Robustness of Bayesian Pool-based Active Learning Against Prior Misspecification

Nguyen Viet Cuong[1] **Nan Ye**[2] **Wee Sun Lee**[3]

[1]Department of Mechanical Engineering, National University of Singapore, Singapore, *nvcuong@nus.edu.sg*
[2]Mathematical Sciences School & ACEMS, Queensland University of Technology, Australia, *n.ye@qut.edu.au*
[3]Department of Computer Science, National University of Singapore, Singapore, *leews@comp.nus.edu.sg*

Abstract

We study the robustness of active learning (AL) algorithms against prior misspecification: whether an algorithm achieves similar performance using a perturbed prior as compared to using the true prior. In both the average and worst cases of the maximum coverage setting, we prove that all α-approximate algorithms are robust (i.e., near α-approximate) if the utility is Lipschitz continuous in the prior. We further show that robustness may not be achieved if the utility is non-Lipschitz. This suggests we should use a Lipschitz utility for AL if robustness is required. For the minimum cost setting, we can also obtain a robustness result for approximate AL algorithms. Our results imply that many commonly used AL algorithms are robust against perturbed priors. We then propose the use of a mixture prior to alleviate the problem of prior misspecification. We analyze the robustness of the uniform mixture prior and show experimentally that it performs reasonably well in practice.

1 Introduction

In pool-based active learning (AL), training examples are sequentially selected and labeled from a pool of unlabeled data, with the aim of obtaining a good classifier using as few labeled examples as possible (McCallum and Nigam 1998). To achieve computational efficiency, most commonly used methods greedily select one example at a time based on some criterion.

In this paper, we consider Bayesian pool-based AL that assumes data labels are generated from a prior distribution. In theory, the true prior is generally assumed to be known (Golovin and Krause 2011; Cuong et al. 2013; Cuong, Lee, and Ye 2014). In practice, it is often unknown and misspecified; that is, the prior used is different from the true one. This work is the first one investigating the *robustness* of AL algorithms against prior misspecification – that is, whether an algorithm achieves similar performance using a perturbed prior as compared to using the true prior.

We focus on the analysis of approximate algorithms for two commonly studied problems: the *maximum coverage problem* which aims to maximize the expected or worst-case utility of the chosen examples given a fixed budget of queries, and the *minimum cost problem* which aims to minimize the expected number of queries needed to identify the true labeling of all examples. We focus on approximate algorithms because previous works have shown that, in general, it is computationally intractable to find the optimal strategy for choosing the examples, while some commonly used AL algorithms can achieve good approximation ratios compared to the optimal strategies (Golovin and Krause 2011; Chen and Krause 2013; Cuong et al. 2013; Cuong, Lee, and Ye 2014). For example, with the version space reduction utility, the maximum Gibbs error algorithm achieves a $(1 - 1/e)$-approximation of the optimal expected utility (Cuong et al. 2013), while the least confidence algorithm achieves the same approximation of the optimal worst-case utility (Cuong, Lee, and Ye 2014).

Our work shows that many commonly used AL algorithms are robust. In the maximum coverage setting, our main result is that if the utility function is Lipschitz continuous in the prior, all α-approximate algorithms are robust, i.e., they are near α-approximate when using a perturbed prior. More precisely, their performance guarantee on the expected or worst-case utility degrades by at most a constant factor of the ℓ_1 distance between the perturbed prior and the true prior. It follows from this result that the maximum Gibbs error and the least confidence algorithms are near $(1 - 1/e)$-approximate. Our result also implies the robustness of the batch and generalized versions of the maximum Gibbs error algorithm. If the utility is non-Lipschitz, we show that even an optimal algorithm for the perturbed prior may not be robust. This suggests we should use a Lipschitz continuous utility for AL in order to achieve robustness. Similarly, we prove a robustness result for the minimum cost setting that implies the robustness of the generalized binary search AL algorithm (Dasgupta 2004; Nowak 2008).

