured data such as sequences or trees. For example, in a sequence labeling problem, the input sequence $\vec{x}$ has the form $\vec{x} = (x_1, x_2, \ldots, x_{|\vec{x}|})$ and the corresponding label sequence $\vec{y}$ is $(y_1, y_2, \ldots, y_{|\vec{x}|})$. In this case, the label sequence is usually predicted using sequence models such as hidden Markov models (HMMs) (Rabiner, 1989) or conditional random fields (Lafferty et al., 2001). Settles and Craven (2008) evaluated a number of active learning strategies for sequence labeling tasks using various probabilistic sequence models. These algorithms can be adapted to other probabilistic sequence models such as probabilistic context-free grammars (Baldridge and Osborne, 2004; Hwa, 2004) or HMMs (Dagan and Engelson, 1995; Scheffer et al., 2001). Active learning with structured data can be either in the stream-based setting or the pool-based setting. Batch mode active learning is a variant of pool-based active learning in which the learning algorithm may choose from the pool a fixed-size batch of unlabeled examples to query in each iteration (Hoi et al., 2006b). The labels of the whole batch will be used to update or retrain the model for the next iteration. Batch mode active learning has been applied to medical image classification (Hoi et al., 2006b), large-scale text categorization (Hoi et al., 2006a), and image retrieval (Hoi et al., 2009). Pool-based active learning is also related to adaptive informative path planning (Lim et al., 2015a) and adaptive stochastic optimization (Lim et al., 2015b) problems.

2.2 Approaches to Querying for Pool-based Active Learning

Since the focus of this thesis is pool-based active learning, in this section we discuss various approaches for choosing the unlabeled examples to query in this setting. We first describe the simplest and most intuitive approaches called uncertainty sampling. Some uncertainty sampling criteria will be analyzed in this thesis, and the new criteria proposed in this thesis also belong to this group. We then discuss two other main approaches called expected error minimization and query-by-committee. Finally, we will briefly describe some other approaches.

2.2.1 Uncertainty Sampling

The general idea of the uncertainty sampling approach (Lewis and Gale, 1994) is to choose the unlabeled data examples whose labels the current model is least certain about. For instance, when considering a binary classification problem with a probabilistic model, this approach is equivalent to querying the data example whose probability of being labeled 1 is nearest to 0.5 (Lewis and Catlett, 1994; Lewis and Gale, 1994). To extend the approach to multiclass classification problems with more labels, we may use the least confidence algorithm which queries the examples according to the following greedy criterion:

$$x^* = \arg\min_x \{\max_y p[y; x]\},$$

where $p[y; x]$ is the posterior probability that y is the label of x. In other words, the model is least certain about the most probable label of x^*. This algorithm can be used conveniently in sequence labeling or structured prediction problems since the most probable label sequence can be computed easily using the Viterbi algorithm (Culotta and McCallum, 2005; Settles and Craven, 2008).

Although the least confidence algorithm above is useful in some applications, it uses only the most probable label and discards the information in other labels. Thus, the margin sampling algorithm (Scheffer et al., 2001) was proposed to overcome this limitation. According to this algorithm, we query the example:

$$x^* = \arg\min_x \{p[y_1; x] - p[y_2; x]\},$$

where y_1 and y_2 are the most and second most probable labels of x respectively. In other words, the model is least certain when distinguishing between the best two labels of x^*. The margin sampling algorithm can also be used in sequence labeling problems because the best two label sequences of an input sequence can be computed easily using dynamic programming (Settles and Craven, 2008).

A more general algorithm for uncertainty sampling is the maximum entropy algorithm (Settles and Craven, 2008; Settles, 2010). This algorithm can make use of the probabilities of all the labels instead of just the best one or two labels as in the least confidence or margin sampling algorithm. More specifically, the maximum entropy algorithm queries the unlabeled example that maximizes the Shannon entropy (Shannon, 1948) of the label distribution. That is, it chooses the example:

$$x^* = \arg\max_x \left\{ -\sum_y p[y; x] \ln p[y; x] \right\},$$

where y ranges over all the possible labels of x. For binary classification problems, the maximum entropy algorithm reduces to the least confidence algorithm and the margin sampling algorithm. However, it can also be used in more complex problems such as problems with sequence data (Settles and Craven, 2008) or tree data (Hwa, 2004).

2.2.2 Expected Error Minimization

Another approach to query selection in pool-based active learning is to choose the examples that minimize the expected error of the model (Roy and McCallum, 2001). For example, if we consider 0-1 loss, this approach will query the example x^* such that:

$$x^* = \arg\min_x \sum_y p[y;x] \left(\sum_{x_u \in \mathcal{U}} (1 - p'[y_u; x_u]) \right),$$

where p is the current posterior probability, p' is the new posterior probability if (x, y) is added to the labeled training set, $\mathcal{U}$ is the set of all remaining unlabeled examples, and $y_u = \arg\max_y p'[y; x_u]$ is the most probable label of x_u with respect to p'. If log-loss is considered, then the example x^* chosen is:

$$x^* = \arg\min_x \sum_y p[y;x] \left(- \sum_{x_u \in \mathcal{U}} \sum_{y'} p'[y'; x_u] \log p'[y'; x_u] \right).$$

This expected error minimization approach has been used successfully with various types of models such as naive Bayes (Roy and McCallum, 2001), support vector machines (Moskovitch et al., 2007), Gaussian random fields (Zhu et al., 2003), and logistic regression (Guo and Greiner, 2007). However, computing the exact expected error is usually very expensive. Thus, in practice, we often approximate it using Monte Carlo sampling methods (Roy and McCallum, 2001).

2.2.3 Query-By-Committee

In the query-by-committee approach (Seung et al., 1992), the active learning algorithm maintains a set $\mathcal{C} = \{\theta_1, \theta_2, \ldots, \theta_C\}$ of different models trained on the current labeled training set. This set $\mathcal{C}$ is called a committee. For each unlabeled example, the members of the committee may vote for its label. Then, the example with the most disagreement among the committee members is queried. The idea of query-by-committee is very similar to minimizing the version space described in Section 2.1.

There are two main issues involving the use of query-by-committee. The first issue is how to construct a good committee of models, and the second issue is how to measure the disagreement among the committee members. The first issue can be addressed in a variety of ways. For example, Seung et al. (1992) randomly sampled two models that are consistent with the current labeled training set, while McCallum and Nigam (1998) sampled the committee members from a Dirichlet distribution over the model parameters. The committee members can also be sampled from a normal distribution as in (Dagan and Engelson, 1995).

For the second issue, there are two common measures for the disagreement among the committee members: the Kullback-Leibler (KL) divergence (McCallum and Nigam, 1998) and the vote entropy (Dagan and Engelson, 1995). If KL divergence is used, the example x^* chosen is:

$$x^* = \arg\max_x \frac{1}{C} \sum_{c=1}^{C} D(p_{\theta_c} || p_{\mathcal{C}}),$$

where $D(p_{\theta_c} || p_{\mathcal{C}}) = \sum_y p_{\theta_c}[y;x] \log \frac{p_{\theta_c}[y;x]}{p_{\mathcal{C}}[y;x]}$ and $p_{\mathcal{C}}[y;x] = \frac{1}{C} \sum_{c=1}^{C} p_{\theta_c}[y;x]$.

If vote entropy is used, the example x^* chosen is:

$$x^* = \arg\max_x \left\{ - \sum_y \frac{V(y)}{C} \log \frac{V(y)}{C} \right\},$$

where y ranges over all the possible labels and $V(y)$ is the number of votes that y receives from the committee.

2.2.4 Other Approaches

There are also other approaches for selecting the examples in pool-based active learning. One approach to querying is to select examples that would make the most changes to the current model. This approach measures the expected gradient length of the objective function over the labeled training set (Settles and Craven, 2008; Settles et al., 2008b). In this approach, we define $\bigtriangledown l_\theta(\mathcal{L})$ to be the gradient of the objective function l with respect to the current model parameters θ, where $\mathcal{L}$ is the current labeled training set. Then the example x^* that is queried would satisfy the equation:

$$x^* = \arg\max_x \sum_y p_\theta[y; x] \left\| \bigtriangledown l_\theta(\mathcal{L} \cup \{(x, y)\}) \right\| .$$

Another approach is to use the information density of the unlabeled data examples (Settles, 2008, 2010; Settles and Craven, 2008). In this approach, we define the informative examples to be those that we are uncertain about or those that are in the dense regions of the input space. Then the example x^* queried is:

$$x^* = \arg\max_x \phi(x) \left(\frac{1}{|\mathcal{U}|} \sum_{x_u \in \mathcal{U}} \text{sim}(x, x_u) \right)^\beta ,$$

where $\phi(x)$ is the informativeness of x, $\text{sim}(\cdot, \cdot)$ is the similarity function, and β is a parameter controlling the importance of the density term.

2.3 Theoretical Analyses of Active Learning

In this section, we summarize some theoretical results on active learning. Although empirical analyses have shown the usefulness of active learning (Settles and Craven, 2008; Tomanek and Olsson, 2009), the theoretical analyses of active learning algorithms are not yet completed, especially for the pool-based setting. There are some positive results on the effect of active learning; however, most of them are based on intractable or complex algorithms that are difficult to be used in practice. Furthermore, many theoretical results are limited to minimizing 0-1 loss, and they cannot be easily adapted to other loss functions.

For the membership query setting, there have been some strong positive results by Angluin (1988, 2001). However, as discussed in Section 2.1, this setting is very difficult for human annotators to label the queried examples. Besides, most real-world applications assume that unlabeled data from the actual distribution are freely available. So, these theoretical results do not have much practical impact on real applications.

For the stream-based and pool-based settings, we have an early strong theoretical analysis of the query-by-committee algorithm (Freund et al., 1997). Under the Bayesian assumption, the algorithm can achieve generalization error ϵ after seeing $O(d/\epsilon)$ unlabeled examples and querying $O(d \log 1/\epsilon)$ labels, where d is the Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis, 1971) of the hypothesis space. This is an exponential improvement over normal supervised learning, whose sample complexity is $O(d/\epsilon)$. However, the query-by-committee algorithm may be computationally expensive in some practical problems.

For the stream-based setting, Dasgupta et al. (2005) proposed an active learning version of the perceptron algorithm together with a variant of the perceptron update rule which can achieve exponential improvement on the sample complexity compared to the usual passive learning. Asymptotically, some active learning algorithms can always perform better than passive learning in the limit (Balcan et al., 2010). Many results for active learning assumed data are separable (or noiseless), i.e., there exists a hypothesis that can classify the examples perfectly. Such an assumption is usually not true in practice. Thus, there are works that consider the non-separable case (also called the agnostic or noisy case) where the perfect classifier does not exist. In the non-separable setting, Hanneke (2007) proved an upper bound on the sample complexity of active learning. Dasgupta et al. (2007) also proposed a polynomial-time reduction algorithm from active learning to supervised learning for an arbitrary input distribution and hypothesis space for this agnostic case. In a more recent work, Agarwal (2013) proved the fast learning rates of active learning algorithms for the cost-sensitive multiclass classification problem under Tsybakov's margin assumption (Tsybakov, 2004), a form of low noise assumption. Fast learning rates can also be achieved by some passive learning algorithms under the same margin assumption (Audibert and Tsybakov, 2007; Dinh et al., 2015).

A notable family of active learning algorithms in the agnostic stream-based setting is the importance weighted active learning algorithms (Beygelzimer et al., 2009, 2010). Since the training data selected from active learning are usually biased, these algorithms consider a weighted average of the losses such that the resulting weighted loss is unbiased. Beygelzimer et al. (2009) described a constrained version of importance weighted active learning for general loss functions, while Beygelzimer et al. (2010) proposed an unconstrained version of the algorithm for the 0-1 loss. These algorithms were shown to have better label complexity than passive learning when the disagreement coefficient is bounded.

For pool-based active learning, greedy algorithms are the simplest and most common (Settles, 2010). However, they usually do not have any theoretical guarantee except for the non-adaptive case or the case where data are noiseless. Under the noiseless assumption, Dasgupta (2004) proved that the sample complexity of linear classifiers is $O(1/\epsilon)$ in the worst case, which is the same as passive learning. He also provided some upper and lower bounds for active learning in the pool-based setting. In the noiseless Bayesian setting, an algorithm called generalized binary search (Nowak, 2008, 2011) was proven to be near-optimal: its expected number of queries is within a factor of $(\ln \frac{1}{\min_h p_0[h]} + 1)$ of the optimum, where p_0 is the prior (Golovin and Krause, 2011). This result was obtained using the adaptive submodularity of the version space reduction. Adaptive submodularity is an adaptive version of submodularity, a natural diminishing returns property. The adaptive submodularity of version space reduction was also applied to the batch mode setting to prove the near-optimality of a batch greedy algorithm that maximizes the average version space reduction for each selected batch (Chen and

Krause, 2013). The algorithms that we propose in Chapter 6 of this thesis can be seen as generalizations of these version space reduction algorithms to the noisy setting. When the hypotheses are deterministic, our algorithms are equivalent to these version space reduction algorithms.

For the case of noisy data, a noisy version of the generalized binary search was proposed (Nowak, 2009). This algorithm was proven to be optimal under the neighborly condition, a very limited setting where “each hypothesis is locally distinguishable from all others” (Nowak, 2009). In another work, Bayesian active learning was modeled by the Equivalence Class Determination problem and a greedy algorithm called EC^2 was proposed for this problem (Golovin et al., 2010). Although the cost of EC^2 is provably near-optimal, this formulation requires an explicit noise model and the near-optimality bound is only useful when the support of the noise model is small. Our algorithms in this thesis, in contrast, are simpler and do not require an explicit noise model: the noise model is implicit in the probabilistic model and our algorithms are only limited by computational concerns. Aside from the near-optimality analysis considered in this thesis, other types of bounds were also obtained for pool-based active learning using minimax analysis (Castro et al., 2005; Castro and Nowak, 2006, 2008).

Chapter 3

Preliminaries

In this thesis, we consider the Bayesian setting and study pool-based active learning (McCallum and Nigam, 1998) where training data are sequentially selected from a finite set (called a pool) of unlabeled examples, with the aim of having good performance after only a small number of examples are labeled. If a large enough pool is sampled from the true distribution, good performance of a classifier on the pool implies good generalization performance of the classifier. Previous theoretical works on Bayesian active learning mainly deal with the noiseless case, which assumes a prior distribution on a collection of deterministic mappings from observations to labels (Golovin and Krause, 2011; Chen and Krause, 2013). A fixed deterministic mapping is then drawn from the prior, and it is used to label the examples.

In contrast to these works, we consider the case where probabilistic hypotheses, rather than deterministic ones, are used to label the examples. We formulate the objective for pool-based active learning as a maximum coverage objective with a fixed budget: *given a budget of k queries, we aim to adaptively select from the pool the best k examples with respect to some appropriate objective function.*[1] In practice, the selection of the next example to be labeled is usually done by greedy optimization of the objective function.

This chapter formally introduces the notations and settings for this scenario. Then it

[1] In our setting with probabilistic hypotheses, the usual objective of determining the true labeling of the pool is infeasible unless the number of allowed labelings is small. When the prior gives non-zero probabilities to all the possible labelings of the pool, we need to query the whole pool in order to determine its true labeling.

will present a result on the equivalence between probabilistic hypothesis spaces and deterministic hypothesis spaces. This result forms the basis for various arguments and proofs throughout the thesis.

3.1 Notations and Settings

Let $\mathcal{X}$ be a (possibly infinite) set of examples (or items), $\mathcal{Y}$ be a fixed finite set of labels (or states), and $\mathcal{H}$ be a set of probabilistic hypotheses. In this thesis, we assume $\mathcal{H}$ is finite, but our results extend readily to any general $\mathcal{H}$. For any probabilistic hypothesis $h \in \mathcal{H}$, its application to an example $x \in \mathcal{X}$ is a categorical random variable with support $\mathcal{Y}$, and we write $\mathbb{P}[h(x) = y \mid h]$ to denote the probability that $h(x)$ has value $y \in \mathcal{Y}$.

Let $X \subseteq \mathcal{X}$ be a finite set of examples from which the active learning algorithms will choose their training data. We usually call X a *pool* of examples. A *labeling* of X is a function from X to $\mathcal{Y}$, and a *partial labeling* is a partial function from X to $\mathcal{Y}$. For any $S \subseteq X$, we use the notation $h(S)$ to denote the label sequence $(h(x_1), h(x_2), \ldots, h(x_i))$ whenever S is a sequentially constructed set $(x_1, x_2, \ldots, x_i)$, or simply the set $\{h(x) : x \in S\}$ if the examples in S are not ordered. We also write $\mathbb{P}[h(S) = \mathbf{y} \mid h]$ for the probability that h assigns the label sequence $\mathbf{y}$ to examples in the sequence S.

We operate within the Bayesian setting and assume a prior probability $p_0[h]$ on $\mathcal{H}$. After observing a labeled set (i.e., a partial labeling) $\mathcal{D}$, we can obtain the posterior $p_{\mathcal{D}}[h] \overset{\text{def}}{=} p_0[h|\mathcal{D}]$ using Bayes' rule. For any distribution $p[h]$ on $\mathcal{H}$ and any example sequence $S \subseteq X$, we write $p[\mathbf{y}; S]$ to denote the probability that the example sequence S is assigned the label sequence $\mathbf{y}$ by a hypothesis drawn randomly from $p[h]$. Formally,

$$p[\mathbf{y}; S] \overset{\text{def}}{=} \sum_{h \in \mathcal{H}} p[h] \, \mathbb{P}[h(S) = \mathbf{y} \mid h].$$

Note that $p[\,\cdot\,; S]$ is a probability distribution on the set of all label sequences $\mathbf{y}$ of S.

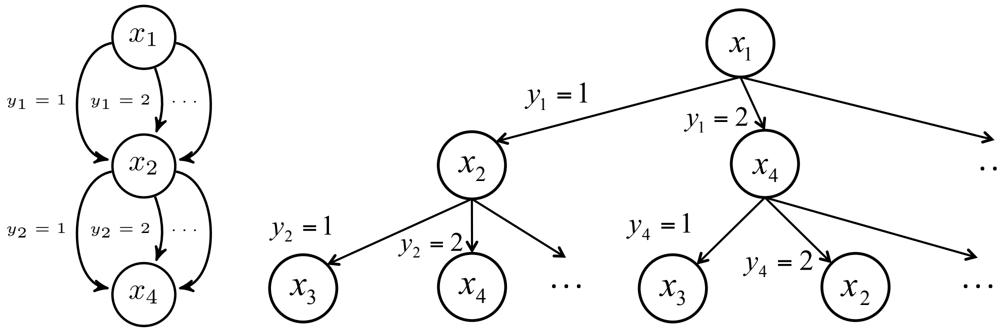

Figure 3.1: Examples of a non-adaptive policy tree (left) and an adaptive policy tree (right).

When S is a singleton $\{x\}$, we write $p[y; x]$ for $p[\{y\}; \{x\}]$.

A pool-based active learning algorithm corresponds to a policy for choosing training examples from the pool X. A *policy* is a mapping from a partial labeling to the next unlabeled example to query. At the beginning, a fixed label sequence $\mathbf{y}^*$ of X is given by a hypothesis h drawn from the prior $p_0[h]$ and is hidden from the learner. Equivalently, $\mathbf{y}^*$ can be thought of as being drawn from the prior distribution $p_0[\mathbf{y}^*; X]$ over the label sequences of X. During the learning process, each time the active learning policy selects an unlabeled example, its label according to $\mathbf{y}^*$ will be revealed to the learner.

A policy can be represented by a policy tree, where a node represents the next example to be queried, and each edge below the node corresponds to a possible label. In this thesis, we use the terms policy and policy tree interchangeably. Figure 3.1 illustrates two policy trees with their top three levels: in the non-adaptive setting, the policy ignores the labels of the previously selected examples, so all examples at the same depth of the policy tree are the same; in the adaptive setting, the policy takes into account the observed labels when choosing the next example.

A full policy tree for the pool X is a policy tree of height $|X|$. A partial policy tree is a subtree of a full policy tree with the same root. The class of policies of height k will be denoted by Π_k. Note that $\Pi_{|X|}$ contains full policy trees, while Π_k with $k < |X|$ contains partial policy trees.

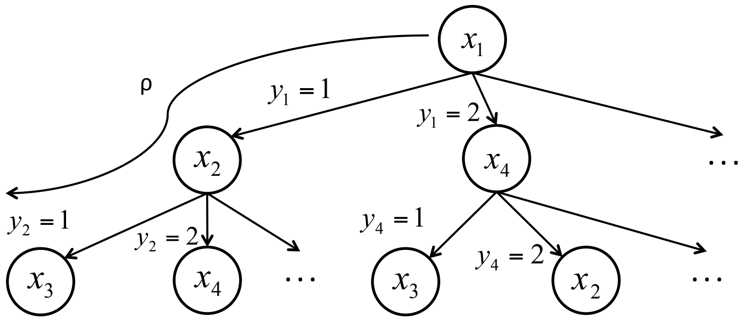

Figure 3.2: An example of a path ρ in an adaptive policy tree. In this example, the probability of ρ is $p_0^\pi[\rho] = p_0[(y_1 = 1, y_2 = 1); (x_1, x_2)]$.

For any (full or partial) policy tree π and any prior p_0, we define a probability distribution $p_0^\pi[\,\cdot\,]$ over all possible paths from the root to the leaves of π. This distribution over paths is induced from the uncertainty in the fixed label sequence $\mathbf{y}^*$ of X: since $\mathbf{y}^*$ is drawn randomly from $p_0[\mathbf{y}^*; X]$, the path ρ followed from the root to a leaf of the policy tree during the execution of π is also a random variable. If x_ρ (respectively, y_ρ) is the sequence of examples (respectively, labels) along path ρ, then the probability of ρ is $p_0^\pi[\rho] \overset{\text{def}}{=} p_0[y_\rho; x_\rho]$. Figure 3.2 gives an illustration of a path and its probability in an adaptive policy tree. In this thesis, some objective functions for pool-based active learning can be defined using this probability distribution over paths.

3.2 Equivalence of Probabilistic Hypothesis Spaces and Deterministic Hypothesis Spaces

In this section, we establish a result on the equivalence between the probabilistic model in Section 3.1 and a deterministic model. Recall that any $h \in \mathcal{H}$ is a probabilistic hypothesis and $\mathbb{P}[h(x) = y \,|\, h] \in [0, 1]$. For any partial labeling $\mathcal{D}$, let $x_\mathcal{D} \overset{\text{def}}{=} \text{dom}(\mathcal{D})$ be the domain of $\mathcal{D}$, and let $y_\mathcal{D} \overset{\text{def}}{=} \mathcal{D}(x_\mathcal{D})$ be the label sequence assigned to $x_\mathcal{D}$ by the partial labeling $\mathcal{D}$.

Let p_0 be a prior on $\mathcal{H}$. After a partial labeling $\mathcal{D}$ is observed, the posterior $p_\mathcal{D}$ can be obtained using Bayes' rule:

$$p_{\mathcal{D}}[h] = p_0[h|\mathcal{D}] = \frac{p_0[h]\,\mathbb{P}[h(x_{\mathcal{D}}) = y_{\mathcal{D}} \mid h]}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]}.$$

From this noisy model with probabilistic hypotheses, we construct an equivalent noiseless model with deterministic hypotheses as follows.

Consider a new hypothesis space $\overline{\mathcal{H}}$ such that $\overline{\mathcal{H}} \stackrel{\text{def}}{=} \{\bar{h} : \bar{h} \in \mathcal{Y}^X\}$, where for any $S \subseteq X$, we write $\mathcal{Y}^S$ to denote the set of all partial labelings with domain S. Thus, $\overline{\mathcal{H}} = \mathcal{Y}^X$ is the set of all labelings of X. For any $\bar{h} \in \overline{\mathcal{H}}$ and $x \in X$, $\bar{h}(x)$ is the label of x according to the labeling $\bar{h}$. Hence, each $\bar{h} \in \overline{\mathcal{H}}$ is a deterministic hypothesis. For a sequence of examples $S \subseteq X$, we write $\bar{h}(S)$ to denote the label sequence of S according to the labeling $\bar{h}$.

Furthermore, we construct a new prior $\bar{p}_0$ over $\overline{\mathcal{H}}$ such that:

$$\bar{p}_0[\bar{h}] \stackrel{\text{def}}{=} p_0[\bar{h}(X); X] = \sum_{h \in \mathcal{H}} p_0[h]\,\mathbb{P}[h(X) = \bar{h}(X) \mid h].$$

In the above formula, $\bar{p}_0[\bar{h}]$ is the probability in the probabilistic model that the label sequence of X is $\bar{h}(X)$. In essence, we have "moved" the uncertainty associated with the likelihood $\mathbb{P}[h(X) = \bar{h}(X) \mid h]$ into the new prior $\bar{p}_0$. Let $\mathbf{1}(\cdot)$ be the indicator function. Given a partial labeling $\mathcal{D}$, the posterior $\bar{p}_{\mathcal{D}}$ on $\overline{\mathcal{H}}$ is obtained from $\bar{p}_0$ by using Bayes' rule as:

$$\bar{p}_{\mathcal{D}}[\bar{h}] = \frac{\bar{p}_0[\bar{h}]\,\mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})}{\sum_{\bar{h} \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}]\,\mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})}.$$

We now prove that the probabilistic and deterministic models above are in fact equivalent in the sense that $p_{\mathcal{D}}[\mathbf{y}; S] = \bar{p}_{\mathcal{D}}[\mathbf{y}; S]$ for any $S \subseteq X \setminus x_{\mathcal{D}}$ and any label sequence $\mathbf{y}$ of S. This means that both models always give the same probability for the event that $\mathbf{y}$ is the label sequence of S. To prove this result, we first need the following lemma about $p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]$.