We also address the difficulty of choosing a good prior in practice. Practically, it is often easier to come up with a set of distributions and combine them using a mixture. Our theoretical results imply robustness when the mixture prior is close to the true prior. In the mixture setting, another interesting question is robustness when the true prior is one of the components of the mixture. In this case, the mixture

prior may not necessarily be close to the true prior in terms of ℓ_1 distance. We prove that for the uniform mixture prior, approximate AL algorithms are still robust in the sense that they are competitive with the optimum performance of the mixture, which is the performance we expect when modeling. Our experiments show that the uniform mixture performs well in practice.

Related Works: Greedy algorithms for pool-based AL usually optimize some measure of uncertainty of the selected examples (Settles 2009; Cuong et al. 2013). In the Bayesian setting, they can be viewed as greedily optimizing some corresponding average-case or worst-case objective. For instance, the maximum entropy algorithm (Settles and Craven 2008), which maximizes the Shannon entropy of the selected examples, attempts to greedily maximize the policy entropy in the average case (Cuong et al. 2013). The maximum Gibbs error algorithm, which maximizes the Gibbs error of the selected examples, attempts to greedily maximize the version space reduction in the average case (Cuong et al. 2013); and the least confidence algorithm, which minimizes the probability of the most likely label of the selected examples, attempts to maximize the version space reduction in the worst case (Cuong, Lee, and Ye 2014).

Analyses of these algorithms typically investigate their near-optimality guarantees in the average or worst case. The maximum entropy algorithm was shown to have no constant factor approximation guarantee in the average case (Cuong, Lee, and Ye 2014). In contrast, the maximum Gibbs error algorithm has a $(1 - 1/e)$-factor approximation guarantee for the average version space reduction objective (Cuong et al. 2013). This algorithm is a probabilistic version of the generalized binary search algorithm (Dasgupta 2004; Golovin and Krause 2011). It can also be applied to the batch mode setting (Hoi et al. 2006), and was shown to provide a $(1 - e^{-(e-1)/e})$-factor approximation to the optimal batch AL algorithm (Cuong et al. 2013). In the noiseless case, this batch maximum Gibbs error algorithm is equivalent to the BatchGreedy algorithm (Chen and Krause 2013).

Cuong, Lee, and Ye (2014) showed the least confidence algorithm (Culotta and McCallum 2005) has a $(1 - 1/e)$-factor approximation guarantee with respect to the worst-case version space reduction objective. A similar result in the worst case was also shown for the generalized maximum Gibbs error algorithm with an arbitrary loss (Cuong, Lee, and Ye 2014). These results are due to the pointwise submodularity of version space reduction. AL that exploits submodularity was also studied in (Guillory and Bilmes 2010; Wei, Iyer, and Bilmes 2015).

2 Preliminaries

Let $\mathcal{X}$ be a finite set (a pool) of examples and $\mathcal{Y}$ be a finite set of labels. Consider the hypothesis space $\mathcal{H} \stackrel{\text{def}}{=} \mathcal{Y}^{\mathcal{X}}$ consisting of all functions from $\mathcal{X}$ to $\mathcal{Y}$. Each hypothesis $h \in \mathcal{H}$ is a *labeling* of $\mathcal{X}$. In the Bayesian setting, we assume an unknown true labeling h_{true} drawn from a prior $p_0[h]$ on $\mathcal{H}$. After observing a labeled set $\mathcal{D}$, we obtain the posterior $p_{\mathcal{D}}[h] \stackrel{\text{def}}{=} p_0[h \mid \mathcal{D}]$ using Bayes' rule.

The true labeling h_{true} may be generated by a complex process rather than directly drawn from a prior p_0. For instance, if probabilistic models (e.g., naive Bayes) are used to generate labels for the examples and a prior is imposed on these models instead of the labelings, we can convert this prior into an equivalent prior on the labelings and work with this induced prior. The construction of the induced prior involves computing the probability of labelings with respect to the original prior (Cuong et al. 2013). In practice, we do not need to compute or maintain the induced prior explicitly as this process is very expensive. Instead, we compute or approximate the AL criteria directly from the original prior on the probabilistic models (Cuong et al. 2013; Cuong, Lee, and Ye 2014).

For any distribution $p[h]$, any example sequence $S \subseteq \mathcal{X}$, and any label sequence $\mathbf{y}$ with the same length, $p[\mathbf{y}; S]$ denotes the probability that $\mathbf{y}$ is the label sequence of S. Formally, $p[\mathbf{y}; S] \stackrel{\text{def}}{=} \sum_h p[h]\, \mathbb{P}[h(S) = \mathbf{y} \mid h]$, where $h(S) = (h(x_1), \ldots, h(x_i))$ if $S = (x_1, \ldots, x_i)$. In the above, $\mathbb{P}[h(S) = \mathbf{y} \mid h]$ is the probability that S has label sequence $\mathbf{y}$ given the hypothesis h. If h is deterministic as in our setting, $\mathbb{P}[h(S) = \mathbf{y} \mid h] = \mathbf{1}(h(S) = \mathbf{y})$, where $\mathbf{1}(\cdot)$ is the indicator function. Note that $p[\,\cdot\,; S]$ is a probability distribution on the label sequences of S. We also write $p[y; x]$ to denote $p[\{y\}; \{x\}]$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Given a prior, a pool-based AL algorithm is equivalent to a policy for choosing training examples from $\mathcal{X}$. A policy is a mapping from a partial labeling (labeling of a subset of $\mathcal{X}$) to the next unlabeled example to query. It can be represented by a policy tree whose nodes correspond to unlabeled examples to query, and edges from a node correspond to its labels. When an unlabeled example is queried, we receive its true label according to h_{true}. We will focus on adaptive policies, which use the observed labels of previously chosen examples to query the next unlabeled example.

3 Robustness: Maximum Coverage Problem

We now consider the robustness of AL algorithms for the *maximum coverage* problem: find an adaptive policy maximizing the expected or worst-case utility given a budget of k queries (Cuong et al. 2013; Cuong, Lee, and Ye 2014). The *utility* is a non-negative function $f(S, h) : 2^{\mathcal{X}} \times \mathcal{H} \to \mathbb{R}_{\geq 0}$. Intuitively, a utility measures the value of querying examples S when the true labeling is h. Utilities for AL usually depend on the prior, so we shall use the notation $f_p(S, h)$ to denote that the utility f_p depends on a distribution p over $\mathcal{H}$.

f_p is said to be *Lipschitz continuous (in the prior)* with a Lipschitz constant L if for any S, h, and two priors p, p',

$$|f_p(S, h) - f_{p'}(S, h)| \leq L\|p - p'\|, \tag{1}$$

where $\|p - p'\| \stackrel{\text{def}}{=} \sum_h |p[h] - p'[h]|$ is the ℓ_1 distance between p and p'. Lipschitz continuity implies boundedness.[1]

An AL algorithm is a mapping from a utility and a prior to a policy. Let x_h^π denote the set of examples selected by a policy π when the true labeling is h. We now analyze the robustness of AL algorithms for both the average case and the worst case.

[1] Choose an arbitrary p', then for any p, S, h, we have $f_p(S, h) \leq f_{p'}(S, h) + L\|p - p'\| \leq f_{p'}(S, h) + 2L \leq \max_{S,h} f_{p'}(S, h) + 2L$. Similarly, a lower bound exists.

3.1 The Average Case

In this case, our objective is to find a policy with maximum expected utility. If p_0 is the true prior, the expected utility of a policy π is $f_{p_0}^{\text{avg}}(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{h\sim p_0}\left[f_{p_0}(x_h^\pi, h)\right]$.

We consider the case where we have already chosen a utility, 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 p_1 (i.e., a prior not very different from the true prior p_0 in terms of ℓ_1 distance), can an AL algorithm achieve performance competitive to that obtained using the true prior?

Our first robustness result is for *α-approximate* algorithms that return an α-approximate policy of the optimal one. Formally, an *average-case α-approximate* ($0 < \alpha \leq 1$) algorithm A outputs, for any prior p, a policy $A(p)$ satisfying

$$f_p^{\text{avg}}(A(p)) \geq \alpha \max_\pi f_p^{\text{avg}}(\pi).$$

When $\alpha = 1$, we call A an exact algorithm. For notational convenience, we drop the dependency of A on the utility as we assumed a fixed utility here. We consider approximate algorithms because practical algorithms are generally approximate due to computational intractability of the problem. We have the following robustness result.