Lemma 1. *For any partial labeling $\mathcal{D}$, we have:*

$$p_0[y_{\mathcal{D}}; x_{\mathcal{D}}] = \sum_{\bar{h} \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}}).$$

Proof. In the probabilistic model, $p_0[y_{\mathcal{D}}; x_{\mathcal{D}}] = \sum_{h \in \mathcal{H}} p_0[h]\, \mathbb{P}[h(x_{\mathcal{D}}) = y_{\mathcal{D}} \,|\, h]$. Since h is a probabilistic hypothesis, we can expand $\mathbb{P}[h(x_{\mathcal{D}}) = y_{\mathcal{D}} \,|\, h]$ by summing over all possible labelings that agree with $\mathcal{D}$ on $x_{\mathcal{D}}$. This process is equivalent to summing over all possible label sequences of the remaining examples in $X \setminus x_{\mathcal{D}}$. Thus, we have:

$$\begin{aligned} p_0[y_{\mathcal{D}}; x_{\mathcal{D}}] &= \sum_{h \in \mathcal{H}} p_0[h] \sum_{\bar{h} \in \overline{\mathcal{H}}} \mathbb{P}[h(X) = \bar{h}(X) \,|\, h]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}}) \\ &= \sum_{\bar{h} \in \overline{\mathcal{H}}} \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}}) \sum_{h \in \mathcal{H}} p_0[h]\, \mathbb{P}[h(X) = \bar{h}(X) \,|\, h] \\ &= \sum_{\bar{h} \in \overline{\mathcal{H}}} \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})\, \bar{p}_0[\bar{h}]. \end{aligned}$$

Therefore, the lemma holds. □

Using Lemma 1, we can prove the following lemma about the equivalence of the probabilistic model and deterministic model.

Lemma 2. *Let $p_{\mathcal{D}}$ and $\bar{p}_{\mathcal{D}}$ be the posteriors of the probabilistic model and the deterministic model respectively after observing a partial labeling $\mathcal{D}$. For any $S \subseteq X \setminus x_{\mathcal{D}}$ and any label sequence $\mathbf{y}$ of S, we have:*

$$p_{\mathcal{D}}[\mathbf{y}; S] = \bar{p}_{\mathcal{D}}[\mathbf{y}; S].$$

Proof. For the probabilistic model, we have:

$$p_{\mathcal{D}}[\mathbf{y}; S] = \sum_{h \in \mathcal{H}} p_{\mathcal{D}}[h]\, \mathbb{P}[h(S) = \mathbf{y}|h] = \sum_{h \in \mathcal{H}} \frac{p_0[h]\, \mathbb{P}[h(x_{\mathcal{D}}) = y_{\mathcal{D}}|h]}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]}\, \mathbb{P}[h(S) = \mathbf{y}|h].$$

We expand $\mathbb{P}[h(x_{\mathcal{D}}) = y_{\mathcal{D}}|h]\, \mathbb{P}[h(S) = \mathbf{y}|h]$ by summing over all possible labelings that both agree with $\mathcal{D}$ on $x_{\mathcal{D}}$ and have $\mathbf{y}$ as the label sequence for S. Thus, we have:

$$\begin{aligned} p_{\mathcal{D}}[\mathbf{y}; S] &= \sum_{h \in \mathcal{H}} \frac{p_0[h]}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]} \sum_{\bar{h} \in \overline{\mathcal{H}}} \mathbb{P}[h(X) = \bar{h}(X) | h]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})\, \mathbf{1}(\bar{h}(S) = \mathbf{y}) \\ &= \sum_{\bar{h} \in \overline{\mathcal{H}}} \frac{\mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})\, \mathbf{1}(\bar{h}(S) = \mathbf{y})}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]} \sum_{h \in \mathcal{H}} p_0[h]\, \mathbb{P}[h(X) = \bar{h}(X) | h] \\ &= \sum_{\bar{h} \in \overline{\mathcal{H}}} \frac{\mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})\, \mathbf{1}(\bar{h}(S) = \mathbf{y})}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]}\, \bar{p}_0[\bar{h}]. \end{aligned}$$

The last equality above is from the definition of $\bar{p}_0[\bar{h}]$. From Lemma 1 and the definition of $\bar{p}_{\mathcal{D}}[\bar{h}]$, we have:

$$\frac{\bar{p}_0[\bar{h}]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})}{p_0[y_{\mathcal{D}}; x_{\mathcal{D}}]} = \frac{\bar{p}_0[\bar{h}]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})}{\sum_{\bar{h} \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}]\, \mathbf{1}(\bar{h}(x_{\mathcal{D}}) = y_{\mathcal{D}})} = \bar{p}_{\mathcal{D}}[\bar{h}].$$

Thus, $\displaystyle p_{\mathcal{D}}[\mathbf{y}; S] = \sum_{\bar{h} \in \overline{\mathcal{H}}} \bar{p}_{\mathcal{D}}[\bar{h}]\, \mathbf{1}(\bar{h}(S) = \mathbf{y}) = \bar{p}_{\mathcal{D}}[\mathbf{y}; S].$ □

The equivalence in Lemma 2 will be used to prove some results in the later chapters. Furthermore, because of this equivalence, some objective functions for Bayesian pool-based active learning can be formulated using either the probabilistic model or the corresponding deterministic model, or both.

We note that although the construction of the deterministic model requires an exponential increase in the hypothesis space size, this does not affect the computational efficiency of the algorithms considered in this thesis. In actual usages of those algorithms, we can do the computation or approximation directly on the probabilistic model without explicitly constructing the deterministic model. The deterministic model considered here is mainly used only for theoretical analysis of the algorithms.

CHAPTER 4

Submodular Function Maximization

In Bayesian pool-based active learning, our objective can often be stated as maximizing some average or worst-case performance with respect to some utility function $f(S)$ in the non-adaptive case, or $f(S, \bar{h})$ in the adaptive case, where S is the set of chosen examples and $\bar{h}$ is the true labeling of all examples (Golovin and Krause, 2011; Chen and Krause, 2013). When $f(S)$ is monotone submodular or $f(S, \bar{h})$ is adaptive monotone submodular, greedy algorithms are known to be near-optimal (Nemhauser et al., 1978; Golovin and Krause, 2011). In this chapter, we shall briefly summarize some results about greedy optimization of monotone submodular functions and adaptive monotone submodular functions, then prove a new result about the worst-case near-optimality of a greedy algorithm for maximizing pointwise monotone submodular functions. In the subsequent chapters, we will use the theorems in this chapter to prove near-optimal guarantees for various active learning algorithms. However, we note that the results in this chapter are general and can also be applied to settings other than active learning.

4.1 Near-optimality of Submodular Function Maximization

A set function $f : 2^X \to \mathbb{R}$ is *submodular* if it satisfies the following diminishing return property: for all $A \subseteq B \subseteq X$ and $x \in X \setminus B$,

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

The function f is called *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B$.

To select from X a set of size k that maximizes a monotone submodular function, one greedy strategy is to iteratively select the next example x^* that satisfies

$$x^* = \arg\max_{x \in X} \{f(S \cup \{x\}) - f(S)\}, \tag{4.1}$$

where S is the set of previously selected examples. The following theorem by Nemhauser et al. (1978) states the near-optimality of this greedy algorithm when maximizing a monotone submodular function.

Theorem 1 (Nemhauser et al. 1978)**.** *Let f be a monotone submodular function such that $f(\emptyset) = 0$, and let S_k be the set of examples selected up to iteration k using the greedy criterion in Equation* (4.1)*. Then for all $k > 0$, we have:*

$$f(S_k) \geq \left(1 - \frac{1}{e}\right) \max_{|S|=k} f(S).$$

4.2 Near-optimality of Adaptive Submodular Function Maximization

Adaptive submodularity (Golovin and Krause, 2011) is an extension of submodularity in the non-adaptive setting to the adaptive setting. For a partial labeling $\mathcal{D}$ and a full labeling $\bar{h}$ of X, we write $\bar{h} \sim \mathcal{D}$ to denote that $\mathcal{D}$ is consistent with $\bar{h}$. That is, $\mathcal{D} \subseteq \bar{h}$ when we view a labeling as a set of (x, y) pairs. For two partial labelings $\mathcal{D}$ and $\mathcal{D}'$, we call $\mathcal{D}$ a sub-labeling of $\mathcal{D}'$ if $\mathcal{D} \subseteq \mathcal{D}'$.

We consider a utility function $f : 2^X \times \mathcal{Y}^X \to \mathbb{R}_{\geq 0}$ which depends on the selected examples and the true labeling of X. For a partial labeling $\mathcal{D}$ and an example x, we define the expected utility gain when choosing x after observing $\mathcal{D}$ as:

$$\Delta(x|\mathcal{D}) \overset{\text{def}}{=} \mathbb{E}_{\bar{h}} \left[f(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f(x_{\mathcal{D}}, \bar{h}) \,|\, \bar{h} \sim \mathcal{D} \right],$$

where the expectation is with respect to $\bar{p}_0[\bar{h} \mid \bar{h} \sim \mathcal{D}]$ and $x_{\mathcal{D}}$ is the domain of $\mathcal{D}$.

From the definitions in (Golovin and Krause, 2011), f is *adaptive submodular* with respect to the prior $\bar{p}_0$ if for all $\mathcal{D}$ and $\mathcal{D}'$ such that $\mathcal{D} \subseteq \mathcal{D}'$, and for all $x \in X \setminus x_{\mathcal{D}'}$, we have:

$$\Delta(x|\mathcal{D}) \geq \Delta(x|\mathcal{D}').$$

Furthermore, f is adaptive monotone with respect to $\bar{p}_0$ if for all $\mathcal{D}$ with $\bar{p}_0[\bar{h} \sim \mathcal{D}] > 0$ and for all $x \in X$, we have $\Delta(x|\mathcal{D}) \geq 0$.

Let π be a policy for selecting the examples and $x_{\pi,\bar{h}}$ be the set of examples selected by π under the true labeling $\bar{h}$. We define the expected utility of π as

$$f_{\text{avg}}(\pi) \overset{\text{def}}{=} \mathbb{E}_{\bar{h} \sim \bar{p}_0} \left[f(x_{\pi,\bar{h}}, \bar{h}) \right].$$

To adaptively select from X a set of size k that maximizes f_{avg}, one greedy strategy is to iteratively select the next example x^* that satisfies

$$x^* = \arg\max_{x \in X} \Delta(x|\mathcal{D}), \tag{4.2}$$

where $\mathcal{D}$ is the partial labeling that has already been observed. The following theorem by Golovin and Krause (2011) states the near-optimality of this greedy policy when f is adaptive monotone submodular.

Theorem 2 (Golovin and Krause 2011)**.** *Let f be an adaptive monotone submodular function with respect to $\bar{p}_0$, let π be the adaptive policy selecting k examples using Equation* (4.2)*, and let π^* be the optimal policy with respect to f_{avg} that selects k examples. Then for all $k > 0$, we have:*

$$f_{avg}(\pi) > \left(1 - \frac{1}{e}\right) f_{avg}(\pi^*).$$

4.3 Near-optimality of Pointwise Submodular Function Maximization

In the previous section, Theorem 2 gives the near-optimal average-case performance guarantee for greedily optimizing an adaptive monotone submodular function. In this section, we prove a new near-optimal worst-case performance guarantee for greedily optimizing a pointwise monotone submodular function. A utility function $f : 2^X \times \mathcal{Y}^X \to \mathbb{R}_{\geq 0}$ is said to be *pointwise submodular* if the set function $f_{\bar{h}}(S) \stackrel{\text{def}}{=} f(S, \bar{h})$ is submodular for all labelings $\bar{h} \in \mathcal{Y}^X$. Similarly, f is pointwise monotone if $f_{\bar{h}}(S)$ is monotone for all $\bar{h}$. When f is pointwise monotone submodular, the average utility $f_{\text{avg}}(S) \stackrel{\text{def}}{=} \mathbb{E}_{\bar{h} \sim \bar{p}_0}[f(S, \bar{h})]$ is monotone submodular, and thus the non-adaptive greedy algorithm is a near-optimal non-adaptive policy for maximizing $f_{\text{avg}}(S)$ (Golovin and Krause, 2011). However, we are more interested in the adaptive policies in this section. In the following, for any partial labeling $\mathcal{D}$, any $x \in X \setminus x_{\mathcal{D}}$, and any $y \in \mathcal{Y}$, we write $\mathcal{D} \cup \{(x, y)\}$ to denote the partial labeling $\mathcal{D}$ with an additional mapping from x to y.

We assume that for any $S \subseteq X$ and any labeling $\bar{h}$, the value of $f(S, \bar{h})$ does not depend on the labels of examples in $X \setminus S$. We call this the *minimal dependency* property. Let us extend the definition of f so that its second parameter can be a partial labeling. The minimal dependency property implies that for any partial labeling $\mathcal{D}$ and any labeling $\bar{h} \sim \mathcal{D}$, we have $f(x_{\mathcal{D}}, \bar{h}) = f(x_{\mathcal{D}}, \mathcal{D})$. Without this minimal dependency property, the theorem in this section may not hold. We will see some examples of functions that satisfy or do not satisfy the minimal dependency property in Chapter 5 and 7.

For a partial labeling $\mathcal{D}$ and an example x, we define the worst-case utility gain when choosing x after observing $\mathcal{D}$ as:

$$\delta(x|\mathcal{D}) \stackrel{\text{def}}{=} \min_{y \in \mathcal{Y}} \{f(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x, y)\}) - f(x_{\mathcal{D}}, \mathcal{D})\}.$$

We consider the adaptive greedy strategy that iteratively selects the next example x^*

satisfying

$$x^* = \arg\max_{x \in X} \delta(x|\mathcal{D}), \tag{4.3}$$

where $\mathcal{D}$ is the partial labeling that has already been observed. For any policy π, let the worst-case utility of π be

$$f_{\text{worst}}(\pi) \stackrel{\text{def}}{=} \min_{\bar{h} \in \overline{\mathcal{H}}} f(x_{\pi,\bar{h}}, \bar{h}).$$

The following theorem states the near-optimality of the above greedy policy with respect to f_{worst} when f is pointwise monotone submodular.[1] The main idea in proving this theorem is to show that at every step, the greedy policy can cover at least $(1/k)$-fraction of the optimal remaining utility. This property can be proven by replacing the current greedy step with the optimal policy and considering the adversary's path for this optimal policy.

Theorem 3. *Let f be a pointwise monotone submodular function such that $f(\emptyset, \bar{h}) = 0$ for all $\bar{h} \in \overline{\mathcal{H}}$, and assume f satisfies the minimal dependency property. Let π be the adaptive policy selecting k examples using Equation* (4.3)*, and let π^* be the optimal policy with respect to f_{worst} that selects k examples. Then for all $k > 0$, we have:*

$$f_{worst}(\pi) > \left(1 - \frac{1}{e}\right) f_{worst}(\pi^*).$$

Proof. Consider the policy π. Let the worst-case labeling of π with respect to f be $\bar{h}_\pi = \arg\min_{\bar{h} \in \overline{\mathcal{H}}} f(x_{\pi,\bar{h}}, \bar{h})$. Then we have $f_{\text{worst}}(\pi) = f(x_{\pi,\bar{h}_\pi}, \bar{h}_\pi)$. Note that $\bar{h}_\pi$ corresponds to a path from the root to a leaf of the policy tree of π. Let the examples and labels along the path $\bar{h}_\pi$ (from root to leaf) be

$$\bar{h}_\pi = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}.$$

[1] Note that in the definition of $f_{\text{worst}}(\pi)$, $\bar{h}$ has to range over the set $\mathcal{Y}^X$ of *all* possible labelings. Otherwise, Theorem 3 does not necessarily hold.

Since f satisfies the minimal dependency property, let us abuse the notation and write $f(\{x_t\}_{t=1}^i, \{y_t\}_{t=1}^i)$ to denote $f(\{x_t\}_{t=1}^i, \bar{h}_\pi)$. Define

$$u_i = f\left(\{x_t\}_{t=1}^i, \{y_t\}_{t=1}^i\right) - f\left(\{x_t\}_{t=1}^{i-1}, \{y_t\}_{t=1}^{i-1}\right),$$

$$v_i = \sum_{t=1}^{i} u_t \qquad \text{and} \qquad z_i = f_{\text{worst}}(\pi^*) - v_i.$$

We prove the following claims.

Claim 1. *For all i, we have $u_{i+1} \geq \dfrac{z_i}{k}$.*

Proof of Claim 1. Consider the case that after observing $(x_1, y_1), \ldots, (x_i, y_i)$, we run the optimal policy π^* from its root and only follow the paths consistent with $(x_1, y_1), \ldots, (x_i, y_i)$ down to a leaf. In this case, all the paths of the policy π^* must obtain a value at least $z_i = f_{\text{worst}}(\pi^*) - v_i$, because running π^* without any observation would obtain at least $f_{\text{worst}}(\pi^*)$ and the observations $(x_1, y_1), \ldots, (x_i, y_i)$ cover a value v_i. Now we consider the adversary's path of the policy π^* in this scenario which is defined as:

$$\bar{h}^{\text{adv}} = \{(x_1^{\text{adv}}, y_1^{\text{adv}}), (x_2^{\text{adv}}, y_2^{\text{adv}}), \ldots, (x_k^{\text{adv}}, y_k^{\text{adv}})\},$$

where $$\begin{aligned} y_j^{\text{adv}} = \arg\min_y \{ & f(\{x_t\}_{t=1}^i \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{x_j^{\text{adv}}\},\ \{y_t\}_{t=1}^i \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{y\}) \\ & - f(\{x_t\}_{t=1}^i \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1},\ \{y_t\}_{t=1}^i \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1})\} \end{aligned}$$
if x_j^{adv} has not appeared in $\{x_1, \ldots, x_i\}$. Otherwise, $y_j^{\text{adv}} = y_t$ if $x_j^{\text{adv}} = x_t$ for some $t \in \{1, \ldots, i\}$. From the previous discussion, $\bar{h}^{\text{adv}}$ covers a value at least z_i in k steps. Thus, one of its steps must cover a value at least z_i/k.

Hence, what remains is to show that doing the greedy step in π after observing $(x_1, y_1), \ldots, (x_i, y_i)$ is better than any single step along $\bar{h}^{\text{adv}}$. In the trivial case where $(x_j^{\text{adv}}, y_j^{\text{adv}}) \in \{(x_1, y_1), \ldots, (x_i, y_i)\}$, we obtain nothing in this step since $(x_j^{\text{adv}}, y_j^{\text{adv}})$ has already been observed. Thus, the above is true in this case. In the non-trivial case, we have:

$$\begin{aligned}
u_{i+1} &= f\left(\{x_t\}_{t=1}^{i+1}, \{y_t\}_{t=1}^{i+1}\right) - f\left(\{x_t\}_{t=1}^{i}, \{y_t\}_{t=1}^{i}\right) \\
&\geq \min_y \left\{ f\left(\{x_t\}_{t=1}^{i} \cup \{x_{i+1}\},\, \{y_t\}_{t=1}^{i} \cup \{y\}\right) - f\left(\{x_t\}_{t=1}^{i},\, \{y_t\}_{t=1}^{i}\right) \right\} \\
&\geq \min_y \left\{ f\left(\{x_t\}_{t=1}^{i} \cup \{x_j^{\text{adv}}\},\, \{y_t\}_{t=1}^{i} \cup \{y\}\right) - f\left(\{x_t\}_{t=1}^{i},\, \{y_t\}_{t=1}^{i}\right) \right\} \\
&\geq \min_y \{ f\left(\{x_t\}_{t=1}^{i} \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{x_j^{\text{adv}}\},\, \{y_t\}_{t=1}^{i} \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{y\}\right) \\
&\qquad - f\left(\{x_t\}_{t=1}^{i} \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1},\, \{y_t\}_{t=1}^{i} \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1}\right) \} \\
&= f\left(\{x_t\}_{t=1}^{i} \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{x_j^{\text{adv}}\},\, \{y_t\}_{t=1}^{i} \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1} \cup \{y_j^{\text{adv}}\}\right) \\
&\qquad - f\left(\{x_t\}_{t=1}^{i} \cup \{x_t^{\text{adv}}\}_{t=1}^{j-1},\, \{y_t\}_{t=1}^{i} \cup \{y_t^{\text{adv}}\}_{t=1}^{j-1}\right).
\end{aligned}$$

In the above, the second inequality is due to the greedy criterion, and the third inequality is due to the submodularity of f. Therefore, this claim is true. □

Claim 2. *For all $i \geq 0$, we have $z_i \leq (1 - \frac{1}{k})^i f_{\mathit{worst}}(\pi^*)$.*

Proof of Claim 2. We prove this claim by induction. For $i = 0$, this holds because $z_0 = f_{\text{worst}}(\pi^*)$ by definition. Assume that $z_i \leq (1 - \frac{1}{k})^i f_{\text{worst}}(\pi^*)$, then

$$\begin{aligned}
z_{i+1} &= f_{\text{worst}}(\pi^*) - v_{i+1} = f_{\text{worst}}(\pi^*) - v_i - u_{i+1} = z_i - u_{i+1} \\
&\leq z_i - \frac{z_i}{k} = (1 - \frac{1}{k}) z_i \leq (1 - \frac{1}{k})^{i+1} f_{\text{worst}}(\pi^*).
\end{aligned}$$

In the above, the first inequality is due to Claim 1. Therefore, this claim is true. □

To prove Theorem 3, we apply Claim 2 with $i = k$ and have

$$z_k \leq (1 - \frac{1}{k})^k f_{\text{worst}}(\pi^*) < \frac{1}{e} f_{\text{worst}}(\pi^*).$$

Hence, $f_{\text{worst}}(\pi) = v_k = f_{\text{worst}}(\pi^*) - z_k > (1 - \frac{1}{e}) f_{\text{worst}}(\pi^*)$. □

We note that in the worst-case setting, Golovin and Krause (2011) also considered the problem of minimizing the number of queries needed to achieve a target utility value.

However, their results mainly rely on the condition that the utility function is adaptive submodular, not the pointwise submodular condition considered in this section. It is also worth noting that our new greedy criterion in Equation (4.3) is different from the greedy criterion considered by Golovin and Krause (2011), which is essentially Equation (4.2). Thus, our result does not follow from their result and is developed using a different argument.

Another remark is that the pointwise submodularity property considered in this section does not necessarily imply adaptive submodularity. In Chapter 7, we will show an example of a function that is pointwise submodular but not adaptive submodular (in Theorem 13). Another example was also given by Golovin and Krause (2011).

CHAPTER 5

Properties of Maximum Entropy and Least Confidence Active Learning Algorithms

This chapter analyzes new theoretical properties of two commonly used greedy active learning algorithms: the maximum entropy algorithm and the least confidence algorithm. These two algorithms have been shown to have good performance in practice (Settles and Craven, 2008). The algorithms are equivalent in the binary-class case in the sense that they both choose the same examples to query, but they are different in the multiclass case. In this chapter, we prove that the maximum entropy algorithm may not have a near-optimality guarantee when maximizing the policy entropy, a natural objective function for the algorithm. On the other hand, we also prove that the least confidence algorithm has a near-optimality guarantee for maximizing the worst-case version space reduction objective.

5.1 The Maximum Entropy Criterion

A commonly used objective for active learning in the non-adaptive setting is to choose k training examples such that their Shannon entropy (Shannon, 1948) is maximal, as this reduces uncertainty in the later stage. In this section, we first give a generalization for the concept of Shannon entropy to general (both adaptive and non-adaptive) policies.

Formally, we define the *policy entropy* of a policy π as:

$$H_{\text{ent}}(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{\rho \sim p_0^\pi}[\, -\ln p_0^\pi[\rho]\,].$$

We recall that p_0^π is the distribution over all the paths ρ of the policy tree of π (see Section 3.1 for details). From this definition, policy entropy is the Shannon entropy of the paths in the policy tree. The policy entropy reduces to the Shannon entropy on a set of examples when the policy is non-adaptive.

For pool-based active learning, we argue that it is desirable to maximize the policy entropy $H_{\text{ent}}(\pi)$ as maximizing the policy entropy will minimize the expected posterior label entropy given the observations. More specifically, suppose a path ρ has been observed, then the labels of the remaining examples in $X \setminus x_\rho$ follow the distribution $p_\rho[\,\cdot\,; X \setminus x_\rho]$, where p_ρ is the posterior obtained after observing (x_ρ, y_ρ). We denote the entropy of this distribution by $G(\rho)$ and call it the posterior label entropy of the remaining examples given ρ. Formally,

$$G(\rho) \stackrel{\text{def}}{=} -\sum_{\mathbf{y}} p_\rho[\,\mathbf{y}; X \setminus x_\rho\,] \ln p_\rho[\,\mathbf{y}; X \setminus x_\rho\,],$$

where the summation is over all the possible label sequences $\mathbf{y}$ of $X \setminus x_\rho$. The posterior label entropy of a policy π is defined as:

$$G(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{\rho \sim p_0^\pi} G(\rho).$$

The following result gives a formal statement that maximizing policy entropy minimizes the posterior label entropy of the policy, or the uncertainty on the labels of the remaining unlabeled examples in the pool. Note that in the theorem, Π_k is the set of policies that select k examples.

Theorem 4. *For any $k \geq 1$, if a policy π in Π_k maximizes $H_{ent}(\pi)$, then π minimizes the posterior label entropy $G(\pi)$.*

Proof. Recall that $p_0[\mathbf{y}; S]$ is the probability that examples in S are assigned the label sequence $\mathbf{y}$. We also use $p_0[(\mathbf{y}, \mathbf{y}'); (S, S')]$ to refer to the probability that examples in S and S' are assigned label sequences $\mathbf{y}$ and $\mathbf{y}'$ respectively. Let $\mathbf{1}(A)$ be the indicator function for the event A. In this proof, note that if we fix a label sequence $\mathbf{y}$ of X, the path ρ followed from the root to a leaf of the policy tree during the execution of the policy π is unique (since we only consider deterministic policies).

We shall prove that $H_{\text{ent}}(\pi) + G(\pi)$ is the Shannon entropy of the label sequence distribution $p_0[\,\cdot\,; X]$ which is a constant; and the theorem will follow. Indeed, the Shannon entropy of the distribution $p_0[\,\cdot\,; X]$ is:

$$
\begin{aligned}
& -\sum_{\mathbf{y}} p_0[\mathbf{y}; X] \ln p_0[\mathbf{y}; X] \\
= \;& -\sum_{\mathbf{y}} \sum_{\rho} \mathbf{1}(\mathbf{y} \text{ is consistent with } \rho)\, p_0[\mathbf{y}; X]\; \ln p_0[\mathbf{y}; X] \\
= \;& -\sum_{\rho} \sum_{\mathbf{y}} \mathbf{1}(\mathbf{y} \text{ is consistent with } \rho)\, p_0[\mathbf{y}; X]\; \ln p_0[\mathbf{y}; X] \\
= \;& -\sum_{\rho} \sum_{\mathbf{y}'} p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)]\; \ln p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)] \\
= \;& -\sum_{\rho} \sum_{\mathbf{y}'} p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)]\; (\ln p_0[y_\rho; x_\rho] + \ln p_\rho[\mathbf{y}'; X \setminus x_\rho]) \\
= \;& -\sum_{\rho} \sum_{\mathbf{y}'} p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)] \ln p_0[y_\rho; x_\rho] \\
& -\sum_{\rho} \sum_{\mathbf{y}'} p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)] \ln p_\rho[\mathbf{y}'; X \setminus x_\rho] \\
= \;& -\sum_{\rho} p_0[y_\rho; x_\rho] \ln p_0[y_\rho; x_\rho] - \sum_{\rho} \sum_{\mathbf{y}'} p_0[y_\rho; x_\rho] p_\rho[\mathbf{y}'; X \setminus x_\rho] \ln p_\rho[\mathbf{y}'; X \setminus x_\rho] \\
= \;& H_{\text{ent}}(\pi) + \sum_{\rho} p_0[y_\rho; x_\rho] G(\rho) \\
= \;& H_{\text{ent}}(\pi) + G(\pi).
\end{aligned}
$$

Therefore, Theorem 4 holds. □

To maximize the policy entropy H_{ent}, one natural greedy method is to choose at every iteration the example whose posterior label distribution has the maximum Shannon entropy. This is in fact the well-known *maximum entropy* active learning criterion

(Settles, 2010). Formally, this criterion chooses the next example x^* that satisfies

$$x^* = \arg\max_{x \in X} \mathbb{E}_{y \sim p_{\mathcal{D}}[\,\cdot\,;x]} \left[- \ln p_{\mathcal{D}}[y;x] \right], \tag{5.1}$$

where $p_{\mathcal{D}}$ is the posterior obtained after observing the partial labeling $\mathcal{D}$.

Due to the monotonicity and submodularity of Shannon entropy (Fujishige, 1978), we can construct a non-adaptive greedy policy that achieves near-optimality with respect to the objective function H_{ent} in the non-adaptive setting. In the adaptive setting, however, the maximum entropy criterion may not be near-optimal with respect to the objective function H_{ent} in general. We prove this negative result in Theorem 5 below.

The main idea in proving this theorem is to construct a set of independent distractor examples that have highest entropy but provide no information about the true hypothesis. The greedy criterion is tricked to choose only these distractor examples. On the other hand, there is an identifier example which gives the identity of the true hypothesis but has a lower entropy than the distractor examples. Once the label of the identifier example is revealed, there will be a number of high entropy examples to query, so that the policy entropy achieved is higher than that of the greedy algorithm.

Theorem 5. *Let π be the adaptive policy in Π_k selecting examples using Equation* (5.1)*, and let π^* be the optimal adaptive policy in Π_k with respect to H_{ent}. For any $0 < \alpha < 1$, there exists a problem where $\dfrac{H_{ent}(\pi)}{H_{ent}(\pi^*)} < \alpha$.*

Proof. Let $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ with n probabilistic hypotheses, and assume a uniform prior on them. We construct k independent distractor instances $x_1, x_2, \ldots, x_k$ with identical output distributions for the n probabilistic hypotheses. Our aim is to trick the greedy algorithm π to select these k instances. Since the hypotheses are identical on these instances, the greedy algorithm learns nothing when receiving each label.

Let $H(Y_1)$ be the Shannon entropy of the prior label distribution of any x_i (this entropy is the same for all x_i). Since the greedy algorithm always selects the k instances

$x_1, x_2, \ldots, x_k$ and their labels are independent, we have

$$H_{\text{ent}}(\pi) = kH(Y_1).$$

Next, we construct an instance x_0 where its label will deterministically identify the probabilistic hypotheses. Specifically, $\mathbb{P}[h_i(x_0) = i \,|\, h_i] = 1$ for all i. Note that $H(Y_0) = \ln n$.

To make sure that the greedy algorithm π selects the distractor instances instead of x_0, a constraint is that $H(Y_1) > H(Y_0) = \ln n$. This constraint can be satisfied by, for example, allowing $\mathcal{Y}$ to have $n + 1$ labels and letting $\mathbb{P}[h(x_j)|h]$ be the uniform distribution for all $j \geq 1$ and $h \in \mathcal{H}$. In this case, $H(Y_1) = \ln(n + 1) > \ln n$.

We will compare the greedy algorithm π with an algorithm π_A that selects x_0 first, and hence knows the true hypothesis after observing its label.

Finally, we construct $n(k - 1)$ more instances, and the algorithm π_A will select the appropriate $k - 1$ instances from them after figuring out the true hypothesis. Let the instances be $\{x_{(i,j)} : 1 \leq i \leq n \text{ and } 1 \leq j \leq k - 1\}$. Let $Y^h_{(i,j)}$ be the (random) label of $x_{(i,j)}$ according to the hypothesis h. For all $h \in \mathcal{H}$, $Y^h_{(i,j)}$ has identical distributions for $1 \leq j \leq k - 1$. Thus, we only need to specify $Y^h_{(i,1)}$.

We specify $Y^h_{(i,1)}$ as follows. If $h \neq h_i$, then let $\mathbb{P}[Y^h_{(i,1)} = 0] = 1$. Otherwise, let $\mathbb{P}[Y^h_{(i,1)} = 0] = 0$, and the distribution on other labels has entropy $H(Y^{h_1}_{(1,1)})$, as all hypotheses are defined the same way.

When the true hypothesis is unknown, the distribution for $Y_{(1,1)}$ has entropy

$$H(Y_{(1,1)}) = H(1 - \frac{1}{n}) + \frac{1}{n} H(Y^{h_1}_{(1,1)}),$$

where $H(1 - \frac{1}{n})$ is the entropy of the Bernoulli distribution $(1 - \frac{1}{n}, \frac{1}{n})$. As we want the greedy algorithm to select the distractors, we also need $H(Y_1) > H(Y_{(1,1)})$, giving $H(Y^{h_1}_{(1,1)}) < n(H(Y_1) - H(1 - \frac{1}{n}))$.

Algorithm π_A first selects x_0, identifies the true hypothesis exactly, and then selects $k-1$ instances with entropy $H(Y^{h_1}_{(1,1)})$. Thus,

$$H_{\text{ent}}(\pi_A) = \ln n + (k-1)H(Y^{h_1}_{(1,1)}).$$

Hence, we have $\dfrac{H_{\text{ent}}(\pi)}{H_{\text{ent}}(\pi_A)} = \dfrac{kH(Y_1)}{\ln n + (k-1)H(Y^{h_1}_{(1,1)})}$.

Set $H(Y^{h_1}_{(1,1)})$ to $n(H(Y_1) - H(1-\frac{1}{n})) - c$ for some small constant c. The above equation becomes

$$\frac{H_{\text{ent}}(\pi)}{H_{\text{ent}}(\pi_A)} = \frac{kH(Y_1)}{\ln n + (k-1)n(H(Y_1) - H(1-\frac{1}{n})) - (k-1)c}.$$

Since $H(1-\frac{1}{n})$ approaches 0 as n grows and $H(Y_1) = \ln(n+1)$, we can make the ratio $H_{\text{ent}}(\pi)/H_{\text{ent}}(\pi_A)$ as small as we like by increasing n. Furthermore,

$$\frac{H_{\text{ent}}(\pi)}{H_{\text{ent}}(\pi_A)} \geq \frac{H_{\text{ent}}(\pi)}{H_{\text{ent}}(\pi^*)}.$$

Thus, Theorem 5 holds. □

5.2 The Least Confidence Criterion

Another well-known active learning algorithm in the pool-based setting uses the *least confidence* criterion. This criterion chooses the next example whose most likely label has minimal posterior probability (Lewis and Gale, 1994; Culotta and McCallum, 2005). Formally, this criterion chooses the next examples x^* that satisfies

$$x^* = \arg\min_{x \in X} \left\{ \max_{y \in \mathcal{Y}} p_{\mathcal{D}}[y; x] \right\}. \tag{5.2}$$

Note that $x^* = \arg\max_x \{1 - \max_y p_{\mathcal{D}}[y; x]\}$. Thus, the least confidence criterion greedily optimizes the error rate of the Bayes classifier on the distribution $p_{\mathcal{D}}[\,\cdot\,; x]$. Theoretically, little has been known about the near-optimality of this criterion in the multiclass setting. In this section, we use the theory in Section 4.3 to prove that the least confidence criterion near-optimally maximizes the worst-case version space reduction. For $S \subseteq X$ and $\bar{h} \in \overline{\mathcal{H}}$, the *version space reduction* function is defined as:

$$f(S, \bar{h}) \stackrel{\text{def}}{=} 1 - \bar{p}_0[\bar{h}(S); S]. \tag{5.3}$$

This is the probability that a random labeling drawn from $\bar{p}_0$ does not agree with $\bar{h}$ on S. For a policy π, we define the *worst-case version space reduction* objective as:

$$H_{\text{lc}}(\pi) \stackrel{\text{def}}{=} \min_{\bar{h} \in \overline{\mathcal{H}}} f(x_{\pi,\bar{h}}, \bar{h}),$$

where $x_{\pi,\bar{h}}$ is the set of examples selected by π under the true labeling $\bar{h}$.

It can be shown that f is pointwise monotone submodular, and the least confidence criterion is equivalent to the criterion in Equation (4.3). Thus, it follows from Theorem 3 that the least confidence criterion is near-optimal with respect to the objective function H_{lc}. Theorem 6 below proves this result.

Theorem 6. *Let π be the adaptive policy in Π_k selecting examples using Equation* (5.2)*, and let π^* be the optimal adaptive policy in Π_k with respect to H_{lc}. Then for all $k > 0$, we have:*

$$H_{lc}(\pi) > \left(1 - \frac{1}{e}\right) H_{lc}(\pi^*).$$

Proof. It is clear that the version space reduction function f satisfies the minimal dependency property, is pointwise monotone, and $f(\emptyset, \bar{h}) = 0$ for all $\bar{h}$. For a partial labeling $\mathcal{D}$, let $x_{\mathcal{D}}$ be the domain of $\mathcal{D}$, and let $y_{\mathcal{D}} = \mathcal{D}(x_{\mathcal{D}})$. From Equation (4.3), we have:

$$\begin{aligned}
& \arg\max_x \min_y \{f\left(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x,y)\}\right) - f(x_{\mathcal{D}}, \mathcal{D})\} \\
= \quad & \arg\max_x \min_y f\left(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x,y)\}\right) \\
= \quad & \arg\max_x \min_y \left[1 - \bar{p}_0\left[y_{\mathcal{D}} \cup \{y\};\, x_{\mathcal{D}} \cup \{x\}\right]\right]
\end{aligned}$$

$$\begin{aligned}
&= \arg\min_x \max_y \bar{p}_0 \left[y_{\mathcal{D}} \cup \{y\}; x_{\mathcal{D}} \cup \{x\} \right] \\
&= \arg\min_x \max_y \frac{\bar{p}_0 \left[y_{\mathcal{D}} \cup \{y\}; x_{\mathcal{D}} \cup \{x\} \right]}{\bar{p}_0 \left[y_{\mathcal{D}}; x_{\mathcal{D}} \right]} \\
&= \arg\min_x \max_y \bar{p}_{\mathcal{D}}[y; x] \\
&= \arg\min_x \max_y p_{\mathcal{D}}[y; x].
\end{aligned}$$

In the above, the second equality is from the definition of f, and the last equality is from Lemma 2 in Section 3.2. Thus, Equation (5.2) is equivalent to Equation (4.3).

To apply Theorem 3, what remains is to show that f is pointwise submodular. Consider $f_{\bar{h}}(S) = f(S, \bar{h})$ for any $\bar{h} \in \overline{\mathcal{H}}$. Fix $A \subseteq B \subseteq X$ and $x \in X \setminus B$. We have:

$$\begin{aligned}
f_{\bar{h}}(A \cup \{x\}) - f_{\bar{h}}(A) &= \bar{p}_0[\bar{h}(A); A] - \bar{p}_0[\bar{h}(A \cup \{x\}); A \cup \{x\}] \\
&= \sum_{\bar{h}'(A) = \bar{h}(A)} \bar{p}_0[\bar{h}'] - \sum_{\substack{\bar{h}'(A) = \bar{h}(A) \\ \bar{h}'(x) = \bar{h}(x)}} \bar{p}_0[\bar{h}'] \\
&= \sum_{\bar{h}' \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}'] \, \mathbf{1}(\bar{h}'(A) = \bar{h}(A)) \, \mathbf{1}(\bar{h}'(x) \neq \bar{h}(x)).
\end{aligned}$$

Similarly, we have:

$$f_{\bar{h}}(B \cup \{x\}) - f_{\bar{h}}(B) = \sum_{\bar{h}' \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}'] \, \mathbf{1}(\bar{h}'(B) = \bar{h}(B)) \, \mathbf{1}(\bar{h}'(x) \neq \bar{h}(x)).$$

Since $A \subseteq B$, all pairs $\bar{h}, \bar{h}'$ such that $\bar{h}'(B) = \bar{h}(B)$ also satisfy $\bar{h}'(A) = \bar{h}(A)$. Thus, $f_{\bar{h}}(A \cup \{x\}) - f_{\bar{h}}(A) \geq f_{\bar{h}}(B \cup \{x\}) - f_{\bar{h}}(B)$ and $f_{\bar{h}}$ is submodular. Therefore, f is pointwise submodular and Theorem 6 holds. □

CHAPTER 6

The Maximum Gibbs Error Criterion

In the previous chapter, we have proven that the maximum entropy algorithm does not always have a near-optimality guarantee in the average case while the least confidence algorithm has a near-optimality guarantee in the worst case. In this chapter, we shall introduce a new greedy criterion, the maximum Gibbs error criterion, that has a near-optimality guarantee in the average case.

First, we propose a new objective for Bayesian pool-based active learning: the policy Gibbs error. This new objective is a lower bound of the policy entropy introduced in Section 5.1 and we shall prove that the maximum Gibbs error criterion greedily optimizes this new objective with a near-optimality guarantee in the average case. Intuitively, the policy Gibbs error of a policy π is the expected probability for a Gibbs classifier to make an error on the set adaptively selected by π. Formally, we define the *policy Gibbs error* of a policy π as:

$$H_{\text{gibbs}}(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{\rho \sim p_0^\pi} \left[1 - p_0^\pi[\rho] \right]. \tag{6.1}$$

In the above equation, $1 - p_0^\pi[\rho]$ is the probability that a Gibbs classifier makes an error on the selected set along the path ρ. Theorem 7 below, which is straightforward from the inequality $x \geq 1 + \ln x$, states that the policy Gibbs error is a lower bound of the policy entropy.

Theorem 7. *For any (full or partial) policy π, we have $H_{gibbs}(\pi) \leq H_{ent}(\pi)$.*

As an analogue to Theorem 4 in Chapter 5, we show in Theorem 8 below that any policy maximizing the policy Gibbs error will minimize the average weighted posterior Gibbs error of the remaining examples. Formally, the average weighted posterior Gibbs error is defined as:

$$\mathcal{G}(\pi) \overset{\text{def}}{=} \mathbb{E}_{\rho \sim p_0^\pi} \left[p_0^\pi[\rho] \, \mathcal{G}(\rho) \right],$$

where $\mathcal{G}(\rho) \overset{\text{def}}{=} 1 - \sum_{\mathbf{y}} p_\rho^2[\,\mathbf{y}; X \setminus x_\rho\,]$ is the posterior Gibbs error of the remaining examples after selecting the path ρ.

Theorem 8. *For any $k \geq 1$, if a policy π in Π_k maximizes $H_{gibbs}(\pi)$, then π minimizes the average weighted posterior Gibbs error $\mathcal{G}(\pi)$.*

Proof. We shall prove that $H_{\text{gibbs}}(\pi) + \mathcal{G}(\pi)$ is a constant, and the theorem will follow. Using similar notations as in the proof of Theorem 4, we have:

$$\begin{aligned}
& \sum_{\mathbf{y}} p_0[\mathbf{y}; X](1 - p_0[\mathbf{y}; X]) \\
= \;& \sum_{\mathbf{y}} \sum_{\rho} \mathbf{1}(\mathbf{y} \text{ is consistent with } \rho) \, p_0[\mathbf{y}; X](1 - p_0[\mathbf{y}; X]) \\
= \;& \sum_{\rho} \sum_{\mathbf{y}} \mathbf{1}(\mathbf{y} \text{ is consistent with } \rho) \, p_0[\mathbf{y}; X](1 - p_0[\mathbf{y}; X]) \\
= \;& \sum_{\rho} \sum_{\mathbf{y}'} p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)](1 - p_0[(y_\rho, \mathbf{y}'); (x_\rho, X \setminus x_\rho)]) \\
= \;& \sum_{\rho} \sum_{\mathbf{y}'} p_0[y_\rho; x_\rho] p_\rho[\mathbf{y}'; X \setminus x_\rho] \, (1 - p_0[y_\rho; x_\rho] p_\rho[\mathbf{y}'; X \setminus x_\rho]) \\
= \;& \sum_{\rho} p_0[y_\rho; x_\rho](1 - p_0[y_\rho; x_\rho] \sum_{\mathbf{y}'} p_\rho^2[\mathbf{y}'; X \setminus x_\rho]) \\
= \;& H_{\text{gibbs}}(\pi) + (\sum_{\rho} p_0[y_\rho; x_\rho](1 - p_0[y_\rho; x_\rho] \sum_{\mathbf{y}'} p_\rho^2[\mathbf{y}'; X \setminus x_\rho]) \\
& \qquad - \sum_{\rho} p_0[y_\rho; x_\rho](1 - p_0[y_\rho; x_\rho])) \\
= \;& H_{\text{gibbs}}(\pi) + \sum_{\rho} p_0^2[y_\rho; x_\rho](1 - \sum_{\mathbf{y}'} p_\rho^2[\mathbf{y}'; X \setminus x_\rho]) \\
= \;& H_{\text{gibbs}}(\pi) + \mathbb{E}_{\rho \sim p_0^\pi}[\, p_0[y_\rho; x_\rho] \, \mathcal{G}(\rho) \,] \\
= \;& H_{\text{gibbs}}(\pi) + \mathcal{G}(\pi).
\end{aligned}$$

Since $\sum_{\mathbf{y}} p_0[\mathbf{y}; X](1 - p_0[\mathbf{y}; X])$ is a constant, Theorem 8 holds. □

Given a budget of k queries, our proposed objective is to find

$$\pi^* = \arg\max_{\pi \in \Pi_k} H_{\text{gibbs}}(\pi),$$

the height k policy with maximum policy Gibbs error. By maximizing $H_{\text{gibbs}}(\pi)$, we hope to maximize the policy entropy $H_{\text{ent}}(\pi)$, and thus minimize the uncertainty in the remaining examples. Furthermore, we also hope to obtain a small expected error of a posterior Gibbs classifier, which upper bounds the Bayes error but is at most twice of it. In the next section, we shall describe greedy algorithms that are provably near-optimal for optimizing this objective.

6.1 Near-optimal Greedy Algorithms for Maximizing Policy Gibbs Error

In this section, we consider three settings: non-adaptive, adaptive and batch mode settings. We shall describe the corresponding greedy algorithms for each setting and prove their near-optimality when maximizing the policy Gibbs error.

6.1.1 The Non-adaptive Setting

In the non-adaptive setting, the policy π ignores the observed labels: it never updates the posterior. This is equivalent to selecting a set of examples before any labeling is done. In this setting, the examples selected along all paths of π are the same. Let x_π be the set of examples selected by π. The policy Gibbs error of a non-adaptive policy π is simply

$$H_{\text{gibbs}}(\pi) = \mathbb{E}_{\mathbf{y} \sim p_0[\,\cdot\,; x_\pi]}[1 - p_0[\mathbf{y}; x_\pi]],$$

where $p_0[\,\cdot\,; x_\pi]$ is the probability distribution induced by p_0 on the set of all label sequences of x_π. Thus, the optimal non-adaptive policy selects a set S of examples maximizing its Gibbs error, which is defined by:

$$\epsilon_g^{p_0}(S) \stackrel{\text{def}}{=} 1 - \sum_{\mathbf{y}} p_0[\mathbf{y}; S]^2,$$

where the summation is over all the label sequences $\mathbf{y}$ of S.

In general, the Gibbs error of a distribution P is $1 - \sum_i P[i]^2$, where the summation is over elements in the support of P. The Gibbs error is a special case of the Tsallis entropy used in nonextensive statistical mechanics (Tsallis and Brigatti, 2004) and is known to be monotone submodular (Sayrafi et al., 2008). From the properties of monotone submodular functions introduced in Section 4.1, the greedy non-adaptive policy that selects the next example satisfying

$$x^* = \arg\max_{x \in X} \left\{ \epsilon_g^{p_0}(S_i \cup \{x\}) \right\} = \arg\max_{x \in X} \left\{ 1 - \sum_{\mathbf{y}} p_0[\mathbf{y}; S_i \cup \{x\}]^2 \right\}, \qquad (6.2)$$

where S_i is the set of previously selected examples, is near-optimal compared to the best non-adaptive policy. In the above formula, the summation is over all the label sequences $\mathbf{y}$ of $S_i \cup \{x\}$. This result is formally stated in Theorem 9 below. The theorem is a direct consequence of Theorem 1 in Section 4.1.

Theorem 9. *Given a budget of $k \geq 1$ queries, let π_n be the non-adaptive policy in Π_k selecting examples using Equation* (6.2)*, and let π_n^* be the non-adaptive policy in Π_k with the maximum policy Gibbs error. Then for all $k \geq 1$, we have:*

$$H_{gibbs}(\pi_n) \geq \left(1 - \frac{1}{e}\right) H_{gibbs}(\pi_n^*).$$

6.1.2 The Adaptive Setting

In the adaptive setting, a policy takes into account the observed labels when choosing the next example. This is done via the posterior update after observing the label of a selected example. The adaptive setting is the most common setting for active learning.

We now describe a greedy adaptive algorithm for this setting that is near-optimal.

Assume that the current posterior obtained after observing the labeled examples $\mathcal{D}$ is $p_{\mathcal{D}}$. Our greedy algorithm selects the next example x^* that maximizes $\epsilon_g^{p_{\mathcal{D}}}(x)$:

$$x^* = \arg\max_{x \in X} \epsilon_g^{p_{\mathcal{D}}}(x) = \arg\max_{x \in X} \left\{ 1 - \sum_{y \in \mathcal{Y}} p_{\mathcal{D}}[y; x]^2 \right\}. \tag{6.3}$$

From the definition of $\epsilon_g^{p_{\mathcal{D}}}$ in Section 6.1.1, $\epsilon_g^{p_{\mathcal{D}}}(x)$ is in fact the policy Gibbs error of a 1-step policy with respect to the prior $p_{\mathcal{D}}$. Thus, we call this greedy criterion the adaptive *maximum Gibbs error criterion* (maxGEC).

Note that in binary classification where $|\mathcal{Y}| = 2$, maxGEC selects the same examples as the maximum Shannon entropy and the least confidence criteria. However, they are different in the multiclass case. Theorem 10 below states that maxGEC is near-optimal compared to the best adaptive policy with respect to the objective in Equation (6.1).

Theorem 10. *Given a budget of $k \geq 1$ queries, let π^{maxGEC} be the adaptive policy in Π_k selecting examples using* maxGEC *and let π^* be the adaptive policy in Π_k with the maximum policy Gibbs error. Then for all $k \geq 1$, we have:*

$$H_{gibbs}(\pi^{\text{maxGEC}}) > \left(1 - \frac{1}{e}\right) H_{gibbs}(\pi^*).$$

The main idea to prove this theorem is to reduce probabilistic hypotheses to deterministic ones by expanding the hypothesis space as described in Section 3.2 and then show that maxGEC greedily maximizes the average version space reduction in the deterministic model. Recall from Section 5.2 that the version space reduction function is $f(S, \bar{h}) = 1 - \bar{p}_0[\bar{h}(S); S]$. In the deterministic model $\overline{\mathcal{H}}$, the version space reduction function $f(S, \bar{h})$ is known to be adaptive monotone submodular (Golovin and Krause, 2011). Thus, the greedy adaptive policy selecting

$$x^* = \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}}[f(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f(x_{\mathcal{D}}, \bar{h})] \tag{6.4}$$

is near-optimal, where $x_{\mathcal{D}}$ is the previously selected set and $\bar{p}_{\mathcal{D}}$ is the current posterior of the deterministic model. This property is stated in Lemma 3 below and is a direct consequence of Theorem 5.2 by Golovin and Krause (2011).

Lemma 3. *For any $k \geq 1$, in the deterministic model, let π be the greedy adaptive policy that selects k examples by the criterion* (6.4)*. Let π^* be the adaptive policy that selects the optimal k examples in terms of the average version space reduction objective. We have:*

$$\mathbb{E}_{\bar{h}\sim\bar{p}_0}[f(x_{\pi,\bar{h}}, \bar{h})] > \left(1 - \frac{1}{e}\right) \mathbb{E}_{\bar{h}\sim\bar{p}_0}[f(x_{\pi^*,\bar{h}}, \bar{h})],$$

where $x_{\pi,\bar{h}}$ is the set of unlabeled examples selected by π assuming the true labeling of X is $\bar{h}$.

Using this lemma, what remains to prove Theorem 10 is to show that H_{gibbs} is equal to the average version space reduction, and maxGEC is equivalent to greedily maximizing the version space reduction at every iteration. We now prove these results.