Theorem 1. *Assume the utility f_p is Lipschitz continuous with a Lipschitz constant L. Let M be an upper bound of f_p. If A is an average-case α-approximate algorithm, then for any true prior p_0 and any perturbed prior p_1,*

$$f_{p_0}^{avg}(A(p_1)) \geq \alpha \max_\pi f_{p_0}^{avg}(\pi) - (\alpha+1)(L+M)\|p_1 - p_0\|.$$

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

Proof. Let $C = L + M$. For any policy π, note that:

$$\begin{aligned} & |f_{p_0}^{\text{avg}}(\pi) - f_{p_1}^{\text{avg}}(\pi)| \\ = \; & |(\sum_h p_0[h] f_{p_0}(x_h^\pi, h) - \sum_h p_0[h] f_{p_1}(x_h^\pi, h)) \\ & +(\sum_h p_0[h] f_{p_1}(x_h^\pi, h) - \sum_h p_1[h] f_{p_1}(x_h^\pi, h))| \\ \leq \; & C\|p_1 - p_0\|, \end{aligned}$$

where the last inequality holds due to the Lipschitz continuity and boundedness of the utility function f_p. Thus, if $\pi_1 = \arg\max_\pi f_{p_1}^{\text{avg}}(\pi)$ and $\pi_0 = \arg\max_\pi f_{p_0}^{\text{avg}}(\pi)$, it follows that:

$$\begin{aligned} f_{p_1}^{\text{avg}}(\pi_1) \; &\geq \; f_{p_1}^{\text{avg}}(\pi_0) \geq f_{p_0}^{\text{avg}}(\pi_0) - C\|p_1 - p_0\|, \text{ and} \\ f_{p_0}^{\text{avg}}(\pi) \; &\geq \; f_{p_1}^{\text{avg}}(\pi) - C\|p_1 - p_0\| \text{ for all } \pi. \end{aligned}$$

Hence,

$$\begin{aligned} f_{p_0}^{\text{avg}}(A(p_1)) \; &\geq \; f_{p_1}^{\text{avg}}(A(p_1)) - C\|p_1 - p_0\| \\ &\geq \; \alpha f_{p_1}^{\text{avg}}(\pi_1) - C\|p_1 - p_0\| \\ &\geq \; \alpha(f_{p_0}^{\text{avg}}(\pi_0) - C\|p_1 - p_0\|) - C\|p_1 - p_0\| \\ &= \; \alpha \max_\pi f_{p_0}^{\text{avg}}(\pi) - C(\alpha+1)\|p_1 - p_0\|, \end{aligned}$$

where the first and third inequalities are from the above discussions and the second inequality holds as A is α-approximate. □

$f_{p_0}^{\text{avg}}(A(p_1))$ is the expected utility of the policy returned by A using p_1 as prior. The expected utility is always computed with respect to the true prior p_0. Theorem 1 shows that when we use a perturbed prior p_1, the expected utility achieved by an average-case α-approximate algorithm degrades by at most a constant factor of the ℓ_1 distance between the perturbed prior and the true prior.

Application to Maximum Gibbs Error: Theorem 1 implies the robustness of the maximum Gibbs error algorithm (Cuong et al. 2013). This algorithm greedily selects the next example x^* satisfying $x^* = \arg\max_x \mathbb{E}_{y\sim p_{\mathcal{D}}[\cdot;x]}[1 - p_{\mathcal{D}}[y;x]]$, where $p_{\mathcal{D}}$ is the current posterior and $p_{\mathcal{D}}[y;x]$ is the probability (w.r.t. $p_{\mathcal{D}}$) that x has label y. In the binary-class and noiseless setting, it is equivalent to the generalized binary search algorithm (Dasgupta 2004; Nowak 2008; Golovin and Krause 2011). Consider the version space reduction utility $f_p(S,h) \stackrel{\text{def}}{=} 1 - p[h(S);S]$, where $p[h(S);S]$ is the probability (w.r.t. p) that S has the labels $h(S)$. We have the following corollary about the algorithm. The proofs of this corollary and the remaining results are in the supplementary material.