Proof of Theorem 10. First, we prove that H_{gibbs} is equal to the average version space reduction. For any policy π, note that once we assume the true labeling of X to be a fixed $\bar{h}$, the policy π follows exactly one path from the root to a leaf in its policy tree. We denote this path by $\rho_{\pi,\bar{h}}$. We have:

$$\begin{aligned}
\mathbb{E}_{\bar{h}\sim\bar{p}_0}[f(x_{\pi,\bar{h}}, \bar{h})] &= \sum_{\bar{h}\in\overline{\mathcal{H}}} \bar{p}_0[\bar{h}] \left(1 - \bar{p}_0[\bar{h}(x_{\pi,\bar{h}}); x_{\pi,\bar{h}}]\right) \\
&= \sum_{\rho} \sum_{\bar{h}:\rho_{\pi,\bar{h}}=\rho} \bar{p}_0[\bar{h}] \left(1 - \bar{p}_0[y_\rho; x_\rho]\right) \\
&= \sum_{\rho} \left(1 - \bar{p}_0[y_\rho; x_\rho]\right) \sum_{\bar{h}:\rho_{\pi,\bar{h}}=\rho} \bar{p}_0[\bar{h}] \\
&= \sum_{\rho} \left(1 - \bar{p}_0[y_\rho; x_\rho]\right) \bar{p}_0[y_\rho; x_\rho] \\
&= \sum_{\rho} \left(1 - p_0^\pi[\rho]\right) p_0^\pi[\rho] \\
&= H_{\text{gibbs}}(\pi).
\end{aligned}$$

In the above, the first equality is from definition of f and expectation, the second equality is from the fact that $x_{\pi,\bar{h}} = x_\rho$ and $\bar{h}(x_{\pi,\bar{h}}) = y_\rho$ for all $\bar{h}$ satisfying $\rho_{\pi,\bar{h}} = \rho$, the fifth equality is from the equivalence in Lemma 2 and the definition of p_0^π, and the last equality is from the definition of H_{gibbs}.

Hence, the inequality in Lemma 3 is equivalent to $H_{\text{gibbs}}(\pi) > (1 - 1/e) H_{\text{gibbs}}(\pi^*)$. Thus, to prove Theorem 10, we need to prove that the example x^* selected by π^{maxGEC} using Equation (6.3) also satisfies Equation (6.4). In the deterministic model, for any $x \in X$, we have:

$$\begin{aligned}
\mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} \left[1 - f(x_{\mathcal{D}} \cup \{x\}, \bar{h}) \right] &= \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} \left[\bar{p}_0[\bar{h}(x_{\mathcal{D}} \cup \{x\}); x_{\mathcal{D}} \cup \{x\}] \right] \\
&= \sum_{\bar{h} \in \bar{\mathcal{H}} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_{\mathcal{D}}[\bar{h}] \, \bar{p}_0[\bar{h}(x_{\mathcal{D}} \cup \{x\}); x_{\mathcal{D}} \cup \{x\}] \\
&= \sum_{y \in \mathcal{Y}} \sum_{\substack{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0, \\ \bar{h}(x) = y}} \bar{p}_{\mathcal{D}}[\bar{h}] \, \bar{p}_0[\bar{h}(x_{\mathcal{D}} \cup \{x\}); x_{\mathcal{D}} \cup \{x\}].
\end{aligned}$$

For all $\bar{h}$ satisfying $\bar{p}_{\mathcal{D}}[\bar{h}] > 0$, we have $\bar{p}_{\mathcal{D}}[\bar{h}] = \bar{p}_0[\bar{h}] / \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_0[\bar{h}]$. Thus, if $\bar{h}$ also satisfies $\bar{h}(x) = y$, then:

$$\bar{p}_0[\bar{h}(x_{\mathcal{D}} \cup \{x\}); x_{\mathcal{D}} \cup \{x\}] = \sum_{\substack{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0, \\ \bar{h}(x) = y}} \bar{p}_0[\bar{h}] = \sum_{\substack{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0, \\ \bar{h}(x) = y}} \left(\bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_0[\bar{h}] \right).$$

Hence,

$$\begin{aligned}
&\mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} \left[1 - f(x_{\mathcal{D}} \cup \{x\}, \bar{h}) \right] \\
&= \sum_{y \in \mathcal{Y}} \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0 \wedge \bar{h}(x) = y} \left(\bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0 \wedge \bar{h}(x) = y} \left(\bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_0[\bar{h}] \right) \right) \\
&= \left(\sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_0[\bar{h}] \right) \left(\sum_{y \in \mathcal{Y}} \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0 \wedge \bar{h}(x) = y} \left(\bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0 \wedge \bar{h}(x) = y} \bar{p}_{\mathcal{D}}[\bar{h}] \right) \right) \\
&= \left(\sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0} \bar{p}_0[\bar{h}] \right) \left(\sum_{y \in \mathcal{Y}} \left(\sum_{\bar{h} : \bar{p}_{\mathcal{D}}[\bar{h}] > 0 \wedge \bar{h}(x) = y} \bar{p}_{\mathcal{D}}[\bar{h}] \right)^2 \right)
\end{aligned}$$

$$= \left(\sum_{\bar{h}:\bar{p}_{\mathcal{D}}[\bar{h}]>0} \bar{p}_0[\bar{h}] \right) \left(\sum_{y \in \mathcal{Y}} \bar{p}_{\mathcal{D}}[y;x]^2 \right).$$

By Lemma 2 and the above equality, the example x^* selected by Equation (6.3) satisfies

$$\begin{aligned} x^* = \arg\max_{x \in X} \{1 - \sum_{y \in \mathcal{Y}} p_{\mathcal{D}}[y;x]^2\} \quad &= \quad \arg\max_{x \in X} \{1 - \sum_{y \in \mathcal{Y}} \bar{p}_{\mathcal{D}}[y;x]^2\} \\ = \arg\min_{x \in X} \sum_{y \in \mathcal{Y}} \bar{p}_{\mathcal{D}}[y;x]^2 \quad &= \quad \arg\min_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [1 - f(x_{\mathcal{D}} \cup \{x\}, \bar{h})] \\ = \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [f(x_{\mathcal{D}} \cup \{x\}, \bar{h})] &= \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [f(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f(x_{\mathcal{D}}, \bar{h})]. \end{aligned}$$

Thus, x^* also satisfies Equation (6.4) and therefore Theorem 10 holds. □

6.1.3 The Batch Mode Setting

In the batch mode setting (Hoi et al., 2006b), we query the labels of s (instead of one) examples each time, and we do this for a given number of k iterations. After each iteration, we query the labels of the selected batch and update the posterior based on these labels. The new posterior can be used to select the next batch of examples. We call a policy in this setting a batch policy. A non-adaptive policy can be seen as a batch policy that selects only one batch.

Algorithm 6.1 describes a greedy algorithm for this setting which we call the *batch maxGEC* algorithm. At iteration i of the algorithm with the posterior p_i, the batch S is first initialized to be empty, then s examples are greedily chosen one at a time using the criterion:

$$x^* = \arg\max_{x \in X} \epsilon_g^{p_i}(S \cup \{x\}). \tag{6.5}$$

This is equivalent to running the non-adaptive greedy algorithm in Section 6.1.1 to select each batch. Query-labels(S) returns the true labels y_S of S and Posterior-update(p_i, S, y_S) returns the new posterior obtained from the prior p_i after observing y_S.

The following theorem states that batch maxGEC is near-optimal compared to the best

Algorithm 6.1: Batch maxGEC for Bayesian Batch Mode Active Learning

```
Input: Unlabeled pool $X$, prior $p_0$, number of iterations $k$, and batch size $s$.
for $i = 0$ to $k - 1$ do
    $S \leftarrow \emptyset$
    for $j = 0$ to $s - 1$ do
        $x^* \leftarrow \arg\max_{x \in X} \epsilon_g^{p_i}(S \cup \{x\})$
        $S \leftarrow S \cup \{x^*\}$
        $X \leftarrow X \setminus \{x^*\}$
    end
    $y_S \leftarrow$ Query-labels($S$)
    $p_{i+1} \leftarrow$ Posterior-update($p_i, S, y_S$)
end
```

batch policy with respect to the objective H_{gibbs} in Equation (6.1). The proof for this theorem also makes use of the reduction to deterministic hypotheses and the adaptive submodularity of version space reduction.

Theorem 11. *Given a budget of k batches of size s, let π_b^{maxGEC} be the batch policy selecting k batches using* batch maxGEC *and let π_b^* be the batch policy selecting k batches with maximum policy Gibbs error. We have:*

$$H_{gibbs}(\pi_b^{\text{maxGEC}}) > \left(1 - e^{-(e-1)/e}\right) H_{gibbs}(\pi_b^*).$$

Proof. In each iteration of Algorithm 6.1, the example x^* selected for the current batch by Equation (6.5) satisfies

$$x^* = \arg\max_{x \in X} \epsilon_g^p(S \cup \{x\}) = \arg\max_{x \in X} \left\{\epsilon_g^p(S \cup \{x\}) - \epsilon_g^p(S)\right\},$$

where p is the current posterior in the probabilistic model. From Theorem 9, the batch S

selected in each iteration of Algorithm 6.1 is near-optimal:

$$\epsilon_g^p(S) > \left(1 - \frac{1}{e}\right) \max_{S':|S'|=s} \epsilon_g^p(S').$$

To prove the near-optimality for the whole batch algorithm, we can employ the same deterministic model $\overline{\mathcal{H}}$ as in Section 3.2. From the definition of $\epsilon_g^p(S)$ and Lemma 2,

$$\epsilon_g^p(S) = 1 - \sum_{\mathbf{y}} p[\mathbf{y}; S]^2 = 1 - \sum_{\mathbf{y}} \bar{p}[\mathbf{y}; S]^2,$$

where $\bar{p}$ is the corresponding posterior of p in the deterministic model and the summations are over all possible label sequences $\mathbf{y}$ of S. Note that $1 - \sum_{\mathbf{y}} \bar{p}[\mathbf{y}; S]^2$ (and hence $\epsilon_g^p(S)$) is equivalent to the expected version space reduction in the deterministic model if S is chosen. So, in the deterministic model, Algorithm 6.1 is equivalent to the BatchGreedy algorithm proposed by Chen and Krause (2013). According to their results, the version space reduction after observing the label sequence of each batch is adaptive monotone submodular. Furthermore, the average version space reduction after selecting each batch by Algorithm 6.1 is an $e/(e-1)$-approximate greedy step. Using Theorem 5.2 of Golovin and Krause (2011), we have:

$$\mathbb{E}_{\bar{h} \sim \bar{p}_0} \left[f(x_{\pi_b^{\text{maxGEC}}, \bar{h}}, \bar{h}) \right] > \left(1 - e^{-(e-1)/e}\right) \mathbb{E}_{\bar{h} \sim \bar{p}_0} \left[f(x_{\pi_b^*, \bar{h}}, \bar{h}) \right],$$

where f is the version space reduction function, $\bar{p}_0$ is the prior of the deterministic model and $x_{\pi_b, \bar{h}}$ is the set of all examples selected by the batch algorithm π_b after k iterations ($k \times s$ examples in total), assuming the true labeling of the pool X is $\bar{h}$. From the proof of Theorem 10, $\mathbb{E}_{\bar{h} \sim \bar{p}_0}[f(x_{\pi_b, \bar{h}}, \bar{h})] = H_{\text{gibbs}}(\pi_b)$ for any policy π_b. Thus, we obtain Theorem 11. □

Theorem 11 has a different bounding constant than those in Theorems 9 and 10 because it uses two levels of approximation to compute the batch policy: at each iteration, it approximates the optimal batch by greedily choosing one example at a time using

Equation (6.5) (first approximation). Then it uses these chosen batches to approximate the optimal batch policy (second approximation). In contrast, the fully adaptive case has batch size 1 and only needs the second approximation, while the non-adaptive case chooses 1 batch and only needs the first approximation.

In non-adaptive and batch mode settings, our algorithms need to sum over all label sequences of the previously selected examples in a batch to choose the next example. This summation is usually expensive and it restricts the algorithms to small batches. However, we note that small batches may be preferred in some real problems. For example, if there is a small number of annotators and labeling one example takes a long time, we may want to select a batch size that matches the number of annotators. In this case, the annotators can label the examples concurrently while we can make use of the labels as soon as they are available. It would take a longer time to label a larger batch and we cannot use the labels until all the examples in the batch are labeled.

6.2 Computing maxGEC

In this section, we discuss how to compute maxGEC and batch maxGEC for some useful probabilistic models. Computing the values is often difficult and we discuss some sampling methods for this task.

6.2.1 MaxGEC for Bayesian Conditional Exponential Models

A conditional exponential model defines the conditional probability $P_\lambda[\vec{y} \,|\, \vec{x}]$ of a structured label $\vec{y}$ given a structured input $\vec{x}$ as:

$$P_\lambda[\vec{y} \,|\, \vec{x}] \stackrel{\text{def}}{=} \frac{1}{Z_\lambda(\vec{x})} \exp\left(\sum_{i=1}^{m} \lambda_i F_i(\vec{y}, \vec{x})\right),$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the parameter vector, $F_i(\vec{y}, \vec{x})$ is the total score of the i-th feature, and $Z_\lambda(\vec{x}) = \sum_{\vec{y}} \exp\left(\sum_{i=1}^{m} \lambda_i F_i(\vec{y}, \vec{x})\right)$ is the partition function.

A well-known conditional exponential model is the linear-chain conditional random field (CRF) (Lafferty et al., 2001) in which $\vec{x}$ and $\vec{y}$ both have sequence structures. That is, $\vec{x} = (x_1, x_2, \ldots, x_{|\vec{x}|}) \in \mathcal{X}^{|\vec{x}|}$ and $\vec{y} = (y_1, y_2, \ldots, y_{|\vec{x}|}) \in \mathcal{Y}^{|\vec{x}|}$. In this model,

$$F_i(\vec{y}, \vec{x}) = \sum_{j=1}^{|\vec{x}|} f_i(y_j, y_{j-1}, \vec{x}),$$

where $f_i(y_j, y_{j-1}, \vec{x})$ is the score of the i-th feature at position j.

In the Bayesian setting, we assume a prior $p_0[\lambda] = \prod_{i=1}^m p_0[\lambda_i]$ on λ, where $p_0[\lambda_i] = \mathcal{N}(\lambda_i|0, \sigma^2)$ is a Gaussian distribution with a known σ. After observing the labeled examples (partial labeling) $\mathcal{D} = \{(\vec{x}_j, \vec{y}_j)\}_{j=1}^t$, we can obtain the posterior

$$p_{\mathcal{D}}[\lambda] = p_0[\lambda|\mathcal{D}] \propto \prod_{j=1}^t \frac{1}{Z_\lambda(\vec{x}_j)} \exp\left(\sum_{i=1}^m \lambda_i F_i(\vec{y}_j, \vec{x}_j)\right) \exp\left(-\frac{1}{2}\sum_{i=1}^m \left(\frac{\lambda_i}{\sigma}\right)^2\right).$$

For active learning, we need to estimate the Gibbs error in Equation (6.3) from the posterior $p_{\mathcal{D}}$. For each $\vec{x}$, we can approximate the Gibbs error $\epsilon_g^{p_{\mathcal{D}}}(\vec{x}) = 1 - \sum_{\vec{y}} p_{\mathcal{D}}[\vec{y}; \vec{x}]^2$ by sampling N hypotheses $\lambda^1, \lambda^2, \ldots, \lambda^N$ from the posterior $p_{\mathcal{D}}$. In particular,

$$\begin{aligned}
& \sum_{\vec{y}} p_{\mathcal{D}}[\vec{y}; \vec{x}]^2 \\
\approx\ & \sum_{\vec{y}} \left(\frac{1}{N}\sum_{j=1}^N P_{\lambda^j}[\vec{y}|\vec{x}]\right)^2 \\
=\ & \frac{1}{N^2}\sum_{\vec{y}} \left(\sum_{j=1}^N \frac{\exp\left(\sum_{i=1}^m \lambda_i^j F_i(\vec{y}, \vec{x})\right)}{Z_{\lambda^j}(\vec{x})}\right)^2 \\
=\ & \frac{1}{N^2}\sum_{j=1}^N\sum_{t=1}^N \frac{1}{Z_{\lambda^j}(\vec{x}) Z_{\lambda^t}(\vec{x})} \sum_{\vec{y}} \exp\left(\sum_{i=1}^m \lambda_i^j F_i(\vec{y}, \vec{x})\right) \exp\left(\sum_{i=1}^m \lambda_i^t F_i(\vec{y}, \vec{x})\right) \\
=\ & \frac{1}{N^2}\sum_{j=1}^N\sum_{t=1}^N \frac{1}{Z_{\lambda^j}(\vec{x}) Z_{\lambda^t}(\vec{x})} \sum_{\vec{y}} \exp\left(\sum_{i=1}^m (\lambda_i^j + \lambda_i^t) F_i(\vec{y}, \vec{x})\right) \\
=\ & \frac{1}{N^2}\sum_{j=1}^N\sum_{t=1}^N \frac{Z_{\lambda^j+\lambda^t}(\vec{x})}{Z_{\lambda^j}(\vec{x}) Z_{\lambda^t}(\vec{x})}.
\end{aligned}$$

Thus, $\epsilon_g^{p_{\mathcal{D}}}(\vec{x}) \approx 1 - \frac{1}{N^2} \sum_{j=1}^{N} \sum_{t=1}^{N} \frac{Z_{\lambda^j + \lambda^t}(\vec{x})}{Z_{\lambda^j}(\vec{x}) Z_{\lambda^t}(\vec{x})}.$

If we only use the MAP hypothesis λ^* to approximate the Gibbs error (i.e., the non-Bayesian setting), then $N = 1$ and

$$\epsilon_g^{p_{\mathcal{D}}}(\vec{x}) \approx 1 - \frac{Z_{2\lambda^*}(\vec{x})}{Z_{\lambda^*}(\vec{x})^2}.$$

This approximation can be done efficiently if we can compute the partition functions $Z_\lambda(\vec{x})$ efficiently for any λ. This condition holds for a wide range of models including logistic regression, linear-chain CRF, semi-Markov CRF (Sarawagi and Cohen, 2004), and sparse high-order semi-Markov CRF (Nguyen et al., 2011; Cuong et al., 2014b, 2015).

We note that the above equation to approximate maxGEC from N sampled hypotheses demonstrates an advantage of the algorithm compared to the maximum entropy or least confidence algorithms. To our knowledge, there is no simple way to compute the latter criteria from a finite sample of hypotheses except for using only the MAP estimation. In particular, it is difficult to sum (or minimize) over all the outputs $\vec{y}$. For maxGEC, the summation can be rearranged to obtain the partition functions, which can be computed efficiently using known inference algorithms.

6.2.2 Batch maxGEC for Bayesian Transductive Naive Bayes

We now discuss an algorithm to approximate batch maxGEC for non-adaptive and batch mode active learning with Bayesian transductive Naive Bayes. First, we describe the Bayesian transductive Naive Bayes model for text classification. Let $Y \in \mathcal{Y}$ be a random variable denoting the label of a document and $W \in \mathcal{W}$ be a random variable denoting a word. In a Naive Bayes model, the parameters are $\theta = \{\theta_y\}_{y \in \mathcal{Y}} \cup \{\theta_{w|y}\}_{w \in \mathcal{W}, y \in \mathcal{Y}}$, where $\theta_y = \mathbb{P}[Y = y]$ and $\theta_{w|y} = \mathbb{P}[W = w | Y = y]$. For a document Z and a label Y, if $Z = \{W_1, W_2, \ldots, W_{|Z|}\}$ where W_i is a word in the document, then we model the

Algorithm 6.2: Approximation for Equation (6.5) in Bayesian transductive Naive Bayes model.

```
Input: Selected unlabeled examples $S$, current unlabeled example $x$, current
       posterior $p^{c}_{\mathcal{D}}$.
Sample $M$ label vectors $(\mathbf{y}^i)_{i=0}^{M-1}$ of $(X \setminus T) \cup \mathcal{T}$ from $p^{c}_{\mathcal{D}}$ using Gibbs
sampling.
$r \leftarrow 0$
for $i = 0$ to $M - 1$ do
    for $y \in \mathcal{Y}$ do
        $\widehat{p^{c}_{\mathcal{D}}}[\mathbf{y}^i_S \cup \{y\}; S \cup \{x\}] \leftarrow \frac{1}{M} \left| \left\{ \mathbf{y}^j : \mathbf{y}^j_S = \mathbf{y}^i_S \wedge \mathbf{y}^j_{\{x\}} = y \right\} \right|$
        $r \leftarrow r + \widehat{p^{c}_{\mathcal{D}}}[\mathbf{y}^i_S \cup \{y\}; S \cup \{x\}]^2$
    end
end
return $1 - r$
```

joint distribution $\mathbb{P}[Z, Y]$ as:

$$\mathbb{P}[Z, Y] \overset{\text{def}}{=} \theta_Y \prod_{i=1}^{|Z|} \theta_{W_i|Y}.$$

In the Bayesian setting, we assume that there is a prior $p_0[\theta]$ such that $\theta_y \sim \text{Dirichlet}(\alpha)$ and $\theta_{w|y} \sim \text{Dirichlet}(\alpha_y)$ for each y. When we observe the labeled documents, we update the posterior by counting the labels and the words in each document label. The posterior parameters also follow Dirichlet distributions.

Let X be the original pool of training examples and $\mathcal{T}$ be the unlabeled testing examples. In the transductive setting, we work with the conditional prior $p^{c}_{0}[\theta] = p_0[\theta | X; \mathcal{T}]$. For a set $\mathcal{D} = (T, \mathbf{y}_T)$ of labeled examples where $T \subseteq X$ is the set of unlabeled examples and $\mathbf{y}_T$ is the label sequence of T, the conditional posterior is $p^{c}_{\mathcal{D}}[\theta] = p_0[\theta | X; \mathcal{T}; \mathcal{D}] = p_{\mathcal{D}}[\theta | (X \setminus T) \cup \mathcal{T}]$, where $p_{\mathcal{D}}[\theta] = p_0[\theta | \mathcal{D}]$ is the Dirichlet posterior of the non-transductive model.

To implement the batch maxGEC algorithm, we need to estimate the Gibbs error in Equation (6.5) from the conditional posterior. Let S be the currently selected batch. For each unlabeled example $x \notin S$, we need to estimate:

$$1 - \sum_{\mathbf{y},y} p^c_{\mathcal{D}} \left[\mathbf{y} \cup \{y\}; S \cup \{x\}\right]^2 = 1 - \mathbb{E}_{\mathbf{y}} \left[\frac{\sum_y p^c_{\mathcal{D}} \left[\mathbf{y} \cup \{y\}; S \cup \{x\}\right]^2}{p^c_{\mathcal{D}}[\mathbf{y}; S]} \right],$$

where the expectation is with respect to the distribution $p^c_{\mathcal{D}}[\mathbf{y}; S]$. We can use Gibbs sampling to approximate this expectation. First, we sample M label vectors $\mathbf{y}_{(X \setminus T) \cup \mathcal{T}}$ of the remaining unlabeled examples from $p^c_{\mathcal{D}}$ using Gibbs sampling. Then, for each $\mathbf{y}$, we estimate $p^c_{\mathcal{D}}[\mathbf{y}; S]$ by counting the fraction of the M sampled vectors consistent with $\mathbf{y}$. For each $\mathbf{y}$ and y, we also estimate $p^c_{\mathcal{D}} \left[\mathbf{y} \cup \{y\}; S \cup \{x\}\right]$ by counting the fraction of the M sampled vectors consistent with both $\mathbf{y}$ and y on $S \cup \{x\}$. This approximation is equivalent to Algorithm 6.2. In the algorithm, $\mathbf{y}^i_S$ is the label sequence of S according to $\mathbf{y}^i$.

6.3 Experiments

In this section, we report the experimental results for maxGEC on a named entity recognition task with Bayesian CRF and for batch maxGEC on a text classification task with Bayesian transductive Naive Bayes model.

6.3.1 Named Entity Recognition (NER) with Bayesian CRF

In this experiment, we consider the NER task with the Bayesian CRF model described in Section 6.2.1. In this task, we need to label each word in English sentences with one of the following four types of named entities: persons, locations, organizations, and miscellaneous named entities (those that do not belong to the previous three groups) or we label the word with “none” to indicate that it is not part of any named entity. We use a subset of the CoNLL 2003 NER task (Tjong Kim Sang and De Meulder, 2003) which contains 1928 training and 969 test sentences.

Following the setting in (Settles and Craven, 2008), we let the cost of querying the label sequence of each sentence be 1. We implement two versions of maxGEC with the approximation algorithm in Section 6.2.1: the first version approximates Gibbs error by using only the MAP hypothesis (maxGEC-MAP), and the second version approximates Gibbs error by using 50 hypotheses sampled from the posterior (maxGEC-50). We sample the hypotheses for maxGEC-50 from the posterior by the Metropolis-Hastings algorithm (Chib and Greenberg, 1995) with the MAP hypothesis as the initial point.

We compare the maxGEC algorithms with 4 other learning criteria: passive learner (Passive), active learner which chooses the longest unlabeled sequence (Longest), active learner which chooses the unlabeled sequence with maximum Shannon entropy (SegEnt), and active learner which chooses the unlabeled sequence with the least confidence (LeastConf). For SegEnt and LeastConf, the entropy and confidence are estimated from the MAP hypothesis. As discussed in Section 6.2.1, there is no simple way to compute SegEnt or LeastConf criteria from a finite sample of hypotheses except for using only the MAP estimation. For all the algorithms, we use the MAP hypothesis for Viterbi decoding.

We compare the total area under the F_1 curve (AUC) for each algorithm after querying the first 500 sentences. The absolute AUC scores for the experiment are given in Figure 6.1. As a percentage of the maximum score of 500, algorithms Passive, Longest, SegEnt, LeastConf, maxGEC-MAP and maxGEC-50 attain 72.8, 67.0, 75.4, 75.5, 75.8 and 76.0 respectively. Hence, the maxGEC algorithms perform better than all the other algorithms, and significantly so over the Passive and Longest algorithms.

We test the statistical significance of our results using randomization tests (Noreen, 1989; Yeh, 2000) in which we randomly shuffle the outputs of two systems being compared and compute how likely the shuffle produces a difference in the AUCs. Because of the large data sizes, we use approximate tests where for each comparison, we make 10000 random shuffles and repeat this process 999 times. The significance level p is at most

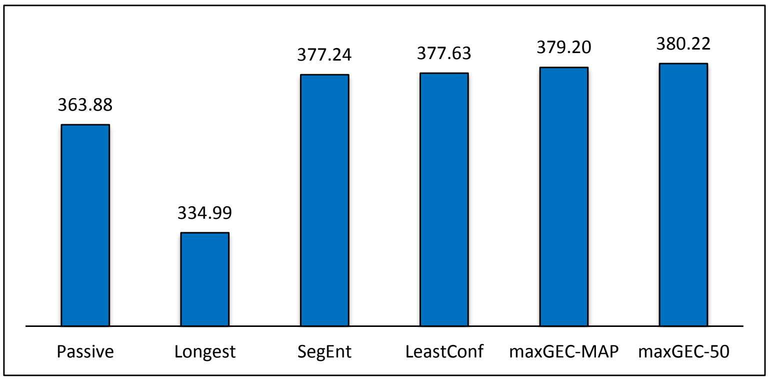

Figure 6.1: Absolute AUC scores under the F_1 curves on the CoNLL 2003 data set.

$(n_c + 1)/(n_t + 1)$, where n_c is the number of trials in which the difference between the AUCs is greater than the original difference, and n_t is the total number of iterations (Noreen, 1989; Yeh, 2000).

Our statistical significance tests show that maxGEC-MAP is significantly better than Passive, Longest (both with $p \leq 0.001$), and SegEnt (with $p \leq 0.005$). It is better than LeastConf with $p \leq 0.013$. On the other hand, maxGEC-50 is significantly better than Passive, Longest, SegEnt, and LeastConf with $p \leq 0.001$.

6.3.2 Text Classification with Bayesian Transductive Naive Bayes

In this experiment, we consider the text classification model in Section 6.2.2 with the meta-parameters $\alpha = (0.1, \ldots, 0.1)$ and $\alpha_y = (0.1, \ldots, 0.1)$ for all y. We implement batch maxGEC (maxGEC) with the approximation in Algorithm 6.2 and compare with 5 other algorithms: passive learner with Bayesian transductive Naive Bayes model (TPass), least confidence active learner with Bayesian transductive Naive Bayes model (LC), passive learner with Bayesian non-transductive Naive Bayes model (NPass), passive learner with logistic regression model (LogPass), and batch mode active learner with Fisher information matrix and logistic regression model (LogFisher) (Hoi et al., 2006b). To implement the least confidence algorithm, we sample M label vectors as in Algorithm

6.2 and use them to estimate the label distribution for each unlabeled example. The algorithm will then select s examples whose label is least confident according to these estimates.

We run the algorithms on 7 binary text classification tasks from the 20 Newsgroups data set (Joachims, 1996). In these tasks, we need to classify English news articles into one of two different topics (see the first column of Table 6.1). We run our experiments with batch size $s = 10, 20, 30$ and report the areas under the accuracy curve (AUC) for these cases in Table 6.1, 6.2, and 6.3 respectively. In the tables, bold figures indicate the best score on a row. The results are obtained by averaging over 5 different runs of the algorithms, and the AUCs are normalized so that their range is from 0 to 100. From the results, maxGEC obtains the best AUC scores on 4/7 tasks for each batch size and also the best average AUC scores. LC also performs well and its scores are only slightly lower than maxGEC. The passive learning algorithms are much worse than the active learning algorithms.

We also run the statistical significance tests described in Section 6.3.1 for the results in this experiments. For the batch size of 10, the tests show that maxGEC is significantly better than TPass, NPass, LogPass, and LogFisher in terms of average AUC scores with $p \leq 0.001$. For the batch sizes of 20 and 30, maxGEC is significantly better than all the competing algorithms (including LC) with $p \leq 0.001$.

Task	TPass	maxGEC	LC	NPass	LogPass	LogFisher
alt.atheism/comp.graphics	87.43	91.69	91.66	84.98	91.63	**93.92**
talk.politics.guns/talk.politics.mideast	84.92	92.03	**92.16**	80.80	86.07	88.36
comp.sys.mac.hardware/comp.windows.x	73.17	**93.60**	92.27	74.41	85.87	88.71
rec.motorcycles/rec.sport.baseball	93.82	**96.40**	96.23	92.33	89.46	93.90
sci.crypt/sci.electronics	60.46	85.51	85.86	60.85	82.89	**87.72**
sci.space/soc.religion.christian	92.38	**95.83**	95.45	89.72	91.16	94.04
soc.religion.christian/talk.politics.guns	91.57	**95.94**	95.59	85.56	90.35	93.96
Average	83.39	**93.00**	92.75	81.24	88.21	91.52

Table 6.1: AUCs (%) of different learning algorithms with batch size $s = 10$.

Task	TPass	maxGEC	LC	NPass	LogPass	LogFisher
alt.atheism/comp.graphics	87.62	91.52	91.70	84.85	91.28	**93.37**
talk.politics.guns/talk.politics.mideast	84.23	92.52	**92.56**	80.61	85.89	86.93
comp.sys.mac.hardware/comp.windows.x	73.96	**91.71**	89.98	74.79	85.83	88.06
rec.motorcycles/rec.sport.baseball	93.65	**95.95**	95.93	92.04	89.25	93.11
sci.crypt/sci.electronics	61.10	86.19	85.97	61.28	82.80	**86.93**
sci.space/soc.religion.christian	92.44	**95.77**	**95.77**	89.67	91.04	93.48
soc.religion.christian/talk.politics.guns	91.11	**94.56**	**94.56**	85.41	90.09	93.12
Average	83.44	**92.60**	92.35	81.23	88.02	90.71

Table 6.2: AUCs (%) of different learning algorithms with batch size $s = 20$.

Task	TPass	maxGEC	LC	NPass	LogPass	LogFisher
alt.atheism/comp.graphics	87.72	92.22	92.22	85.27	91.05	**92.88**
talk.politics.guns/talk.politics.mideast	85.13	**92.20**	92.17	81.00	85.63	86.35
comp.sys.mac.hardware/comp.windows.x	72.81	**88.58**	88.53	74.53	85.75	87.52
rec.motorcycles/rec.sport.baseball	94.03	96.21	**96.22**	92.09	89.03	92.22
sci.crypt/sci.electronics	61.71	86.12	85.25	61.62	82.74	**86.31**
sci.space/soc.religion.christian	91.09	**95.86**	**95.86**	88.76	90.88	92.82
soc.religion.christian/talk.politics.guns	91.00	**95.54**	**95.54**	85.19	89.65	91.89
Average	83.36	**92.39**	92.26	81.21	87.82	90.00

Table 6.3: AUCs (%) of different learning algorithms with batch size $s = 30$.

Chapter 7

The Generalized Maximum Gibbs Error Criterion

In the previous chapter, we have introduced the maximum Gibbs error criterion for Bayesian pool-based active learning that is near-optimal in the average case. In this chapter, we shall propose generalized versions of the maximum Gibbs error criterion that can incorporate any general loss function into the criteria. We shall also analyze the near-optimality of the new criteria in both the average case and worst case, and then show that they perform well in practice.

First, let us reconsider the maximum Gibbs error criterion in Chapter 6. Recall that the policy Gibbs error objective H_{gibbs} can be written as $H_{\text{gibbs}}(\pi) = \mathbb{E}_{\bar{h} \sim \bar{p}_0}[f(x_{\pi,\bar{h}}, \bar{h})]$, where f is the version space reduction function defined in Equation (5.3) of Section 5.2. Note that $f(x_{\pi,\bar{h}}, \bar{h}) = 1 - \bar{p}_0[\bar{h}(x_{\pi,\bar{h}}); x_{\pi,\bar{h}}]$ is the expected 0-1 loss that a random labeling drawn from $\bar{p}_0$ differs from $\bar{h}$ on $x_{\pi,\bar{h}}$. Because of the nature of 0-1 loss, even if the random labeling only differs from $\bar{h}$ on one element of $x_{\pi,\bar{h}}$, it is counted as an error.

To overcome this disadvantage, we formulate a new objective function that can handle an arbitrary general loss function $L : \mathcal{Y}^X \times \mathcal{Y}^X \to \mathbb{R}_{\geq 0}$ satisfying the following two properties: $L(\bar{h}, \bar{h}') = L(\bar{h}', \bar{h})$ for any two labelings $\bar{h}$ and $\bar{h}'$ of X, and $L(\bar{h}, \bar{h}') = 0$ for any $\bar{h} = \bar{h}'$. For $S \subseteq X$ and $\bar{h} \in \overline{\mathcal{H}}$, we define the *generalized version space reduction* function as:

$$f_L(S, \bar{h}) \stackrel{\text{def}}{=} \mathbb{E}_{\bar{h}' \sim \bar{p}_0} \left[L(\bar{h}, \bar{h}') \, \mathbf{1} \left(\bar{h}(S) \neq \bar{h}'(S) \right) \right].$$

This function is the expected loss between the true labeling h and any labeling h' not in the version space. Note that $f_L(S, \bar{h}) = \sum_{\bar{h}' \in \overline{\mathcal{H}}: \bar{h}(S) \neq \bar{h}'(S)} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')$, which can also be written as:

$$\sum_{\bar{h}' \in \overline{\mathcal{H}}} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}') - \sum_{\bar{h}' \in \overline{\mathcal{H}}: \bar{h}(S) = \bar{h}'(S)} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}').$$

If L is the 0-1 loss, i.e., $L(\bar{h}, \bar{h}') = \mathbf{1}(\bar{h} \neq \bar{h}')$, we have $f_{\text{0-1}}(S, \bar{h}) = \sum_{\bar{h}': \bar{h}(S) \neq \bar{h}'(S)} \bar{p}_0[\bar{h}']$, which is equal to the version space reduction function $f(S, \bar{h})$.

Our new objective is to maximize the expected value of the generalized version space reduction:

$$H_L^{\text{avg}}(\pi) \overset{\text{def}}{=} \mathbb{E}_{\bar{h} \sim \bar{p}_0} \left[f_L(x_{\pi, \bar{h}}, \bar{h}) \right].$$

Similar to the above discussion, when L is the 0-1 loss, this objective function is equal to the policy Gibbs error $H_{\text{gibbs}}(\pi)$. Thus, we call $H_L^{\text{avg}}(\pi)$ the *generalized policy Gibbs error*. In the next section, we shall study some properties of a greedy algorithm that attempts to maximize this objective function.

7.1 The Average-case Criterion

To maximize $H_L^{\text{avg}}(\pi)$, a natural algorithm is to greedily maximize the expected value increment of f_L at each step. Specifically, let $\mathcal{D}$ be the previously observed partial labeling, this greedy criterion chooses the next example x^* that satisfies

$$x^* = \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [f_L(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f_L(x_{\mathcal{D}}, \bar{h})], \tag{7.1}$$

where $x_{\mathcal{D}}$ is the domain of $\mathcal{D}$. We call this criterion the *average generalized Gibbs error* criterion.

From the result in Section 4.2, if f_L is adaptive monotone submodular, then using the average generalized Gibbs error criterion above is near-optimal. Theorem 12 below states this result, which is a direct consequence of Theorem 2.

Theorem 12. *Let π_L^{avg} be the adaptive policy in Π_k selecting examples using Equation* (7.1)*, and let π^* be the optimal adaptive policy in Π_k with respect to H_L^{avg}. If f_L is adaptive monotone submodular with respect to the prior $\bar{p}_0$, then*

$$H_L^{avg}(\pi_L^{avg}) > \left(1 - \frac{1}{e}\right) H_L^{avg}(\pi^*).$$

Note that if L is the 0-1 loss, then f_L is adaptive monotone submodular with respect to any prior. Unfortunately, in general, f_L may not be adaptive submodular with respect to a prior $\bar{p}_0$. Theorem 13 below states this result.

Theorem 13. *Let $\bar{p}_0$ be a prior on $\overline{\mathcal{H}}$ with $\bar{p}_0[\bar{h}] > 0$ for all $\bar{h} \in \overline{\mathcal{H}}$. There exists a loss function L such that f_L is not adaptive submodular with respect to $\bar{p}_0$.*

Proof. Fix any two partial labelings $\mathcal{D}$ and $\mathcal{D}'$ where $\mathcal{D}' = \mathcal{D} \cup \mathcal{E}$ with $\mathcal{E} \neq \emptyset$. For any $\mathcal{D}$, let $x_{\mathcal{D}}$ be the domain of $\mathcal{D}$ and $y_{\mathcal{D}} = \mathcal{D}(x_{\mathcal{D}})$. For any $x \in X \setminus x_{\mathcal{D}'}$, we have:

$$\begin{aligned}
\Delta(x|\mathcal{D}) &= \mathbb{E}_{\bar{h}\sim\bar{p}_{\mathcal{D}}}\left[f_L(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f_L(x_{\mathcal{D}}, \bar{h})\right] \\
&= \mathbb{E}_{\bar{h}\sim\bar{p}_{\mathcal{D}}}[\sum_{\bar{h}'(x_{\mathcal{D}})=\bar{h}(x_{\mathcal{D}})} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}') - \sum_{\substack{\bar{h}'(x)=\bar{h}(x),\\ \bar{h}'(x_{\mathcal{D}})=\bar{h}(x_{\mathcal{D}})}} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')] \\
&= \mathbb{E}_{\bar{h}\sim\bar{p}_{\mathcal{D}}}[\sum_{\bar{h}'(x_{\mathcal{D}})=\bar{h}(x_{\mathcal{D}}),\bar{h}'(x)\neq\bar{h}(x)} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')] \\
&= \sum_{\bar{h}:\bar{p}_{\mathcal{D}}[\bar{h}]>0} \left(\bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h}'(x_{\mathcal{D}})=\bar{h}(x_{\mathcal{D}}),\bar{h}'(x)\neq\bar{h}(x)} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')\right).
\end{aligned}$$

Note that if $\bar{p}_{\mathcal{D}}[\bar{h}] > 0$, then $\bar{p}_{\mathcal{D}}[\bar{h}] = \dfrac{\bar{p}_0[\bar{h}]}{\bar{p}_0[y_{\mathcal{D}}; x_{\mathcal{D}}]} = \dfrac{\bar{p}_0[\bar{h}]}{\sum_{\bar{h}:\bar{h}(x_{\mathcal{D}})=y_{\mathcal{D}}} \bar{p}_0[\bar{h}]}$.

Thus, $\Delta(x|\mathcal{D})$

$$= \frac{\sum_{\bar{p}_{\mathcal{D}}[\bar{h}]>0} \sum_{\substack{\bar{p}_{\mathcal{D}}[\bar{h}']>0,\\ \bar{h}'(x)\neq\bar{h}(x)}} \bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')}{\sum_{\bar{h}(x_{\mathcal{D}})=y_{\mathcal{D}}} \bar{p}_0[\bar{h}]} = \frac{\sum_{\bar{h}\sim\mathcal{D}} \sum_{\substack{\bar{h}'\sim\mathcal{D},\\ \bar{h}'(x)\neq\bar{h}(x)}} \bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')}{\sum_{\bar{h}\sim\mathcal{D}} \bar{p}_0[\bar{h}]}.$$

Similarly, for $\mathcal{D}'$, we also have:

$$\begin{aligned}\Delta(x|\mathcal{D}') &= \frac{\sum_{\bar{h}\sim\mathcal{D}'}\sum_{\bar{h}'\sim\mathcal{D}',\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')}{\sum_{\bar{h}\sim\mathcal{D}'}\bar{p}_0[\bar{h}]}\\ &= \frac{1}{\sum_{\bar{h}\sim\mathcal{D}'}\bar{p}_0[\bar{h}]}[\sum_{\bar{h}\sim\mathcal{D}}\sum_{\bar{h}'\sim\mathcal{D},\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')\\ &\qquad -\sum_{\bar{h}\sim\mathcal{D}}\sum_{\bar{h}'\sim\mathcal{D},\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}\nsim\mathcal{E}\vee\bar{h}'\nsim\mathcal{E})],\end{aligned}$$

where $\bar{h}\nsim\mathcal{E}$ denotes that $\bar{h}$ is not consistent with $\mathcal{E}$. Now we can construct the loss function L such that $L(\bar{h},\bar{h}')=0$ for all $\bar{h},\bar{h}'$ satisfying $\bar{h}\nsim\mathcal{E}$ or $\bar{h}'\nsim\mathcal{E}$. Thus,

$$\Delta(x|\mathcal{D}') = \frac{\sum_{\bar{h}\sim\mathcal{D}}\sum_{\bar{h}'\sim\mathcal{D},\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')}{\sum_{\bar{h}\sim\mathcal{D}'}\bar{p}_0[\bar{h}]}.$$

From the assumption $\bar{p}_0[\bar{h}]>0$ for all $\bar{h}$, we have $\sum_{\bar{h}\sim\mathcal{D}'}\bar{p}_0[\bar{h}]<\sum_{\bar{h}\sim\mathcal{D}}\bar{p}_0[\bar{h}]$. Therefore, $\Delta(x|\mathcal{D}')>\Delta(x|\mathcal{D})$ and f_L is not adaptive submodular. □

Sufficient Condition for Adaptive Submodularity of f_L

We now discuss a sufficient condition for f_L to be adaptive submodular with respect to $\bar{p}_0$, and hence satisfy the precondition in Theorem 12 (note that f_L is already adaptive monotone with respect to any prior $\bar{p}_0$). From the previous proof, let

$$A=\sum_{\bar{h}\sim\mathcal{D}}\sum_{\bar{h}'\sim\mathcal{D},\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}'),$$

$$B=\sum_{\bar{h}\sim\mathcal{D}}\sum_{\bar{h}'\sim\mathcal{D},\bar{h}'(x)\neq\bar{h}(x)}\bar{p}_0[\bar{h}]\bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}\nsim\mathcal{E}\vee\bar{h}'\nsim\mathcal{E}),$$

$$C=\sum_{\bar{h}\sim\mathcal{D}}\bar{p}_0[\bar{h}]\qquad\text{and}\qquad D=\sum_{\bar{h}\sim\mathcal{D}}\bar{p}_0[\bar{h}]\,\mathbf{1}(\bar{h}\nsim\mathcal{E}).$$

Now let us allow $\mathcal{E}$ to be possibly empty instead of being just a non-empty set as in the previous proof. Note that $\Delta(x|\mathcal{D})=\frac{A}{C}$ and $\Delta(x|\mathcal{D}')=\frac{A-B}{C-D}$. Thus, a sufficient condition for f_L to be adaptive submodular with respect to $\bar{p}_0$ is that for all $\mathcal{D}$, $\mathcal{D}'$, and x, we have $\frac{A}{C} \geq \frac{A-B}{C-D}$. This condition is equivalent to $\frac{A}{C} \leq \frac{B}{D}$. That means:

$$\begin{aligned} & \frac{\sum_{\bar{h} \sim \mathcal{D}} \sum_{\bar{h}' \sim \mathcal{D}, \bar{h}'(x) \neq \bar{h}(x)} \bar{p}_0[\bar{h}] \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')}{\sum_{\bar{h} \sim \mathcal{D}} \bar{p}_0[\bar{h}]} \\ \leq \quad & \frac{\sum_{\bar{h} \sim \mathcal{D}} \sum_{\bar{h}' \sim \mathcal{D}, \bar{h}'(x) \neq \bar{h}(x)} \bar{p}_0[\bar{h}] \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}') \, \mathbf{1}(\bar{h} \nsim \mathcal{E} \vee \bar{h}' \nsim \mathcal{E})}{\sum_{\bar{h} \sim \mathcal{D}} \bar{p}_0[\bar{h}] \, \mathbf{1}(\bar{h} \nsim \mathcal{E})} \end{aligned}$$

for all $\mathcal{D}$, $\mathcal{D}'$, and x. This condition holds if L is the 0-1 loss. However, it remains open whether this sufficient condition is true for any interesting loss function other than 0-1 loss.

7.2 The Worst-case Criterion

We have shown in Theorem 13 that f_L may not be adaptive submodular, and thus we may not always have a theoretical guarantee for the average generalized Gibbs error criterion. In this section, we will reconsider our objective in the worst case instead of the average case.

In the worst case, we may intuitively want to maximize the worst-case objective function $H_L^{\text{worst}}(\pi) \stackrel{\text{def}}{=} \min_{\bar{h} \in \overline{\mathcal{H}}} f_L(x_{\pi, \bar{h}}, \bar{h})$. However, using this objective function may be too conservative since the generalized version space reduction is computed only from the losses between the surviving labelings[1] and the worst-case labeling. Instead, we propose a less conservative objective function based on the losses among all the surviving labelings. Formally, we define the following *total generalized version space reduction* function:

$$t_L(S, \bar{h}) \stackrel{\text{def}}{=} \sum_{\bar{h}'} \sum_{\bar{h}''} \bar{p}_0[\bar{h}'] L(\bar{h}', \bar{h}'') \bar{p}_0[\bar{h}''] - \sum_{\bar{h}' : \bar{h}'(S) = \bar{h}(S)} \; \sum_{\bar{h}'' : \bar{h}''(S) = \bar{h}(S)} \bar{p}_0[\bar{h}'] L(\bar{h}', \bar{h}'') \bar{p}_0[\bar{h}''].$$

Our new objective is to maximize the following function called the *worst-case total*

[1] The surviving labelings in $f_L(S, \bar{h})$ are the labelings consistent with $\bar{h}$ on S.

generalized policy Gibbs error:

$$T_L^{\text{worst}}(\pi) \overset{\text{def}}{=} \min_{\bar{h} \in \bar{\mathcal{H}}} t_L(x_{\pi,\bar{h}}, \bar{h}).$$

To maximize T_L^{worst}, we propose a greedy algorithm that maximizes the worst-case total generalized version space reduction at every step. Note that $t_L(S, \bar{h})$ satisfies the minimal dependency property, i.e., its value does not depend on the labels of $X \setminus S$ in $\bar{h}$. So, for a partial labeling $\mathcal{D}$, we have $t_L(x_{\mathcal{D}}, \bar{h}) = t_L(x_{\mathcal{D}}, \mathcal{D})$ for any $\bar{h} \sim \mathcal{D}$. Using this notation, the greedy criterion for choosing the next example x^* can be written as:

$$x^* = \arg\max_{x \in X} \left\{ \min_{y \in \mathcal{Y}} \left[t_L(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x, y)\}) - t_L(x_{\mathcal{D}}, \mathcal{D}) \right] \right\}, \qquad (7.2)$$

where $\mathcal{D}$ is the previously observed partial labeling. We call this criterion the *worst-case generalized Gibbs error* criterion. It can be shown that t_L is pointwise monotone submodular and satisfies the minimal dependency property for any loss function L. Furthermore, the criterion in Equation (7.2) is equivalent to the criterion in Equation (4.3). Thus, it follows from Theorem 3 that this greedy criterion is near-optimal with respect to the objective function $T_L^{\text{worst}}(\pi)$. Theorem 14 below proves this result.

Theorem 14. *Let π_L^{worst} be the adaptive policy in Π_k selecting examples using Equation* (7.2)*, and let π^* be the optimal adaptive policy in Π_k with respect to T_L^{worst}. We have:*

$$T_L^{worst}(\pi_L^{worst}) > \left(1 - \frac{1}{e}\right) T_L^{worst}(\pi^*).$$

Proof. It is clear that t_L satisfies the minimal dependency property and Equation (7.2) is equivalent to Equation (4.3). It is also clear that t_L is pointwise monotone and $t_L(\emptyset, \bar{h}) = 0$ for all $\bar{h}$. Thus, to apply Theorem 3, what remains is to show that t_L is pointwise submodular.

Consider $t_{L,\bar{h}}(S) = t_L(S, \bar{h})$ for any $\bar{h}$. Fix $A \subseteq B \subseteq X$ and $x \in X \setminus B$. We have:

$$t_{L,\bar{h}}(A \cup \{x\}) - t_{L,\bar{h}}(A)$$

$$
\begin{aligned}
&= \sum_{\bar{h}'(A)=\bar{h}(A)} \sum_{\bar{h}''(A)=\bar{h}(A)} \bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}''] - \sum_{\substack{\bar{h}'(A)=\bar{h}(A)\\ \bar{h}'(x)=\bar{h}(x)}} \sum_{\substack{\bar{h}''(A)=\bar{h}(A)\\ \bar{h}''(x)=\bar{h}(x)}} \bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}''] \\
&= \sum_{\bar{h}'} \sum_{\bar{h}''} (\bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}'']\,\mathbf{1}(\bar{h}'(A)=\bar{h}(A) \wedge \bar{h}''(A)=\bar{h}(A)) \times \\
&\qquad\qquad\qquad\qquad \mathbf{1}(\bar{h}'(x) \neq \bar{h}(x) \vee \bar{h}''(x) \neq \bar{h}(x))).
\end{aligned}
$$

Similarly, we have: $t_{L,\bar{h}}(B \cup \{x\}) - t_{L,\bar{h}}(B)$

$$
\begin{aligned}
&= \sum_{\bar{h}'} \sum_{\bar{h}''} (\bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}'']\,\mathbf{1}(\bar{h}'(B)=\bar{h}(B) \wedge \bar{h}''(B)=\bar{h}(B)) \times \\
&\qquad\qquad\qquad\qquad \mathbf{1}(\bar{h}'(x) \neq \bar{h}(x) \vee \bar{h}''(x) \neq \bar{h}(x))).
\end{aligned}
$$

Since $A \subseteq B$, all pairs $\bar{h}, \bar{h}'$ such that $\mathbf{1}(\bar{h}'(B) = \bar{h}(B) \wedge \bar{h}''(B) = \bar{h}(B)) = 1$ also satisfy $\mathbf{1}(\bar{h}'(A) = \bar{h}(A) \wedge \bar{h}''(A) = \bar{h}(A)) = 1$. Thus, $t_{L,\bar{h}}(A \cup \{x\}) - t_{L,\bar{h}}(A) \geq t_{L,\bar{h}}(B \cup \{x\}) - t_{L,\bar{h}}(B)$ and hence $t_{L,\bar{h}}$ is submodular. Therefore, t_L is pointwise submodular. □

It is worth noting that, like t_L, the function f_L is also pointwise submodular for any loss function L. The proof for the pointwise submodularity of f_L is essentially similar to the proofs that f and t_L are pointwise submodular in Theorem 6 and 14. Proposition 1 below proves this claim.