Corollary 1. *If A is the maximum Gibbs error algorithm, then for any true prior p_0 and any perturbed prior p_1,*

$$f_{p_0}^{avg}(A(p_1)) \geq \left(1 - \frac{1}{e}\right) \max_\pi f_{p_0}^{avg}(\pi) - \left(4 - \frac{2}{e}\right)\|p_1 - p_0\|.$$

Application to Batch Maximum Gibbs Error: We can also obtain the robustness result for the batch version of the maximum Gibbs error algorithm. In the batch setting, the AL algorithm queries a batch of examples in each iteration instead of only one example (Hoi et al. 2006). The batch maximum Gibbs error algorithm is described in Algorithm 1 of (Cuong et al. 2013), and by Theorem 5 of the same work, it is a $(1 - e^{-(e-1)/e})$-approximate algorithm for the version space reduction utility above. If we restrict the policies to only those in the batch setting, then from Theorem 1, we have the following corollary. Note that the range of the max operation in the corollary is restricted to only batch policies.

Corollary 2. *If A is the batch maximum Gibbs error algorithm, for any true prior p_0 and any perturbed prior p_1,*

$$\begin{aligned} f_{p_0}^{avg}(A(p_1)) \geq & \left(1 - e^{-(e-1)/e}\right) \max_\pi f_{p_0}^{avg}(\pi) \\ & - \left(4 - 2e^{-(e-1)/e}\right)\|p_1 - p_0\|. \end{aligned}$$

3.2 The Worst Case

In this case, our objective is to find a policy with maximum worst-case utility. If p_0 is the true prior, the worst-case utility of a policy π is $f_{p_0}^{\text{worst}}(\pi) \stackrel{\text{def}}{=} \min_h \left[f_{p_0}(x_h^\pi, h)\right]$.

An algorithm A is a *worst-case α-approximate* algorithm ($0 < \alpha \leq 1$) if for any prior p, we have $f_p^{\text{worst}}(A(p)) \geq \alpha \max_\pi f_p^{\text{worst}}(\pi)$. When $\alpha = 1$, A is an exact algorithm.

For worst-case α-approximate algorithms, we can obtain a robustness result similar to Theorem 1.

Theorem 2. *Assume f_p is Lipschitz continuous with a Lipschitz constant L. If A is a worst-case α-approximate algorithm, then for any true prior p_0 and perturbed prior p_1,*

$$f_{p_0}^{worst}(A(p_1)) \geq \alpha \max_{\pi} f_{p_0}^{worst}(\pi) - (\alpha+1)L\|p_1 - p_0\|.$$

The worst-case utility is also computed with respect to the true prior p_0 (i.e., using $f_{p_0}^{\text{worst}}$ instead of $f_{p_1}^{\text{worst}}$). Theorem 2 shows that when we use a perturbed prior, the worst-case utility achieved by a worst-case α-approximate algorithm degrades by at most a constant factor of the ℓ_1 distance between the perturbed prior and the true prior.

Application to Least Confidence: Theorem 2 implies the robustness of the well-known least confidence AL algorithm (Lewis and Gale 1994; Culotta and McCallum 2005) with perturbed priors. This algorithm greedily selects the next example x^* satisfying $x^* = \arg\min_x\{\max_{y\in\mathcal{Y}} p_{\mathcal{D}}[y;x]\}$. If f_p is the version space reduction utility (considered previously for the maximum Gibbs error algorithm), we have the following corollary.

Corollary 3. *If A is the least confidence algorithm, then for any true prior p_0 and any perturbed prior p_1,*

$$f_{p_0}^{worst}(A(p_1)) \geq \left(1-\frac{1}{e}\right) \max_{\pi} f_{p_0}^{worst}(\pi) - \left(2-\frac{1}{e}\right)\|p_1-p_0\|.$$

Application to Generalized Maximum Gibbs Error: Theorem 2 also implies the robustness of the worst-case generalized Gibbs error algorithm (Cuong, Lee, and Ye 2014) with a bounded loss. Intuitively, the algorithm greedily maximizes in the worst case the total generalized version space reduction, which is defined as

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

where L is a non-negative loss function between labelings that satisfies $L(h,h') = L(h',h)$ and $L(h,h) = 0$ for all h, h'. The worst-case generalized Gibbs error algorithm attempts to greedily maximize $t_{p_0}^{\text{worst}}(\pi) \stackrel{\text{def}}{=} \min_h t_{p_0}(x_h^\pi, h)$, and it is a worst-case $(1-1/e)$-approximate algorithm for this objective (Cuong, Lee, and Ye 2014).