Proposition 1. *For any prior $\bar{p}_0$ and any loss function L, the generalized version space reduction function f_L is pointwise submodular.*

Proof. Consider $f_{L,\bar{h}}(S) = f_L(S, \bar{h})$ for any $\bar{h}$. Fix $A \subseteq B \subseteq X$ and $x \in X \setminus B$. We have:

$$
\begin{aligned}
f_{L,\bar{h}}(A \cup \{x\}) - f_{L,\bar{h}}(A) &= \sum_{\bar{h}'(A)=\bar{h}(A)} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}') - \sum_{\bar{h}'(A)=\bar{h}(A),\bar{h}'(x)=\bar{h}(x)} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}') \\
&= \sum_{\bar{h}'} \bar{p}_0[\bar{h}']L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}'(A)=\bar{h}(A))\,\mathbf{1}(\bar{h}'(x) \neq \bar{h}(x)).
\end{aligned}
$$

Similarly, we have

$$f_{L,\bar{h}}(B \cup \{x\}) - f_{L,\bar{h}}(B) = \sum_{\bar{h}'} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')\, \mathbf{1}(\bar{h}'(B) = \bar{h}(B))\, \mathbf{1}(\bar{h}'(x) \neq \bar{h}(x)).$$

Since $A \subseteq B$, all pairs $\bar{h}, \bar{h}'$ such that $\bar{h}'(B) = \bar{h}(B)$ also satisfy $\bar{h}'(A) = \bar{h}(A)$. Thus, $f_{L,\bar{h}}(A \cup \{x\}) - f_{L,\bar{h}}(A) \geq f_{L,\bar{h}}(B \cup \{x\}) - f_{L,\bar{h}}(B)$ and hence $f_{L,\bar{h}}$ is submodular. Therefore, f_L is pointwise submodular. □

However, we note that f_L does not satisfy the minimal dependency property, so we cannot use the theory in Section 4.3 directly on f_L. Besides, Theorem 13 also shows that f_L may not be adaptive submodular. Thus, this is an example that a pointwise submodular function is not necessarily adaptive submodular, and we may not be able to use Golovin and Krause (2011)'s result to obtain a result in the average case for pointwise submodular functions.

7.3 Computing the Criteria

In this section, we discuss the computations of the criteria in Equation (7.1) and Equation (7.2). First, we give two propositions below regarding these equations.

Proposition 2. *The selected example* x^* *in Equation* (7.1) *is equal to*

$$\arg\min_{x \in X} \sum_{y \in \mathcal{Y}} \mathbb{E}_{\bar{h}, \bar{h}' \sim \bar{p}_{\mathcal{D}}} \left[L(\bar{h}, \bar{h}')\, \mathbf{1}(\bar{h}(x) = \bar{h}'(x) = y) \right].$$

Proof. From Equation (7.1) and the definition of f_L, we have:

$$\begin{aligned} x^* &= \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} \left[f_L(x_{\mathcal{D}} \cup \{x\}, \bar{h}) - f_L(x_{\mathcal{D}}, \bar{h}) \right] \\ &= \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} \left[f_L(x_{\mathcal{D}} \cup \{x\}, \bar{h}) \right] \\ &= \arg\max_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [\sum_{\bar{h}'} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}') - \sum_{\bar{h}' : \bar{h}(x_{\mathcal{D}}) = \bar{h}'(x_{\mathcal{D}}), \bar{h}(x) = \bar{h}'(x)} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')] \end{aligned}$$

$$
\begin{aligned}
&= \arg\min_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [\sum_{\bar{h}' : \bar{h}(x_{\mathcal{D}}) = \bar{h}'(x_{\mathcal{D}}), \bar{h}(x) = \bar{h}'(x)} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')] \\
&= \arg\min_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [\sum_{\bar{h}' : \bar{p}_{\mathcal{D}}[\bar{h}'] > 0, \bar{h}(x) = \bar{h}'(x)} \bar{p}_0[\bar{h}'] L(\bar{h}, \bar{h}')].
\end{aligned}
$$

Note that if $\bar{p}_{\mathcal{D}}[\bar{h}'] > 0$, then $\bar{p}_0[\bar{h}'] = \bar{p}_{\mathcal{D}}[\bar{h}'] \bar{p}_0[y_{\mathcal{D}}; x_{\mathcal{D}}]$. Hence, the last expression above is equal to:

$$
\begin{aligned}
& \arg\min_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [\sum_{\bar{h}' : \bar{p}_{\mathcal{D}}[\bar{h}'] > 0, \bar{h}(x) = \bar{h}'(x)} \bar{p}_{\mathcal{D}}[\bar{h}'] \bar{p}_0[y_{\mathcal{D}}; x_{\mathcal{D}}] L(\bar{h}, \bar{h}')] \\
=\ & \arg\min_{x \in X} \mathbb{E}_{\bar{h} \sim \bar{p}_{\mathcal{D}}} [\sum_{\bar{h}' : \bar{p}_{\mathcal{D}}[\bar{h}'] > 0, \bar{h}(x) = \bar{h}'(x)} \bar{p}_{\mathcal{D}}[\bar{h}'] L(\bar{h}, \bar{h}')] \\
=\ & \arg\min_{x \in X} \sum_{\bar{h}} \bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h}' : \bar{h}(x) = \bar{h}'(x)} \bar{p}_{\mathcal{D}}[\bar{h}'] L(\bar{h}, \bar{h}') \\
=\ & \arg\min_{x \in X} \sum_{y \in \mathcal{Y}} \sum_{\bar{h} : \bar{h}(x) = y} \bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h}' : \bar{h}'(x) = y} \bar{p}_{\mathcal{D}}[\bar{h}'] L(\bar{h}, \bar{h}') \\
=\ & \arg\min_{x \in X} \sum_{y \in \mathcal{Y}} \sum_{\bar{h}} \bar{p}_{\mathcal{D}}[\bar{h}] \sum_{\bar{h}'} \bar{p}_{\mathcal{D}}[\bar{h}'] L(\bar{h}, \bar{h}')\, \mathbf{1}(\bar{h}(x) = \bar{h}'(x) = y) \\
=\ & \arg\min_{x \in X} \sum_{y \in \mathcal{Y}} \mathbb{E}_{\bar{h}, \bar{h}' \sim \bar{p}_{\mathcal{D}}} \left[L(\bar{h}, \bar{h}')\, \mathbf{1}(\bar{h}(x) = \bar{h}'(x) = y) \right].
\end{aligned}
$$

Thus, Proposition 2 holds. □

Proposition 3. *The selected example* x^* *in Equation* (7.2) *is equal to*

$$
\arg\min_{x \in X} \left\{ \max_{y \in \mathcal{Y}} \mathbb{E}_{\bar{h}, \bar{h}' \sim \bar{p}_{\mathcal{D}}} \left[L(\bar{h}, \bar{h}')\, \mathbf{1}(\bar{h}(x) = \bar{h}'(x) = y) \right] \right\}.
$$

Proof. From Equation (7.2) and the definition of t_L, we have:

$$
\begin{aligned}
x^* &= \arg\max_{x \in X} \min_{y \in \mathcal{Y}} [t_L(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x, y)\}) - t_L(x_{\mathcal{D}}, \mathcal{D})] \\
&= \arg\max_{x \in X} \min_{y \in \mathcal{Y}} [t_L(x_{\mathcal{D}} \cup \{x\}, \mathcal{D} \cup \{(x, y)\})] \\
&= \arg\max_{x \in X} \min_{y \in \mathcal{Y}} [\sum_{\bar{h}'} \sum_{\bar{h}''} \bar{p}_0[\bar{h}'] L(\bar{h}', \bar{h}'') \bar{p}_0[\bar{h}''] \\
&\qquad - \sum_{\substack{\bar{h}'(x_{\mathcal{D}}) = y_{\mathcal{D}}, \\ \bar{h}'(x) = y}} \sum_{\substack{\bar{h}''(x_{\mathcal{D}}) = y_{\mathcal{D}}, \\ \bar{h}''(x) = y}} \bar{p}_0[\bar{h}'] L(\bar{h}', \bar{h}'') \bar{p}_0[\bar{h}'']]
\end{aligned}
$$

$$\begin{aligned}
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{h}'(x_{\mathcal{D}})=y_{\mathcal{D}},\bar{h}'(x)=y} \quad \sum_{\bar{h}''(x_{\mathcal{D}})=y_{\mathcal{D}},\bar{h}''(x)=y} \bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}''] \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{p}_{\mathcal{D}}[\bar{h}']>0,\bar{h}'(x)=y} \quad \sum_{\bar{p}_{\mathcal{D}}[\bar{h}'']>0,\bar{h}''(x)=y} \bar{p}_0[\bar{h}']L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}''] \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{p}_{\mathcal{D}}[\bar{h}']>0,\bar{h}'(x)=y} \bar{p}_0[\bar{h}'] \sum_{\bar{p}_{\mathcal{D}}[\bar{h}'']>0,\bar{h}''(x)=y} L(\bar{h}',\bar{h}'')\bar{p}_0[\bar{h}''].
\end{aligned}$$

Using the same observation about $\bar{p}_0[\bar{h}']$ and $\bar{p}_0[\bar{h}'']$ as in the previous proof, we note that the last expression above is equal to:

$$\begin{aligned}
&\arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\substack{\bar{p}_{\mathcal{D}}[\bar{h}']>0,\\ \bar{h}'(x)=y}} (\bar{p}_{\mathcal{D}}[\bar{h}']\bar{p}_0[y_{\mathcal{D}};x_{\mathcal{D}}] \sum_{\substack{\bar{p}_{\mathcal{D}}[\bar{h}'']>0,\\ \bar{h}''(x)=y}} L(\bar{h}',\bar{h}'')\bar{p}_{\mathcal{D}}[\bar{h}'']\bar{p}_0[y_{\mathcal{D}};x_{\mathcal{D}}]) \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{p}_{\mathcal{D}}[\bar{h}']>0,\bar{h}'(x)=y} \bar{p}_{\mathcal{D}}[\bar{h}'] \sum_{\bar{p}_{\mathcal{D}}[\bar{h}'']>0,\bar{h}''(x)=y} L(\bar{h}',\bar{h}'')\bar{p}_{\mathcal{D}}[\bar{h}''] \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{h}'(x)=y} \bar{p}_{\mathcal{D}}[\bar{h}'] \sum_{\bar{h}''(x)=y} L(\bar{h}',\bar{h}'')\bar{p}_{\mathcal{D}}[\bar{h}''] \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \sum_{\bar{h}'} \bar{p}_{\mathcal{D}}[\bar{h}'] \sum_{\bar{h}''} \bar{p}_{\mathcal{D}}[\bar{h}'']L(\bar{h}',\bar{h}'')\,\mathbf{1}(\bar{h}''(x)=\bar{h}'(x)=y) \\
&= \arg\min_{x\in X}\max_{y\in\mathcal{Y}} \mathbb{E}_{\bar{h}',\bar{h}''\sim\bar{p}_{\mathcal{D}}}\left[L(\bar{h}',\bar{h}'')\,\mathbf{1}(\bar{h}''(x)=\bar{h}'(x)=y)\right].
\end{aligned}$$

Thus, Proposition 3 holds. □

From these two propositions, we can compute Equation (7.1) and Equation (7.2) by estimating the expectation $\mathbb{E}_{\bar{h},\bar{h}'\sim\bar{p}_{\mathcal{D}}}[L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}(x)=\bar{h}'(x)=y)]$ for each $y\in\mathcal{Y}$. This estimation can be done by sampling from the posterior. More specifically, we can sample directly from $\bar{p}_{\mathcal{D}}$ two sets H and H' which contain samples of $\bar{h}$ and $\bar{h}'$ respectively. Then, the expectation $\mathbb{E}_{\bar{h},\bar{h}'\sim\bar{p}_{\mathcal{D}}}[L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}(x)=\bar{h}'(x)=y)]$ can be approximated by:

$$\frac{1}{|H|\times|H'|}\sum_{\bar{h}\in H}\sum_{\bar{h}'\in H'} L(\bar{h},\bar{h}')\,\mathbf{1}(\bar{h}(x)=\bar{h}'(x)=y).$$

Note that this approximation only requires samples of the labelings from the posterior, and we do not need to explicitly maintain the set of all labelings which may be exponentially large. In practice, we usually sample the labelings directly from the posterior of the probabilistic model without explicitly constructing the equivalent deterministic model. We also note that if only one hypothesis (e.g., the MAP hypothesis) is used to sample the labelings, then the criteria in this section would not depend on the loss function. In this case, we cannot get the advantage of using the loss function. Hence, in practice, we should sample the labelings from the whole posterior distribution instead of using only the MAP hypothesis.

7.4 Experiments

Experimental results comparing the maximum entropy criterion, the maximum Gibbs error criterion, and the least confidence criterion were reported in Chapter 6. In this section, we only focus on the active learning criteria with general loss functions, and conduct experiments with 3 different loss functions: the weighted error types loss, the F-score loss, and the weighted test examples loss. These experimental results are reported in Section 7.4.1, 7.4.2, and 7.4.3 respectively.

We experiment with various binary-class tasks from the 20 Newsgroups data set (Joachims, 1996) and the UCI repository (Bache and Lichman, 2013). The 20 Newsgroups text classification tasks are similar to those in Section 6.3.2. In all the experiments, we use the binary-class logistic regression as our model, and compare the active learners using the greedy criteria in Section 7.1 and 7.2 with the passive learner (Pass) and the maximum Gibbs error active learner (Gibbs). We estimate the average-case criteria (AvgL), the worst-case criteria (WorstL), and the Gibbs criterion by sampling from the posterior using the Metropolis-Hastings algorithm (Chib and Greenberg, 1995) with the MAP hypothesis as the initial point. The Metropolis-Hastings algorithm walks randomly around the data manifold and samples 500 labelings for each approximation. The Gibbs criterion is estimated from Equation (6.3), while the criteria with loss functions are estimated using the approximation in Section 7.3. We note that the Gibbs criterion is equivalent to the maximum entropy and the least confidence criteria in this case since the tasks are all binary-class.

We compare the AUCs (area under the curve) for the F-scores or the corresponding loss of the above algorithms (Pass, Gibbs, AvgL, and WorstL). The AUCs are computed from the first 150 examples and normalized so that their ranges are from 0 to 100. We randomly choose the first 10 examples as a seed set. In each run of the experiments, we use the same seed set for all the algorithms. The final results are obtained by averaging over 100 different runs of the algorithms.

The detailed procedure to compute the AUCs for each run is as follows. We sequentially choose 10 (seed size), 11, . . ., 150 training examples using active learning or passive learning. Then for each training size, we train a model and compute its score (F-score or loss) on a separate test set. Using these scores, we can compute the AUCs. We use the AUC scores because we want to compare the whole learning curves from choosing 10 to 150 training examples, not just the scores at any single point (e.g., 150 examples). This is consistent with previous works such as (Settles and Craven, 2008).

It is worth to note that the loss functions used in this section are all imbalanced in the sense that they give different weights for different examples or labels. From our preliminary experiments not reported here, the active learning criteria with a balanced loss function (Hamming loss, F_1 loss, etc.) do not show advantages when compared to the Gibbs criterion, although their performances are comparable. On the other hand, the experiments in this section show that our active learning criteria with loss functions are better when an imbalanced loss is concerned and the data sets are relatively balanced. In all the results in this section, an asterisk (*) indicates that the corresponding score of the active learning algorithm with a loss function is better than Gibbs. Bold figures indicate the best average scores.

Task	Pass	Gibbs	AvgL	WorstL
alt.atheism/comp.graphics	40.68	42.78	42.75*	38.27*
talk.politics.guns/talk.politics.mideast	57.60	56.30	56.40	52.99*
comp.sys.mac.hardware/comp.windows.x	56.07	56.45	56.10*	54.55*
rec.motorcycles/rec.sport.baseball	43.54	44.99	44.52*	41.49*
sci.crypt/sci.electronics	58.05	64.02	63.94*	60.60*
sci.space/soc.religion.christian	57.27	45.37	45.31*	45.16*
soc.religion.christian/talk.politics.guns	47.58	42.77	42.39*	39.98*
Average	51.54	50.38	50.20*	**47.58***

Table 7.1: AUCs (%) for loss with weighted error types on the 20 Newsgroups data set.

Data set	Pass	Gibbs	AvgL	WorstL
Adult (Kohavi, 1996)	62.46	62.74	62.59*	62.58*
Breast cancer (Wolberg and Mangasarian, 1990)	32.23	34.33	34.98	34.96
Diabetes (Smith et al., 1988)	61.40	63.45	64.28	62.95*
Ionosphere (Sigillito et al., 1989)	30.05	28.78	28.34*	27.36*
Liver disorders (Forsyth, 1990)	75.13	77.37	77.21*	76.85*
Mushrooms (Schlimmer, 1987)	70.70	35.00	35.56	34.55*
Sonar (Gorman and Sejnowski, 1988)	74.48	74.27	74.17*	73.61*
Average	58.06	53.71	53.88	**53.26***

Table 7.2: AUCs (%) for loss with weighted error types on UCI data sets.

7.4.1 Experiments with Weighted Error Types Loss

In this set of experiments, we consider a loss function that gives different weights for different error types in the prediction. Specifically, the loss function here is similar to the Hamming loss, except that it gives the weight 10 to false positives (instead of 1 as in the Hamming loss). This type of loss functions is useful for applications where some types of errors are more important than others.

Table 7.1 and 7.2 report the AUCs for this loss on the 20 Newsgroups and UCI data

Task	F_2				$F_{0.5}$			
	Pass	Gibbs	AvgL	WorstL	Pass	Gibbs	AvgL	WorstL
alt.atheism/comp.graphics	82.47	87.19	87.22*	87.28*	85.60	85.94	85.77	87.01*
talk.politics.guns/talk.politics.mideast	76.33	75.49	76.14*	77.99*	76.36	75.99	76.66*	78.18*
comp.sys.mac.hardware/comp.windows.x	71.58	73.26	73.65*	75.34*	75.26	75.31	75.64*	76.47*
rec.motorcycles/rec.sport.baseball	77.83	80.31	81.24*	81.97*	82.02	82.00	82.14*	83.70*
sci.crypt/sci.electronics	67.13	76.01	76.41*	76.45*	72.42	72.32	72.56*	74.28*
sci.space/soc.religion.christian	84.49	83.60	84.61*	86.15*	78.71	83.07	83.20*	83.66*
soc.religion.christian/talk.politics.guns	79.23	80.31	80.62*	82.12*	80.14	81.75	82.04*	82.83*
Average	77.01	79.45	79.98*	**81.04***	78.64	79.48	79.72*	**80.88***

Table 7.3: AUCs (%) for F_2 and $F_{0.5}$ on the 20 Newsgroups data set.

Task	F_2				$F_{0.5}$			
	Pass	Gibbs	AvgL	WorstL	Pass	Gibbs	AvgL	WorstL
Adult (Kohavi, 1996)	85.85	88.56	88.49	88.46	84.52	84.95	85.01*	85.04*
Breast cancer (Wolberg and Mangasarian, 1990)	91.65	91.22	91.45*	91.53*	93.59	93.16	93.22*	93.39*
Diabetes (Smith et al., 1988)	24.09	28.01	27.56	28.90*	33.30	36.29	35.22	37.11*
Ionosphere (Sigillito et al., 1989)	59.45	62.24	62.15	61.86	78.66	80.32	80.71*	80.96*
Liver disorders (Forsyth, 1990)	74.80	77.13	77.34*	76.23	70.92	69.95	70.11*	70.23*
Mushrooms (Schlimmer, 1987)	76.88	91.41	91.38	91.62*	54.31	80.97	80.89	81.37*
Sonar (Gorman and Sejnowski, 1988)	72.74	73.16	72.90	72.85	69.00	69.18	69.20*	69.50*
Average	69.35	**73.10**	73.04	73.06	69.19	73.55	73.48	**73.94***

Table 7.4: AUCs (%) for F_2 and $F_{0.5}$ on UCI data sets.

sets respectively. From the results, all the active learning algorithms perform better than passive learning in terms of the loss. On the 20 Newsgroups data set, AvgL and WorstL perform better than Gibbs on most tasks, and WorstL achieves the best average AUC overall. On the UCI data sets, WorstL performs slightly better than Gibbs and also achieves the best average score.

7.4.2 Experiments with F-scores

In this set of experiments, we consider the F_2 and $F_{0.5}$ loss functions on the 20 Newsgroups and the UCI data sets. For two labelings $\bar{h}$ and $\bar{h}'$ (viewing them as label vectors), the F_β loss is $1 - F_\beta(\bar{h}, \bar{h}')$ where $F_\beta(\bar{h}, \bar{h}') \in [0, 1]$ is the F_β score between $\bar{h}$ and $\bar{h}'$. The F_β score is defined as:

$$F_\beta \stackrel{\text{def}}{=} \frac{(1+\beta^2)PR}{(\beta^2 P) + R},$$

where P and R are respectively the precision and recall of $\bar{h}'$ compared to $\bar{h}$. This loss function is useful when we want to focus on the F_β scores during our learning process. F_2 and $F_{0.5}$ are two commonly used scores (together with F_1) for various applications in information retrieval.

Table 7.3 and 7.4 report the AUCs for F_2 and $F_{0.5}$ scores in this experiment on the 20 Newsgroups and the UCI data sets respectively. From the results, all the active learning algorithms perform better than passive learning in terms of average F-scores. On the 20 Newsgroups data set, for both types of F-scores, AvgL and WorstL perform better than Gibbs on most tasks, and WorstL achieves the best average AUC overall. On the UCI data sets, AvgL and WorstL do not perform better than Gibbs in terms of the F_2 score; however, WorstL performs better than Gibbs for all tasks on the $F_{0.5}$ score and it also achieves the best average AUC for $F_{0.5}$ score overall.

7.4.3 Experiments with Weighted Test Examples Loss

Unlike the previous experiments, in this set of experiments we assume the test examples are given in advance, and some test examples have a significantly more weight than the others. This scenario may happen in applications where different examples are classified for different users, and some users are more important than the others. With all the information, we put a loss function on the test data during training that is essentially similar to Hamming loss but gives higher weights for the important examples. In this experiment, we set the weight 10 for half of the test examples, while the other half have weight 1.

Table 7.5 and 7.6 report the AUCs for this loss on the 20 Newsgroups and UCI data sets respectively. From the results, all the active learning algorithms perform better

Task	Pass	Gibbs	AvgL	WorstL
alt.atheism/comp.graphics	15.54	13.37	13.25*	12.42*
talk.politics.guns/talk.politics.mideast	24.43	24.06	23.57*	22.16*
comp.sys.mac.hardware/comp.windows.x	24.59	23.83	23.39*	22.86*
rec.motorcycles/rec.sport.baseball	17.49	16.83	16.31*	15.21*
sci.crypt/sci.electronics	26.42	25.10	24.64*	23.40*
sci.space/soc.religion.christian	19.17	16.43	16.31*	14.84*
soc.religion.christian/talk.politics.guns	18.50	16.85	16.46*	15.24*
Average	20.88	19.50	19.13*	**18.02***

Table 7.5: AUCs (%) for loss with weighted test examples on the 20 Newsgroups data set.

Data set	Pass	Gibbs	AvgL	WorstL
Adult (Kohavi, 1996)	22.58	20.65	20.61*	20.44*
Breast cancer (Wolberg and Mangasarian, 1990)	9.75	10.45	10.12*	10.13*
Diabetes (Smith et al., 1988)	33.09	32.99	32.88*	32.57*
Ionosphere (Sigillito et al., 1989)	16.30	15.40	15.39*	15.71
Liver disorders (Forsyth, 1990)	32.57	32.05	31.92*	32.61
Mushrooms (Schlimmer, 1987)	28.30	12.01	12.05	11.69*
Sonar (Gorman and Sejnowski, 1988)	38.35	39.14	38.79*	38.74*
Average	25.85	23.24	**23.11***	23.13*

Table 7.6: AUCs (%) for loss with weighted test examples on UCI data sets.

than passive learning in terms of the loss. On the 20 Newsgroups data set, AvgL and WorstL perform better than Gibbs on all tasks, and WorstL achieves the best average AUC overall. On the UCI data sets, both AvgL and WorstL perform slightly better than Gibbs, and AvgL achieves the best average score overall.

CHAPTER 8

Robustness of Bayesian Pool-based Active Learning

In the previous chapters, we considered Bayesian pool-based active learning with the assumption that the true prior is known and given to the active learner. In practice, however, the prior is often unknown, and we choose a prior that is considered to be close to the true prior. In this chapter, we shall analyze the robustness of active learning algorithms in such setting where a perturbed prior is used instead of the true prior. In particular, we investigate whether an active learning algorithm can achieve similar performance using a perturbed prior as compared to using the true prior. We shall also describe the use of a mixture prior model for more robust active learning when the prior is unknown and then conduct experiments to show that the mixture prior model is robust in practice.

8.1 Robustness of Active Learning Algorithms

In this section, we analyze the robustness of active learning algorithms that use a perturbed prior. We recall from the previous chapters that the utility function $f(S, \bar{h})$ measures the value of querying examples $S \subseteq X$ when the true labeling is $\bar{h} \in \overline{\mathcal{H}}$. The utility function may depend on the prior, but such dependency was omitted from the notation as it was assumed that the true prior is always used. However, in this chapter, the learning algorithm may use a perturbed prior $\bar{p}$ which is different from the true prior; thus we shall use the notation $f_{\bar{p}}(S, \bar{h})$ to make the dependency explicit. Similar to the previous chapters, we also consider the maximum coverage problem in this section.

In our analysis, we will need the following definition about the Lipschitz continuity of the utility functions:

Definition. A utility function $f_{\bar{p}}$ is said to be *Lipschitz continuous (in the prior)* with a Lipschitz constant L, if for any S, any $\bar{h}$, and any two priors $\bar{p}$ and $\bar{p}'$,

$$|f_{\bar{p}}(S,\bar{h}) - f_{\bar{p}'}(S,\bar{h})| \leq L\|\bar{p} - \bar{p}'\|, \tag{8.1}$$

where $\|\bar{p} - \bar{p}'\| \overset{\text{def}}{=} \sum_{\bar{h}} |\bar{p}[\bar{h}] - \bar{p}'[\bar{h}]|$ is the ℓ_1 distance between $\bar{p}$ and $\bar{p}'$.