If we assume L is upper bounded by a constant m, we have the following corollary about this algorithm. Note that the bounded loss assumption is reasonable since it holds for various practical loss functions such as Hamming loss or F_1 loss, which is $1 - F_1(h,h')$ where $F_1(h,h')$ is the F_1 score between h and h'.

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 p_0 and any perturbed prior p_1,*

$$t_{p_0}^{worst}(A(p_1)) \geq \left(1-\frac{1}{e}\right) \max_{\pi} t_{p_0}^{worst}(\pi) - m\left(4-\frac{2}{e}\right)\|p_1-p_0\|.$$

3.3 Discussions

We emphasize that our results are important as they enhance our understanding and confidence about existing AL algorithms. Furthermore, if the utility we want to maximize is not Lipschitz continuous, then even an exact AL algorithm for perturbed priors may not be robust, both in the average and worst cases. We prove this in Theorem 3 below.

Theorem 3. *For both the average and worst cases, there is an AL problem with a non-Lipschitz utility such that: for any $C, \alpha, \epsilon > 0$, there exist a perturbed prior p_1 satisfying $0 < \|p_1 - p_0\| < \epsilon$ and an exact algorithm A^* satisfying*

$$f_{p_0}^{c}(A^*(p_1)) < \alpha \max_{\pi} f_{p_0}^{c}(\pi) - C\|p_1 - p_0\|,$$

where $f_{p_0}^{c} \in \{f_{p_0}^{avg}, f_{p_0}^{worst}\}$ respectively.

This theorem and our results above suggest we should use a Lipschitz utility for AL to maintain the robustness.

By taking $p_1 = p_0$, Corollaries 1 and 2 can recover the approximation ratios for the maximum Gibbs error and batch maximum Gibbs error algorithms in Theorems 4 and 5 of (Cuong et al. 2013) respectively. Similarly, Corollaries 3 and 4 can recover the ratios for the least confidence and generalized Gibbs error algorithms in Theorems 5 and 8 of (Cuong, Lee, and Ye 2014) respectively. Thus, our corollaries are generalizations of these previous theorems.

If A is α-approximate (in the average or worst case) with an optimal constant α under some computational complexity assumption (Golovin and Krause 2011), then it is also optimal in our theorems under the same assumption. This can be proven easily by contradiction and setting $p_1 = p_0$.

If we are only interested in some particular prior p_0 and the perturbed priors within a neighborhood of p_0, we can relax the Lipschitz assumption (1) to the locally Lipschitz assumption at p_0: there exist L and δ such that for all S, h, and p, if $\|p_0 - p\| < \delta$, then $|f_{p_0}(S,h) - f_p(S,h)| \leq L\|p_0 - p\|$. Under this relaxed assumption, the theorems and corollaries above still hold for any p_1 satisfying $\|p_0 - p_1\| < \delta$.

4 Robustness: Minimum Cost Problem

In this section, we investigate the robustness of AL algorithms for the *minimum cost* problem in the average case: find an adaptive policy minimizing the expected number of queries to identify the true labeling h_{true} (Golovin and Krause 2011). This problem assumes h_{true} is drawn from a prior p_0 on a small hypothesis space $\mathcal{H}$ (i.e., $\mathcal{H}$ does not need to contain all functions from $\mathcal{X}$ to $\mathcal{Y}$). After we make a query and observe a label, all the hypotheses inconsistent with the observed label are removed from the current hypothesis space (also called the version space). We stop when there is only one hypothesis h_{true} left.

We do not consider this problem in the worst case because even the optimal worst-case algorithm may not be robust.[2] For instance, if the true prior gives probability 1 to one correct hypothesis but the perturbed prior gives positive probabilities to all the hypotheses, then the cost of using the true prior is 0 while the cost of using the perturbed prior is $|\mathcal{X}|$.

For any policy π and hypothesis h, let $c(\pi,h)$ be the cost of identifying h when running π. This is the length of the path corresponding to h in the policy tree of π. For any prior p_0 and policy π, the expected cost of π with respect to the prior p_0 is defined as $c_{p_0}^{\text{avg}}(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{h\sim p_0}[c(\pi,h)]$.

[2] The Lipschitz assumption is not satisfied in this setting.

We will consider $\alpha(p)$-approximate algorithms that return a policy whose expected cost is within an $\alpha(p)$-factor of the optimal one. Formally, for any prior p, an $\alpha(p)$-approximate ($\alpha(p) \geq 1$) algorithm A outputs a policy $A(p)$ satisfying

$$c_p^{\text{avg}}(A(p)) \leq \alpha(p) \min_\pi c_p^{\text{avg}}(\pi).$$

Note that $\alpha(p)$ may depend on the prior p. When $\alpha(p) = 1$, A is an exact algorithm. We have the following robustness result for the minimum cost problem in the average case.

Theorem 4. *Assume $c(\pi, h)$ is upper bounded by a constant K for all π, h. If A is an $\alpha(p)$-approximate algorithm, then for any true prior p_0 and any perturbed prior p_1,*

$$c_{p_0}^{avg}(A(p_1)) \leq \alpha(p_1) \min_\pi c_{p_0}^{avg}(\pi) + (\alpha(p_1)+1)K\|p_1 - p_0\|.$$

The assumption $c(\pi, h) \leq K$ for all π, h is reasonable since $c(\pi, h) \leq |\mathcal{X}|$ for all π, h. When $\mathcal{H}$ is small, K can be much smaller than $|\mathcal{X}|$.

Application to Generalized Binary Search: Theorem 4 implies the robustness of the generalized binary search algorithm, which is known to be $(\ln \frac{1}{\min_h p[h]} + 1)$-approximate (Golovin and Krause 2011). The result is stated in the corollary below. By taking $p_1 = p_0$, this corollary can recover the previous result by Golovin and Krause (2011) for the generalized binary search algorithm.

Corollary 5. *Assume $c(\pi, h)$ is upper bounded by K for all π, h. If A is the generalized binary search algorithm, then for any true prior p_0 and any perturbed prior p_1,*

$$c_{p_0}^{avg}(A(p_1)) \leq \left(\ln \frac{1}{\min_h p_1[h]} + 1\right) \min_\pi c_{p_0}^{avg}(\pi) + \left(\ln \frac{1}{\min_h p_1[h]} + 2\right) K\|p_1 - p_0\|.$$

Theorem 4 can also provide the robustness of algorithms for problems other than AL. For instance, it can provide the robustness of the RAId algorithm for the adaptive informative path planning problem (Lim, Hsu, and Lee 2015).

5 Mixture Prior

Let us consider methods that minimize a regularized loss. These methods are commonly used and known to be equivalent to finding the maximum a posteriori hypothesis with an appropriate prior. In practice, the best regularization constant is usually unknown, and a common technique (in passive learning) is to split the available data set into a training and a validation set, which is used to select the best regularization constant based on performance of the algorithm trained on the training set. As this method is effective in practice, we construct a Bayesian version and study its performance, particularly the robustness, when used with AL. We assume that we have a candidate set of prior distributions corresponding to different regularization constants, and the true hypothesis is randomly generated by first randomly selecting a distribution and then selecting a hypothesis using that distribution. This corresponds to assuming that the prior distribution is the mixture distribution. For simplicity, we consider the uniform mixture in this work.

First, we note that optimizing the expected cost of the mixture directly has a lower expected cost than trying to separately identify the appropriate component (corresponding to using a validation set in passive learning) and the best hypothesis given the component (corresponding to using the training set). Hence, we would expect the method to perform favorably in comparison to passive learning when the mixture prior is the true prior.

Results in earlier sections assure us that the method is near optimal when the mixture prior is incorrect but generates hypotheses with probabilities similar to the true prior. What if the true prior is far from the mixture distribution in the ℓ_1 distance? In particular, we are interested in the case where the true distribution is one of the mixture components, rather than the mixture itself. The following theorem provides bounds on the performance in such cases. We note that the theorem holds for general priors that may vary in form (e.g., with different probability mass functions) and is not restricted to priors corresponding to regularization constants.