In this chapter, an active learning algorithm A is a mapping from a utility function and a prior to a policy. We use $x_{\pi,\bar{h}}$ to denote the set of examples selected by a policy π when the true labeling is $\bar{h}$. We will analyze the robustness of adaptive active learning algorithms for the average case and the worst case separately in the following sections.

8.1.1 The Average Case

In the average case, the objective of an active learning algorithm is to find a policy with maximum expected utility. Assume $\bar{p}_0$ is the true prior, then the expected utility of a policy π is:

$$f^{\text{avg}}_{\bar{p}_0}(\pi) \overset{\text{def}}{=} \mathbb{E}_{\bar{h}\sim\bar{p}_0}\left[f_{\bar{p}_0}(x_{\pi,\bar{h}}, \bar{h})\right]. \tag{8.2}$$

We consider the case where we have already chosen a utility function, but still need to choose the prior. In practice, the choice is often subjective and may not be the true prior. A natural question is if we choose a *perturbed* prior $\bar{p}_1$, that is, a prior which is not very different from the true prior $\bar{p}_0$ (in terms of ℓ_1 distance), can an active learning algorithm achieve performance competitive to that obtained using the true prior? Note that we focus on perturbed priors, because if it is possible to achieve close to best performance using any prior, then the problem is likely to be easy to solve approximately.

Our first robustness result is on exact algorithms which return a policy with maximum expected utility on any given prior. An *exact* algorithm A for the average-case maximum coverage problem is an algorithm that outputs an optimal policy for any prior $\bar{p}$:

$$A(\bar{p}) = \arg\max_{\pi} f^{\text{avg}}_{\bar{p}}(\pi). \tag{8.3}$$

For notational convenience, we drop the dependency of the algorithm A on the utility function as we assume a fixed utility function here. We shall often need the following property:

The utility function $f_{\bar{p}}$ is upper bounded by a constant M and Lipschitz continuous with a Lipschitz constant L. (*)

We now prove the following theorem for exact algorithms.

Theorem 15. *Assume (*) holds. If A is an exact algorithm for the average-case maximum coverage problem, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{avg}_{\bar{p}_0}(A(\bar{p}_1)) \geq f^{avg}_{\bar{p}_0}(A(\bar{p}_0)) - 2(L+M)\|\bar{p}_1 - \bar{p}_0\|.$$

Thus, A is robust in the sense that it returns a near optimal policy when using a perturbed prior.

Proof. For any policy π, note that:

$$\begin{aligned} |f^{\text{avg}}_{\bar{p}_0}(\pi) - f^{\text{avg}}_{\bar{p}_1}(\pi)| &= |(\sum_{\bar{h}} \bar{p}_0[\bar{h}] f_{\bar{p}_0}(x_{\pi,\bar{h}}, \bar{h}) - \sum_{\bar{h}} \bar{p}_0[\bar{h}] f_{\bar{p}_1}(x_{\pi,\bar{h}}, \bar{h})) \\ &\quad +(\sum_{\bar{h}} \bar{p}_0[\bar{h}] f_{\bar{p}_1}(x_{\pi,\bar{h}}, \bar{h}) - \sum_{\bar{h}} \bar{p}_1[\bar{h}] f_{\bar{p}_1}(x_{\pi,\bar{h}}, \bar{h}))| \\ &\leq (L+M)\|\bar{p}_0 - \bar{p}_1\|, \end{aligned}$$

where the last inequality holds due to the Lipschitz continuity and boundedness of the utility function $f_{\bar{p}}$. Thus, if $\pi_1 = A(\bar{p}_1)$ and $\pi_0 = A(\bar{p}_0)$, it follows that:

$$f^{\text{avg}}_{\bar{p}_0}(\pi_1) \geq f^{\text{avg}}_{\bar{p}_1}(\pi_1) - (L+M)\|\bar{p}_0 - \bar{p}_1\|, \text{ and}$$

$$f^{\text{avg}}_{\bar{p}_1}(\pi_1) \;\; \geq \;\; f^{\text{avg}}_{\bar{p}_1}(\pi_0) \geq f^{\text{avg}}_{\bar{p}_0}(\pi_0) - (L+M)\|\bar{p}_0 - \bar{p}_1\|.$$

Hence, $f^{\text{avg}}_{\bar{p}_0}(\pi_1) \geq f^{\text{avg}}_{\bar{p}_0}(\pi_0) - 2(L+M)\|\bar{p}_0 - \bar{p}_1\|$. □

Note that $f^{\text{avg}}_{\bar{p}_0}(A(\bar{p}_0))$ and $f^{\text{avg}}_{\bar{p}_0}(A(\bar{p}_1))$ are the expected utility of the policies returned by A using $\bar{p}_0$ and $\bar{p}_1$ as the priors respectively. The expected utility is always computed with respect to the true prior $\bar{p}_0$. Theorem 15 shows that when we use a perturbed prior $\bar{p}_1$, the expected utility achieved by an exact algorithm degrades by at most a constant factor of the ℓ_1 distance between the perturbed prior and the true prior.

Theorem 15 considers the robustness of exact algorithms, while practical algorithms are generally approximate due to computational intractability of the problem. Formally, an α-*approximate* ($0 < \alpha \leq 1$) algorithm A for the average-case maximum coverage problem is an algorithm that, for any prior $\bar{p}$, outputs a policy $A(\bar{p})$ satisfying

$$f^{\text{avg}}_{\bar{p}}(A(\bar{p})) \geq \alpha \max_{\pi} f^{\text{avg}}_{\bar{p}}(\pi). \tag{8.4}$$

When $\alpha = 1$, an α-approximate algorithm is an exact algorithm. The following robustness result for α-approximate algorithms is a generalization of Theorem 15 above.

Theorem 16. *Assume (*) holds. If A is an α-approximate algorithm for the average-case maximum coverage problem, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{avg}_{\bar{p}_0}(A(\bar{p}_1)) \geq \alpha \max_{\pi} f^{avg}_{\bar{p}_0}(\pi) - (\alpha+1)(L+M)\|\bar{p}_1 - \bar{p}_0\|.$$

Thus, A is robust in the sense that it returns a near α-approximate policy when using a perturbed prior.

Proof. Let $C = L + M$. Denote $\pi_0 = \arg\max_{\pi} f^{\text{avg}}_{\bar{p}_0}(\pi)$ and $\pi_1 = \arg\max_{\pi} f^{\text{avg}}_{\bar{p}_1}(\pi)$. Note that π_0 and π_1 are exactly those in the proof of Theorem 15. We have:

$$f^{\text{avg}}_{\bar{p}_0}(A(\bar{p}_1)) \;\; \geq \;\; f^{\text{avg}}_{\bar{p}_1}(A(\bar{p}_1)) - C\|\bar{p}_0 - \bar{p}_1\|$$

$$
\begin{aligned}
&\geq \alpha f^{\text{avg}}_{\bar{p}_1}(\pi_1) - C\|\bar{p}_0 - \bar{p}_1\| \\
&\geq \alpha(f^{\text{avg}}_{\bar{p}_0}(\pi_0) - C\|\bar{p}_0 - \bar{p}_1\|) - C\|\bar{p}_0 - \bar{p}_1\| \\
&= \alpha \max_{\pi} f^{\text{avg}}_{\bar{p}_0}(\pi) - C(\alpha+1)\|\bar{p}_0 - \bar{p}_1\|,
\end{aligned}
$$

where the first and third inequalities are from the proof of Theorem 15 and the second inequality holds as A is α-approximate. □

Application to Maximum Gibbs Error

Theorem 16 can be used to prove the robustness of the maximum Gibbs error algorithm in Section 6.1.2 of Chapter 6. We recall that the maximum Gibbs error algorithm greedily selects the next example x^* satisfying the criterion $x^* = \arg\max_x \mathbb{E}_{y \sim p_{\mathcal{D}}[\,\cdot\,;x]}[1 - p_{\mathcal{D}}[y;x]]$, where $p_{\mathcal{D}}$ is the current posterior and $p_{\mathcal{D}}[y;x]$ is the probability (with respect to $p_{\mathcal{D}}$) that x has label y (see Equation (6.3)).

In Section 6.1.2, we have shown that using the maximum Gibbs error criterion above can achieve a constant factor approximation to the optimal policy Gibbs error, which is equivalent to the expected version space reduction. Recall from Equation (5.3) that the version space reduction utility function is $f_{\bar{p}}(S, \bar{h}) = 1 - \bar{p}[\bar{h}(S); S]$, where $\bar{p}[\bar{h}(S); S]$ is the probability (with respect to $\bar{p}$) that S has the labels $\bar{h}(S)$. In this case, the expected utility $f^{\text{avg}}_{\bar{p}}(\pi)$ is the policy Gibbs error in Equation (6.1). It was shown in Theorem 10 that, for any prior $\bar{p}$,

$$
f^{\text{avg}}_{\bar{p}}(A(\bar{p})) \geq \left(1 - \frac{1}{e}\right) \max_{\pi} f^{\text{avg}}_{\bar{p}}(\pi),
$$

where A is the maximum Gibbs error algorithm. That is, the maximum Gibbs error algorithm is $(1 - 1/e)$-approximate.

Furthermore, the version space reduction utility is upper bounded by $M = 1$, and for any priors $\bar{p}$, $\bar{p}'$, we also have:

$$
|f_{\bar{p}}(S, \bar{h}) - f_{\bar{p}'}(S, \bar{h})|
$$

$$\begin{aligned} &= |\bar{p}'[\bar{h}(S);S] - \bar{p}[\bar{h}(S);S]| \\ &= |\sum_{\bar{h}'} \bar{p}'[\bar{h}'] \, \mathbb{P}[\bar{h}'(S) = \bar{h}(S)|\bar{h}'] - \sum_{\bar{h}'} \bar{p}[\bar{h}'] \, \mathbb{P}[\bar{h}'(S) = \bar{h}(S)|\bar{h}']| \\ &\leq \|\bar{p} - \bar{p}'\|. \end{aligned}$$

Thus, the version space reduction utility satisfies (*) with $L = M = 1$. Hence, we have the following corollary about the robustness of the maximum Gibbs error algorithm.

Corollary 1. *If A is the maximum Gibbs error algorithm, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{avg}_{\bar{p}_0}(A(\bar{p}_1)) \geq (1 - \frac{1}{e}) \max_{\pi} f^{avg}_{\bar{p}_0}(\pi) - (4 - \frac{2}{e}) \|\bar{p}_0 - \bar{p}_1\|.$$

Application to Batch Maximum Gibbs Error

We can also obtain a similar result for the batch version of the maximum Gibbs error active learning algorithm in Section 6.1.3 of Chapter 6. We recall that in the batch mode setting, the active learning algorithm selects a batch of examples in each iteration instead of only one example (Hoi et al., 2006b). The batch version of the maximum Gibbs error algorithm is described in Algorithm 6.1, and it is a $(1 - e^{-(e-1)/e})$-approximate algorithm by Theorem 11.

If we restrict the policies to only those in the batch mode setting, then from Theorem 16, we have the following corollary about the batch version of the maximum Gibbs error algorithm. Note that the range of the $\max$ operator in the theorem is restricted to only batch policies.

Corollary 2. *If A is the batch maximum Gibbs error algorithm, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{avg}_{\bar{p}_0}(A(\bar{p}_1)) \geq (1 - e^{-(e-1)/e}) \max_{\pi} f^{avg}_{\bar{p}_0}(\pi) - (4 - 2e^{-(e-1)/e}) \|\bar{p}_0 - \bar{p}_1\|.$$

8.1.2 The Worst Case

In the worst case, the objective of an active learning algorithm is to find a policy with maximum worst-case utility. Assume $\bar{p}_0$ is the true prior, then the worst-case utility of a policy π is:

$$f^{\text{worst}}_{\bar{p}_0}(\pi) \stackrel{\text{def}}{=} \min_{\bar{h} \in \bar{\mathcal{H}}} \left[f_{\bar{p}_0}(x_{\pi,\bar{h}}, \bar{h}) \right].$$

An algorithm A is an *exact* algorithm for the worst-case maximum coverage problem if for any prior $\bar{p}$,

$$A(\bar{p}) = \arg\max_{\pi} f^{\text{worst}}_{\bar{p}}(\pi). \tag{8.5}$$

For exact algorithms, we can obtain a robustness result similar to Theorem 15, but without requiring $f_{\bar{p}}$ to be upper bounded.

Theorem 17. *Assume $f_{\bar{p}}$ is Lipschitz continuous with a Lipschitz constant L. If A is an exact algorithm for the worst-case maximum coverage problem, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{\textit{worst}}_{\bar{p}_0}(A(\bar{p}_1)) \geq f^{\textit{worst}}_{\bar{p}_0}(A(\bar{p}_0)) - 2L\|\bar{p}_1 - \bar{p}_0\|.$$

Thus, A is robust in the sense that it returns a near optimal policy when using a perturbed prior.

Proof. Let $\pi_0 = A(\bar{p}_0)$ and $\pi_1 = A(\bar{p}_1)$. By the definitions of π_1 and $f^{\text{worst}}_{\bar{p}_1}$, we have $f^{\text{worst}}_{\bar{p}_1}(\pi_1) \geq f^{\text{worst}}_{\bar{p}_1}(\pi_0) = \min_{\bar{h}} f_{\bar{p}_1}(x_{\pi_0,\bar{h}}, \bar{h})$. Let $\bar{h}_0 = \arg\min_{\bar{h}} f_{\bar{p}_1}(x_{\pi_0,\bar{h}}, \bar{h})$, the inequality above becomes $f^{\text{worst}}_{\bar{p}_1}(\pi_1) \geq f_{\bar{p}_1}(x_{\pi_0,\bar{h}_0}, \bar{h}_0)$. Using the Lipschitz continuity of $f_{\bar{p}}$ and the definition of $f^{\text{worst}}_{\bar{p}_0}$, we have:

$$\begin{aligned} f_{\bar{p}_1}(x_{\pi_0,\bar{h}_0}, \bar{h}_0) &\geq f_{\bar{p}_0}(x_{\pi_0,\bar{h}_0}, \bar{h}_0) - L\|\bar{p}_0 - \bar{p}_1\| \\ &\geq \min_{\bar{h}} f_{\bar{p}_0}(x_{\pi_0,\bar{h}}, \bar{h}) - L\|\bar{p}_0 - \bar{p}_1\| \\ &= f^{\text{worst}}_{\bar{p}_0}(\pi_0) - L\|\bar{p}_0 - \bar{p}_1\|. \end{aligned}$$

Thus, $f^{\text{worst}}_{\bar{p}_1}(\pi_1) \geq f^{\text{worst}}_{\bar{p}_0}(\pi_0) - L\|\bar{p}_0 - \bar{p}_1\|$.

Similarly, let $\bar{h}_1 = \arg\min_{\bar{h}} f_{\bar{p}_0}(x_{\pi_1,\bar{h}}, \bar{h})$, we also have $f^{\text{worst}}_{\bar{p}_0}(\pi_1) = \min_{\bar{h}} f_{\bar{p}_0}(x_{\pi_1,\bar{h}}, \bar{h}) = f_{\bar{p}_0}(x_{\pi_1,\bar{h}_1}, \bar{h}_1)$. Using the Lipschitz continuity of $f_{\bar{p}}$ and the definition of $f^{\text{worst}}_{\bar{p}_1}$, we have:

$$\begin{aligned} f_{\bar{p}_0}(x_{\pi_1,\bar{h}_1}, \bar{h}_1) &\geq f_{\bar{p}_1}(x_{\pi_1,\bar{h}_1}, \bar{h}_1) - L\|\bar{p}_0 - \bar{p}_1\| \\ &\geq \min_{\bar{h}} f_{\bar{p}_1}(x_{\pi_1,\bar{h}}, \bar{h}) - L\|\bar{p}_0 - \bar{p}_1\| \\ &= f^{\text{worst}}_{\bar{p}_1}(\pi_1) - L\|\bar{p}_0 - \bar{p}_1\|. \end{aligned}$$

Thus, $f^{\text{worst}}_{\bar{p}_0}(\pi_1) \geq f^{\text{worst}}_{\bar{p}_1}(\pi_1) - L\|\bar{p}_0 - \bar{p}_1\| \geq f^{\text{worst}}_{\bar{p}_0}(\pi_0) - 2L\|\bar{p}_0 - \bar{p}_1\|$. □

Similar to Theorem 15, the worst-case utility is always computed with respect to the true prior $\bar{p}_0$. Thus, the left-hand side of the inequality in Theorem 17 is $f^{\text{worst}}_{\bar{p}_0}(A(\bar{p}_1))$ instead of $f^{\text{worst}}_{\bar{p}_1}(A(\bar{p}_1))$. Theorem 17 shows that when we use a perturbed prior $\bar{p}_1$, the worst-case utility achieved by an exact algorithm degrades by at most a constant factor of the ℓ_1 distance between the perturbed prior and the true prior.

We now consider approximate algorithms for the worst case. An algorithm A is an *α-approximate* ($0 < \alpha \leq 1$) algorithm for the worst-case maximum coverage problem if for any prior $\bar{p}$,

$$f^{\text{worst}}_{\bar{p}}(A(\bar{p})) \geq \alpha \max_{\pi} f^{\text{worst}}_{\bar{p}}(\pi). \tag{8.6}$$

For approximate algorithms, we can obtain a robustness result similar to Theorem 16.

Theorem 18. *Assume $f_{\bar{p}}$ is Lipschitz continuous with a Lipschitz constant L. If A is an α-approximate algorithm for the worst-case maximum coverage problem, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f^{\textit{worst}}_{\bar{p}_0}(A(\bar{p}_1)) \geq \alpha \max_{\pi} f^{\textit{worst}}_{\bar{p}_0}(\pi) - (\alpha + 1)L\|\bar{p}_1 - \bar{p}_0\|.$$

Thus, A is robust in the sense that it returns a near α-approximate policy when using a *perturbed prior.*

Proof. Let $\pi = A(\bar{p}_1)$ and $\bar{h}^* = \arg\min_{\bar{h}} f_{\bar{p}_0}(x_{\pi,\bar{h}}, \bar{h})$. Let $\pi_0 = \arg\max_\pi f^{\text{worst}}_{\bar{p}_0}(\pi)$ and $\pi_1 = \arg\max_\pi f^{\text{worst}}_{\bar{p}_1}(\pi)$. Note that π_0 and π_1 are exactly those in the proof of Theorem 17.

From the definition of $f^{\text{worst}}_{\bar{p}_0}$, we have $f^{\text{worst}}_{\bar{p}_0}(\pi) = \min_{\bar{h}} f_{\bar{p}_0}(x_{\pi,\bar{h}}, \bar{h}) = f_{\bar{p}_0}(x_{\pi,\bar{h}^*}, \bar{h}^*)$. By the Lipschitz continuity of $f_{\bar{p}}$, we have:

$$\begin{aligned} f_{\bar{p}_0}(x_{\pi,\bar{h}^*}, \bar{h}^*) &\geq f_{\bar{p}_1}(x_{\pi,\bar{h}^*}, \bar{h}^*) - L\|\bar{p}_0 - \bar{p}_1\| \\ &\geq \min_{\bar{h}} f_{\bar{p}_1}(x_{\pi,\bar{h}}, \bar{h}) - L\|\bar{p}_0 - \bar{p}_1\| \\ &= f^{\text{worst}}_{\bar{p}_1}(\pi) - L\|\bar{p}_0 - \bar{p}_1\| \\ &\geq \alpha \max_\pi f^{\text{worst}}_{\bar{p}_1}(\pi) - L\|\bar{p}_0 - \bar{p}_1\| \\ &= \alpha f^{\text{worst}}_{\bar{p}_1}(\pi_1) - L\|\bar{p}_0 - \bar{p}_1\|, \end{aligned}$$

where the last inequality holds as A is α-approximate. Using the inequality relating $f^{\text{worst}}_{\bar{p}_1}(\pi_1)$ and $f^{\text{worst}}_{\bar{p}_0}(\pi_0)$ in the proof of Theorem 17, we now have:

$$\begin{aligned} f^{\text{worst}}_{\bar{p}_0}(\pi) &\geq \alpha(f^{\text{worst}}_{\bar{p}_0}(\pi_0) - L\|\bar{p}_0 - \bar{p}_1\|) - L\|\bar{p}_0 - \bar{p}_1\| \\ &= \alpha \max_\pi f^{\text{worst}}_{\bar{p}_0}(\pi) - (\alpha + 1)L\|\bar{p}_0 - \bar{p}_1\|. \end{aligned}$$

Thus, the theorem holds. □

Application to Least Confidence

Theorem 18 can be used to prove the robustness of the least confidence active learning algorithm (Lewis and Gale, 1994; Culotta and McCallum, 2005) with the perturbed prior. We recall that the least confidence algorithm greedily selects the next example x^* satisfying the criterion $x^* = \arg\min_x \{\max_{y \in \mathcal{Y}} \bar{p}_{\mathcal{D}}[y; x]\}$. In Section 5.2 of Chapter 5, we have shown that using the least confidence criterion can achieve a constant factor approximation to the optimal worst-case version space reduction.

Formally, if $f_{\bar{p}}(S, \bar{h})$ is the version space reduction utility in Section 8.1.1, then $f_{\bar{p}}^{\text{worst}}(\pi)$ is the worst-case version space reduction of π, and it was shown in Theorem 6 that, for any prior $\bar{p}$,

$$f_{\bar{p}}^{\text{worst}}(A(\bar{p})) \geq \left(1 - \frac{1}{e}\right) \max_{\pi} f_{\bar{p}}^{\text{worst}}(\pi),$$

where A is the least confidence algorithm. That is, the least confidence algorithm is $(1 - 1/e)$-approximate.

Since the version space reduction function is Lipschitz continuous with $L = 1$ as shown in Section 8.1.1, we have the following corollary about the robustness of the least confidence active learning algorithm when the perturbed prior is used.

Corollary 3. *If A is the least confidence algorithm, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$f_{\bar{p}_0}^{worst}(A(\bar{p}_1)) \geq (1 - \frac{1}{e}) \max_{\pi} f_{\bar{p}_0}^{worst}(\pi) - (2 - \frac{1}{e}) \|\bar{p}_0 - \bar{p}_1\|.$$

Application to Generalized Maximum Gibbs Error

Theorem 18 can also be used to prove the robustness of the worst-case generalized Gibbs error algorithm (see Section 7.2) with a bounded loss. We recall that the algorithm greedily maximizes the total generalized version space reduction in the worst case. Formally, the total generalized version space reduction function is defined as:

$$t_{\bar{p}}(S, \bar{h}) \overset{\text{def}}{=} \sum_{\substack{\bar{h}', \bar{h}'' : \bar{h}'(S) \neq \bar{h}(S) \text{ or} \\ \bar{h}''(S) \neq \bar{h}(S)}} \bar{p}[\bar{h}']\, L(\bar{h}', \bar{h}'')\, \bar{p}[\bar{h}''],$$

where L is a non-negative loss function between labelings that satisfies $L(\bar{h}, \bar{h}') = L(\bar{h}', \bar{h})$ and $L(\bar{h}, \bar{h}) = 0$ for all $\bar{h}, \bar{h}'$. The worst-case generalized Gibbs error algorithm attempts to greedily maximize $t_{\bar{p}}^{\text{worst}}(\pi) \overset{\text{def}}{=} \min_{\bar{h}} t_{\bar{p}}(x_{\pi, \bar{h}}, \bar{h})$, and it was shown in

Theorem 14 that, for any prior $\bar{p}$,

$$t_{\bar{p}}^{\text{worst}}(A(\bar{p})) \geq \left(1 - \frac{1}{e}\right) \max_{\pi} t_{\bar{p}}^{\text{worst}}(\pi),$$

where A is the worst-case generalized Gibbs error algorithm. That is, the worst-case generalized Gibbs error algorithm is $(1 - 1/e)$-approximate.

If we assume the loss function L is upper bounded by a constant m, then $t_{\bar{p}}$ is Lipschitz continuous with $L = 2m$. Indeed, for any $S, \bar{h}, \bar{p}$, and $\bar{p}'$, we have:

$$\begin{aligned}
& |t_{\bar{p}}(S, \bar{h}) - t_{\bar{p}'}(S, \bar{h})| \\
= \; & |\sum_{\bar{h}'(S) \neq \bar{h}(S) \text{ or } \bar{h}''(S) \neq \bar{h}(S)} L(\bar{h}', \bar{h}'')(\bar{p}[\bar{h}']\bar{p}[\bar{h}''] - \bar{p}'[\bar{h}']\bar{p}'[\bar{h}''])| \\
\leq \; & m \sum_{\bar{h}'(S) \neq \bar{h}(S) \text{ or } \bar{h}''(S) \neq \bar{h}(S)} |\bar{p}[\bar{h}']\bar{p}[\bar{h}''] - \bar{p}'[\bar{h}']\bar{p}'[\bar{h}'']| \\
= \; & m \sum_{\bar{h}'(S) \neq \bar{h}(S) \text{ or } \bar{h}''(S) \neq \bar{h}(S)} |(\bar{p}[\bar{h}'] - \bar{p}'[\bar{h}'])\bar{p}[\bar{h}''] + \bar{p}'[\bar{h}'](\bar{p}[\bar{h}''] - \bar{p}'[\bar{h}''])| \\
\leq \; & m \sum_{\bar{h}', \bar{h}''} (|\bar{p}[\bar{h}'] - \bar{p}'[\bar{h}']|\bar{p}[\bar{h}''] + \bar{p}'[\bar{h}']|\bar{p}[\bar{h}''] - \bar{p}'[\bar{h}'']|) \\
= \; & 2m\|\bar{p} - \bar{p}'\|.
\end{aligned}$$

Thus, from Theorem 18, we have the following corollary about the robustness of the worst-case generalized Gibbs error algorithm when the perturbed prior is used. We note that the bounded loss assumption is reasonable since it holds for various practical loss functions such as Hamming loss or F_β loss.

Corollary 4. *If A is the worst-case generalized Gibbs error algorithm and the loss function of interest is upper bounded by a constant $m \geq 0$, then for any true prior $\bar{p}_0$ and any perturbed prior $\bar{p}_1$,*

$$t_{\bar{p}_0}^{\textit{worst}}(A(\bar{p}_1)) \geq (1 - \frac{1}{e}) \max_{\pi} t_{\bar{p}_0}^{\textit{worst}}(\pi) - m(4 - \frac{2}{e})\|\bar{p}_0 - \bar{p}_1\|.$$

8.1.3 Remarks

By taking $\bar{p}_1 = \bar{p}_0$, we can use Corollary 1 to recover the known approximation ratio for the maximum Gibbs error algorithm in Theorem 10. Similarly, we can use Corollary 2 to recover the approximation ratio for the batch maximum Gibbs error algorithm in Theorem 11. For the worst case, we can also use Corollary 3 to recover the approximation ratio for the least confidence algorithm in Theorem 6, and use Corollary 4 to recover the approximation ratio for the generalized Gibbs error algorithm in Theorem 14. Thus, our corollaries are generalizations of these previous theorems.

We also note that if the algorithm A is α-approximate (in the average or worst case) with an optimal constant α under some computational complexity assumptions (Golovin and Krause, 2011), then the constant α in our theorems is also optimal under the same assumptions. This can be proven easily by contradiction and setting $\bar{p}_1 = \bar{p}_0$.

8.2 Robust Active Learning Using Mixture Prior

For active learning in the Bayesian setting, it is very important to use a good prior since the prior determines which training examples the active learning algorithm would select and how good the final learned model would be. In practice, however, it is difficult to select the correct prior to use, and we usually resort to a perturbed prior instead.

However, in passive supervised learning methods such as in regularized logistic regression, it is common to use only a small set of potential priors and select the best prior from that set using a validation set, which is set aside for that purpose. Given the practical successes of these algorithms, our robustness analysis suggests that we should design priors that are close to every prior in the small finite set of priors.

A simple prior that overlaps with every prior in a set is the uniform mixture of the priors in the set. While it may not be particularly close to any element in the set, it is at least bounded away from the worst-case distance of 2 between disjoint priors. In fact, a k

Algorithm 8.1: Active learning for the mixture prior model.

```
Input: A set of $n$ priors $\{p^1, p^2, \ldots, p^n\}$, the initial normalized weights for the
       priors $\{w^1, w^2, \ldots, w^n\}$, and the budget of $k$ queries.
$p_0^i \leftarrow p^i$    for all $i = 1, 2, \ldots, n$;
$w_0^i \leftarrow w^i$    for all $i = 1, 2, \ldots, n$;
for $t = 1$ to $k$ do
    Choose an unlabeled example $x^*$ based on an active learning criterion;
    $y^* \leftarrow$ Query-label($x^*$);
    Update and normalize weights:
        $w_t^i \propto w_{t-1}^i \, p_{t-1}^i[y^*; x^*]$ for all $i = 1, 2, \ldots, n$;
    Update each posterior individually using Bayes' rule:
        $p_t^i[h] \propto p_{t-1}^i[h] \, \mathbb{P}[h(x^*) = y^* | h]$ for each $i = 1, 2, \ldots, n$ and $h \in \mathcal{H}$;
end
return $\{p_k^1, p_k^2, \ldots, p_k^n\}$ and $\{w_k^1, w_k^2, \ldots, w_k^n\}$;
```

component uniform mixture has ℓ_1 distance of no more than $2(1 - 1/k)$ from any of the component priors.

Hence, we propose to use a uniform mixture prior for practical robust active learning. In this model, instead of using only one prior, we maintain a weighted set of priors (or posteriors if we have already observed some labels). Every time we receive a label of an example, we update the weights of the posteriors based on how good their predictions on the example are. We also use the new label information to update the posteriors *individually* using Bayes' rule. We give the details of active learning for the mixture prior model in Algorithm 8.1.

In this algorithm, the unlabeled example x^* chosen in each iteration can be computed using any active learning criterion of choice. For instance, if the maximum Gibbs error criterion is used, then at iteration t, we have:

$$x^* = \arg\max_x \mathbb{E}_{y \sim p[\,\cdot\,;x]}[1 - p[y; x]],$$

where $p[y;x] = \sum_{i=1}^{n} w_{t-1}^{i} \, p_{t-1}^{i}[y;x]$. Thus, the active learning criterion can be computed from the weights and posteriors obtained from the previous iteration. After x^* is chosen, we query its label y^* and use this label to update the new weights and posteriors. In our algorithm, the weights and posteriors are always normalized so that $\sum_i w_t^i = 1$ for all t and $\sum_h p_t^i[h] = 1$ for all i and t. The algorithm returns the final weights and posteriors which can be used to make predictions on the new examples. More specifically, the predicted label of a new example x is $\arg\max_y \sum_{i=1}^{n} w_k^i \, p_k^i[y;x]$.

We note that Algorithm 8.1 does not require the hypotheses to be deterministic. In fact, the algorithm can be used with probabilistic hypotheses where $\mathbb{P}[h(x) = y|h]$ is a real value between 0 and 1. We also note that for a posterior p_t^i, computing $p_t^i[y;x]$ can be expensive. In this case, we can approximate this probability by using the MAP hypothesis. Particularly, we can approximate $p_t^i[y;x]$ by $p_{\text{MAP}}^i[y;x]$, the probability that x has label y according to the MAP hypothesis of the posterior p_t^i. This approximation can be used to approximate both the active learning criterion and the predicted label of a new example.

8.3 Experiments

In this section, we report our experimental results with the usage of different priors and the mixture prior model. Our experiments use the logistic regression model with different L_2 regularizers. It is well-known that using an L_2 regularizer is equivalent to imposing a Gaussian prior with mean zero and variance σ^2 on the parameter space. Thus, we can consider different priors for our model by varying the variance σ^2 of the regularizer.

We consider two sets of experiments that use the maximum Gibbs error active learning algorithm. We note that since our data sets are all binary classification data sets, the maximum Gibbs error algorithm is also equivalent to the least confidence algorithm and the maximum entropy algorithm. In our first set of experiments, we confirm our

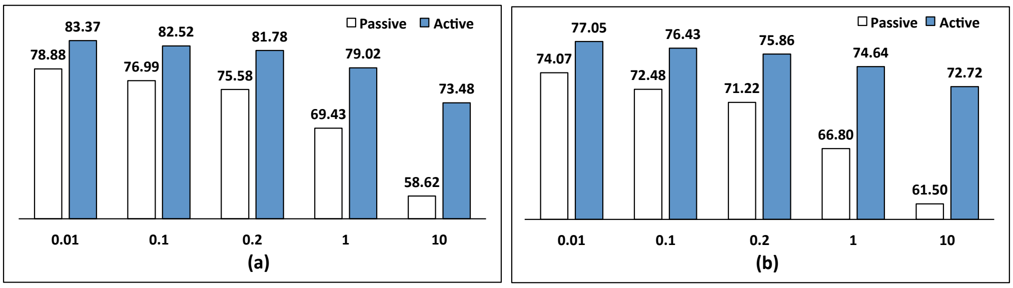

Figure 8.1: The average AUC scores for passive learning and the maximum Gibbs error active learning algorithm with $1/\sigma^2 = 0.01, 0.1, 0.2, 1,$ and 10 on the 20 Newsgroups data set (a) and the UCI data set (b).

theoretical findings by running the logistic regression model with different priors (or equivalently, regularizers). In the second set of experiments, we run the mixture prior model and compare it with models that use only one prior.

For the active learning algorithm in our experiments, we randomly choose the first 10 examples as a seed set. The scores in this section are averaged over 100 runs of the experiments with different seed sets. In all the experiments, we use the MAP hypotheses to approximate both the maximum Gibbs error criterion and the predicted label on the testing set, as described in Section 8.2.

8.3.1 Experiments With Different Priors

In this first set of experiments, we run the maximum Gibbs error active learning algorithm with regularizers $1/\sigma^2 = 0.01,\ 0.1,\ 0.2,\ 1,\ 10$ on 7 binary classification tasks from the 20 Newsgroups data set (Joachims, 1996) and on another 7 binary classification tasks from the UCI data set (Bache and Lichman, 2013). The tasks for the 20 Newsgroups data set are shown in the first column of Table 8.1, and the tasks for the UCI data set are shown in the first column of Table 8.2.

We show in Figure 8.1 the average areas under the accuracy curves (AUC) on the first 150 selected examples for the different regularizers. Figure 8.1a shows the average AUC scores for the 7 tasks from the 20 Newsgroups data set, while Figure 8.1b shows the average scores for the 7 tasks from the UCI data set. The AUCs are computed from the accuracy on a separate testing set. For completeness, we also compare the scores for active learning with passive learning.

From Figure 8.1, active learning is better than passive learning for all the regularizers. We also see that when the regularizers are close to each other (e.g., $1/\sigma^2 = 0.1$ and 0.2), the corresponding scores also tend to be close. When the regularizers are farther apart (e.g., $1/\sigma^2 = 0.1$ and 10), the scores also tend to be far to each other. Thus, this figure, in some sense, confirms our theoretical findings in Section 8.1.

8.3.2 Experiments With Mixture Prior

In this second set of experiments, we investigate the performance of the mixture prior model proposed in Section 8.2. We run the mixture prior model with the regularizers $1/\sigma^2 = 0.01,\ 0.1,\ 1,\ 10$ and compare it with models that use only one of these regularizers. Table 8.1 and 8.2 show the AUC scores of the first 150 selected examples for these models on the 20 Newsgroups data set and the UCI data set respectively. In the tables, double asterisks (**) indicate the best score, while an asterisk (*) indicates the second best score on a row (without the last column). The last columns of the tables show the AUCs of passive learning with the mixture prior model for comparison.

From the results in Table 8.1, the mixture prior model achieves the second best AUC scores for all the tasks in the 20 Newsgroups data set. From the results for the UCI data set in Table 8.2, the mixture prior model achieves the best score on the Ionosphere data set, and the second best scores on three other tasks. For the remaining three tasks, it achieves the third best scores. On average, the mixture prior model achieves the second best scores for both data sets. Thus, the mixture prior model performs reasonably well given the fact that we do not know which regularizer is the best to use for the data.

From Table 8.1 and 8.2, it is also interesting to note that if a bad regularizer is used (e.g.,

Data set	0.01	0.1	1	10	Mixture	Mixture (Passive)
alt.atheism/comp.graphics	87.60**	87.25	84.94	81.46	87.33*	83.92
talk.politics.guns/talk.politics.mideast	80.71**	79.28	74.57	66.76	79.49*	76.34
comp.sys.mac.hardware/comp.windows.x	78.75**	78.21*	75.07	70.54	78.21*	75.02
rec.motorcycles/rec.sport.baseball	86.20**	85.39	82.23	77.35	85.59*	81.56
sci.crypt/sci.electronics	78.08**	77.35	73.92	68.72	77.42*	73.08
sci.space/soc.religion.christian	86.09**	85.12	81.48	75.51	85.50*	80.31
soc.religion.christian/talk.politics.guns	86.16**	85.01	80.91	74.03	85.46*	81.81
Average	83.37**	82.52	79.02	73.48	82.71*	78.86

Table 8.1: AUCs of the maximum Gibbs error algorithm with $1/\sigma^2 = 0.01, 0.1, 1, 10$ and the mixture prior model on the 20 Newsgroups data set.

Data set	0.01	0.1	1	10	Mixture	Mixture (Passive)
Adult	79.38	80.15	80.39**	79.68	80.18*	77.41
Breast cancer	88.28*	88.37**	86.95	83.82	88.14	89.07
Diabetes	65.09*	64.53	64.39	65.48**	64.82	64.24
Ionosphere	82.80*	82.76	81.48	77.88	82.95**	81.91
Liver disorders	66.31**	64.16	61.42	58.42	64.73*	65.89
Mushroom	90.73**	89.56	84.14	82.94	90.33*	73.38
Sonar	66.75**	65.45*	63.74	60.81	65.00	66.53
Average	77.05**	76.43	74.64	72.72	76.59*	74.06

Table 8.2: AUCs of the maximum Gibbs error algorithm with $1/\sigma^2 = 0.01, 0.1, 1, 10$ and the mixture prior model on the UCI data set.

$1/\sigma^2 = 10$), the performance of active learning may be even worse than passive learning with the mixture prior model.

CHAPTER 9

Conclusion and Future Works

9.1 Conclusion

In this work, we have considered greedy algorithms for pool-based active learning in the Bayesian setting. We developed two useful tools for analyzing theoretical properties of these algorithms in the noisy case. The first tool is the equivalence between probabilistic and deterministic models, and the second tool is the near-optimality guarantee for greedy algorithms when maximizing pointwise monotone submodular functions. Using these tools, we proved a near-optimality guarantee for the well-known least confidence algorithm in the worst case. Furthermore, we also gave a negative result for another well-known algorithm, the maximum entropy algorithm, in the average case.

We also proposed a new objective function for Bayesian pool-based active learning: the policy Gibbs error. With this objective, we described the maximum Gibbs error criterion for selecting the examples. This greedy algorithm has average-case near-optimality guarantees in the non-adaptive, adaptive, and batch mode settings. We discussed methods to approximate the maximum Gibbs error criterion for Bayesian CRFs and Bayesian transductive Naive Bayes models. Our experimental results showed that the criterion is useful for named entity recognition with the Bayesian CRF model and for text classification with the Bayesian transductive Naive Bayes model.

As an improvement to the maximum Gibbs error algorithm, we also considered active learning with general losses and proposed two new greedy algorithms, one of which is for the average case and the other is for the worst case. We proved that the worst-case algorithm is always near-optimal, while the average-case algorithm is near-optimal under some conditions. The main theoretical results for these algorithms are based on our newly developed tool for analyzing pointwise monotone submodular functions. Our experiments showed that the new algorithms with general losses perform well in practice.

Lastly, we investigated the robustness of active learning in the Bayesian pool-based setting. We proved new robustness bounds for active learning algorithms that operate on a perturbed prior instead of the true prior. The bounds can be applied to various active learning algorithms such as the maximum Gibbs error, the batch maximum Gibbs error, the generalized Gibbs error, and the least confidence algorithms. We also proposed the use of mixture prior to make active learning algorithms more robust against a wrong prior. Our experiments showed that the mixture prior is a reasonable choice in case we do not know which prior is good for our data.

9.2 Future Works

There are several directions that can be considered for future works. In this section, we briefly describe the following four potential directions.

The Min-Cost Setting: The first direction is to prove the near-optimality guarantees in this thesis for the min-cost setting, rather than the maximum coverage setting as in this work. The min-cost setting for active learning can be generally stated as: *given a target utility value, find the active learning policy that requires the minimum number of queries (in the average case or worst case) to achieve the target utility*. In this setting, we conjecture that the algorithms proposed in this work can also achieve near-optimality. That is, the number of queries required by these algorithms is within a constant factor of the optimal number of queries.

The Multivariate Loss Functions: The total generalized version space in Chapter 7, $\sum_{\bar{h},\bar{h}'} \bar{p}_0[\bar{h}]L(\bar{h},\bar{h}')\bar{p}_0[\bar{h}']$, can be seen as an uncertainty measure of $\bar{p}_0$ on the hypothesis space $\overline{\mathcal{H}}$. This uncertainty measure can be further generalized as follows. Let p be a probability measure on a space $\mathcal{Z}$ with $p[z]$ being the probability of $z \in \mathcal{Z}$. Consider a function $L : \mathcal{Z}^q \to \mathbb{R}_{\geq 0}$ from the set of q-tuples of $\mathcal{Z}$ to non-negative real numbers. We define an uncertainty measure $H_L(p) \overset{\text{def}}{=} \frac{1}{q-1} \sum_{z_1, z_2, \ldots, z_q} p[z_1]p[z_2] \ldots p[z_q] L(z_1, z_2, \ldots, z_q)$ of p. Note that when L is the 0-1 loss, i.e., $L(z_1, z_2, \ldots, z_q) = \mathbf{1}(\exists i, j \text{ s.t. } z_i \neq z_j)$, we have $H_{\text{0-1}}(p) = \frac{1}{q-1}\left(1 - \sum_z p[z]^q\right)$, which is the Tsallis entropy with entropic-index q (Tsallis and Brigatti, 2004). A possible direction for a future work is to investigate whether this new uncertainty measure is useful for active learning or other applications.

The Non-uniform Cost Setting: Another direction for future works is to consider different costs for different types of queries. In the non-adaptive setting and the average-case setting with adaptive monotone submodular functions, the greedy algorithms can take into account the cost of each query by greedily choosing the example with maximum utility gain per unit cost. For these cases, the greedy algorithms can still obtain near-optimality compared to the optimal policy. It would be useful to prove similar results for the worst-case setting considered in Chapter 7 of this work. In addition, using different costs for different queries may be desirable for active learning with segmentation models such as the semi-Markov conditional random fields (Sarawagi and Cohen, 2004). In these models, segments of the input sequence may have different lengths, and the costs or time to label them may also be different. Thus, the active learning algorithms should take into account the costs for labeling the segments.

Improving Batch maxGEC: The fourth direction for future works is to improve the near-optimality bound for the batch mode setting in Chapter 6. Recall that in the batch mode setting, we need to select a batch of size s in each iteration before observing their labels. If in each iteration with the current posterior p, a batch active learning policy π always chooses the ϵ-optimal batch S such that $\epsilon_g^p(S) \geq \max_{S':|S'|=s} \epsilon_g^p(S') - \epsilon$, then from a result by Golovin and Krause (2011), we have $H_{\text{gibbs}}(\pi) \geq \left(1 - \frac{1}{e}\right) H_{\text{gibbs}}(\pi_b^*) - k\epsilon \approx 0.63 H_{\text{gibbs}}(\pi_b^*) - k\epsilon$, where k is the number of selected batches and π_b^* is the optimal batch policy. When compared to the bound in Theorem 11, i.e. $H_{\text{gibbs}}(\pi_b^{maxGEC}) > 0.47 H_{\text{gibbs}}(\pi_b^*)$, the above bound may be better when ϵ is small. To select an ϵ-optimal batch with small ϵ in each iteration, we may consider an exhaustive search strategy and thus trade off the computational efficiency for better accuracy.

References

Alekh Agarwal. Selective sampling algorithms for cost-sensitive multiclass prediction. In *International Conference on Machine Learning (ICML)*, pages 1220–1228, 2013.

Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM 2006 Workshop on Link Analysis, Counterterrorism and Security*, 2006.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Dana Angluin. Queries revisited. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 12–31, 2001.

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4): 343–370, 1988.

Les Atlas, David Cohn, Richard Ladner, M.A. El-Sharkawi, R.J. Marks II, M.E. Aggoune, and D.C. Park. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 566–573, 1990.

Jean-Yves Audibert and Alexandre B. Tsybakov. Fast learning rates for plug-in classifiers. *Annals of Statistics*, 35(2):608–633, 2007.

Kevin Bache and Moshe Lichman. UCI machine learning repository. *School of Information and Computer Science, University of California–Irvine*, 2013.

Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman Vaughan. The true sample complexity of active learning. *Machine Learning*, 80(2-3):111–139, 2010.

Jason Baldridge and Miles Osborne. Active learning and the total cost of annotation. In

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9–16, 2004.

Eric B. Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks (IJCNN)*, volume 8, 1992.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *International Conference on Machine Learning (ICML)*, pages 49–56, 2009.

Alina Beygelzimer, Daniel Hsu, John Langford, and Tong Zhang. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems (NIPS)*, pages 199–207, 2010.

Rui M. Castro and Robert D. Nowak. Upper and lower error bounds for active learning. In *Annual Allerton Conference on Communication, Control and Computing*, volume 2, page 1, 2006.

Rui M. Castro and Robert D. Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008.

Rui M. Castro, Rebecca Willett, and Robert D. Nowak. Faster rates in regression via active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 179–186, 2005.

Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *International Conference on Machine Learning (ICML)*, pages 160–168, 2013.

Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

References

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *National Conference on Artificial Intelligence (AAAI)*, pages 746–751, 2005.

Nguyen Viet Cuong, Vu Dinh, and Lam Si Tung Ho. Mel-frequency cepstral coefficients for eye movement identification. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, 2012.

Nguyen Viet Cuong, Lam Si Tung Ho, and Vu Dinh. Generalization and robustness of batched weighted average algorithm with V-geometrically ergodic Markov data. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 264–278, 2013a.

Nguyen Viet Cuong, Wee Sun Lee, Nan Ye, Kian Ming A. Chai, and Hai Leong Chieu. Active learning for probabilistic hypotheses using the maximum Gibbs error criterion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1457–1465, 2013b.

Nguyen Viet Cuong, Wee Sun Lee, and Nan Ye. Near-optimal adaptive pool-based active learning with general loss. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014a.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Conditional random field with high-order dependencies for sequence labeling and segmentation. *Journal of Machine Learning Research*, 15:981–1009, 2014b.

Nguyen Viet Cuong, Muthu Kumar Chandrasekaran, Min-Yen Kan, and Wee Sun Lee. Scholarly document information extraction using extensible features for efficient higher order semi-CRFs. In *Joint Conference on Digital Libraries (JCDL)*, 2015.

Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *International Conference on Machine Learning (ICML)*, pages 150–157, 1995.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*, pages 337–344, 2004.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In *Conference on Learning Theory (COLT)*, pages 249–263, 2005.

Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 353–360, 2007.

Vu Dinh, Lam Si Tung Ho, Nguyen Viet Cuong, Duy Duc Nguyen, and Binh T. Nguyen. Learning from non-iid data: Fast rates for the one-vs-all multiclass plug-in classifiers. In *Conference on Theory and Applications of Models of Computation (TAMC)*, 2015.

R.S. Forsyth. PC/Beagle User's Guide. *BUPA Medical Research Ltd*, 1990.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

Satoru Fujishige. Polymatroidal dependence structure of a set of random variables. *Information and Control*, 39(1):55–72, 1978.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42(1):427–486, 2011.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal Bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 766–774, 2010.

R. Paul Gorman and Terrence J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1(1):75–89, 1988.

Guodong Guo, Stan Z. Li, and Kap Luk Chan. Face recognition by support vector machines. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 196–201, 2000.

Yuhong Guo and Russ Greiner. Optimistic active learning using mutual information. In *International Joint Conference on Artifical Intelligence (IJCAI)*, pages 823–829, 2007.

Steve Hanneke. A bound on the label complexity of agnostic active learning. In *International Conference on Machine Learning (ICML)*, pages 353–360, 2007.

Steven C.H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *International Conference on World Wide Web (WWW)*, pages 633–642, 2006a.

Steven C.H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *International Conference on Machine Learning (ICML)*, pages 417–424, 2006b.

Steven C.H. Hoi, Rong Jin, and Michael R. Lyu. Batch mode active learning with applications to text categorization and image retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1233–1248, 2009.

Ya-Ping Huang, Si-Wei Luo, and En-Yi Chen. An efficient iris recognition system. In *International Conference on Machine Learning and Cybernetics*, volume 1, pages 450–454, 2002.

Rebecca Hwa. Sample selection for statistical parsing. *Computational Linguistics*, 30 (3):253–276, 2004.

Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Technical report, DTIC Document, 1996.

Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G.K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427 (6971):247–252, 2004.

Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The automation of science. *Science*, 324(5923):85–89, 2009.

Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 202–207, 1996.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

David D. Lewis and Jason Catlett. Heterogenous uncertainty sampling for supervised learning. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1994.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. In *Algorithmic Foundations of Robotics XI*, pages 283–300. 2015a.

Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive stochastic optimization: From sets to paths. In *Advances in Neural Information Processing Systems (NIPS)*. 2015b.

Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and SavageBoost. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1049–1056, 2009.

Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *International Conference on Machine Learning (ICML)*, pages 350–358, 1998.

Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici. Improving the detection of unknown computer worms activity using active learning. In *KI 2007: Advances in Artificial Intelligence*, pages 489–493. 2007.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Semi-Markov conditional random field with high-order features. In *ICML Workshop on Structured Sparsity: Learning and Inference*, 2011.

Eric W. Noreen. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, 1989.

Robert Nowak. Generalized binary search. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 568–574, 2008.

Robert Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1366–1374, 2009.

Robert D. Nowak. The geometry of generalized binary search. *IEEE Transactions on Information Theory*, 57(12):7893–7906, 2011.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning (ICML)*, pages 441–448, 2001.

Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1185–1192, 2004.

Bassem Sayrafi, Dirk Van Gucht, and Marc Gyssens. The implication problem for measure-based constraints. *Information Systems*, 33(2):221–239, 2008.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden Markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318, 2001.

Jeffrey Curtis Schlimmer. Concept acquisition through representational adjustment. *University of California–Irvine*, 1987.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

Burr Settles. *Curious machines: Active learning with structured instances*. PhD thesis, University of Wisconsin–Madison, 2008.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2010.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1070–1079, 2008.

Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, 2008a.

Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296, 2008b.

References

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Annual Workshop on Computational Learning Theory (COLT)*, pages 287–294, 1992.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, pages 262–266, 1989.

Jack W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Annual Symposium on Computer Application in Medical Care*, pages 261–265, 1988.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *International Conference on Machine Learning (ICML)*, pages 406–414, 1999.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Conference on Natural Language Learning at HLT-NAACL (CoNLL)*, pages 142–147, 2003.

Katrin Tomanek and Fredrik Olsson. A web survey on the use of active learning to support annotation of text data. In *NAACL–HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48, 2009.

Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *ACM International Conference on Multimedia*, pages 107–118, 2001.

Constantino Tsallis and Edgardo Brigatti. Nonextensive statistical mechanics: A brief introduction. *Continuum Mechanics and Thermodynamics*, 16(3):223–235, 2004.

Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, pages 135–166, 2004.

Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45 (2):171–186, 2005.

Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

William H. Wolberg and Olvi L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87(23):9193–9196, 1990.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *International Conference on Computational Linguistics (COLING)*, pages 947–953, 2000.

Cha Zhang and Tsuhan Chen. An active learning framework for content-based information retrieval. *IEEE Transactions on Multimedia*, 4(2):260–268, 2002.

Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. 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*, pages 58–65, 2003.