The bounds show that the performance of the mixture is competitive with that of the optimal algorithm, although the constant can be large if some hypotheses have small probabilities under the true distribution. We also provide an absolute bound (instead of competitive bound) which may be more informative in cases where there are hypotheses with small probabilities. The bound (the first bound in Theorem 5) shows that the expected cost is within a constant factor of the optimal expected cost of the mixture, which is the expected cost we would have to pay if our model was correct. The optimal expected cost of the mixture is in turn better than the expected cost of any two-stage identification procedure that identifies the component and the hypothesis given the component separately, assuming the expectation is taken with respect to the mixture.

Theorem 5. *If A is an $\alpha(p)$-approximate algorithm for the minimum cost problem, then for any true prior p_0 and any k component uniform mixture prior $p_1 = \sum_{i=1}^k \frac{1}{k} p_{1,i}$ such that $p_0 \in \{p_{1,i}\}_{i=1}^k$, we have*

$$c_{p_0}^{avg}(A(p_1)) \leq k\alpha(p_1) \min_\pi c_{p_1}^{avg}(\pi), \text{ and}$$

$$c_{p_0}^{avg}(A(p_1)) \leq \alpha(p_1)\Big(\frac{k-1}{\min_h p_0[h]} + 1\Big) \min_\pi c_{p_0}^{avg}(\pi).$$

As a result, if A is generalized binary search, then

$$c_{p_0}^{avg}(A(p_1)) \leq k\Big(\ln \frac{k}{\min_h p_0[h]} + 1\Big) \min_\pi c_{p_1}^{avg}(\pi), \text{ and}$$

$$c_{p_0}^{avg}(A(p_1)) \leq \Big(\ln \frac{k}{\min_h p_0[h]} + 1\Big)\Big(\frac{k-1}{\min_h p_0[h]} + 1\Big) \min_\pi c_{p_0}^{avg}(\pi).$$

The algorithm for greedy AL with the mixture model is shown in Algorithm 1. In the algorithm, the unlabeled example x^* can be chosen using any AL criterion. The criterion can be computed from the weights and posteriors obtained from the previous iteration. For instance, if the maximum Gibbs error algorithm is used, then at iteration t, we have $x^* = \arg\max_x \mathbb{E}_{y \sim p[\cdot; x]}[1 - p[y; x]]$, where $p[y; x] = \sum_{i=1}^n w_{t-1}^i p_{t-1}^i[y; x]$. After x^* is chosen, we query its label y^* and update the new weights and posteriors, which are always normalized so that $\sum_i w_t^i = 1$ for all

Table 1: AUCs of maximum Gibbs error algorithm with $1/\sigma^2 = 0.01, 0.1, 1, 10$ and the mixture prior model on 20 Newsgroups data set (upper half) and UCI data set (lower half). Double asterisks (**) indicate the best score, while an asterisk (*) indicates the second best score on a row (without the last column). The last column shows the AUCs of passive learning with the mixture prior model for comparison.

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 (20 Newsgroups)	83.37**	82.52	79.02	73.48	82.71*	78.86
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 (UCI)	77.05**	76.43	74.64	72.72	76.59*	74.06

Algorithm 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.

$p_0^i \leftarrow p^i$; $\quad w_0^i \leftarrow w^i$; $\quad$ for all $i = 1, 2, \ldots, n$;
for $t = 1$ **to** k **do**
  Choose an unlabeled example x^* based on an AL criterion;
  $y^* \leftarrow$ Query-label(x^*);
  Update and normalize weights:
    $w_t^i \propto w_{t-1}^i \, p_{t-1}^i[y^*; x^*]$ $\quad$ for all $i = 1, 2, \ldots, n$;
  Update each posterior individually using Bayes' rule:
    $p_t^i[h] \propto p_{t-1}^i[h] \, \mathbb{P}[h(x^*) = y^* \mid h]$
      for each $i = 1, 2, \ldots, n$ and $h \in \mathcal{H}$;
end for
return $\{p_k^1, p_k^2, \ldots, p_k^n\}$ and $\{w_k^1, w_k^2, \ldots, w_k^n\}$;

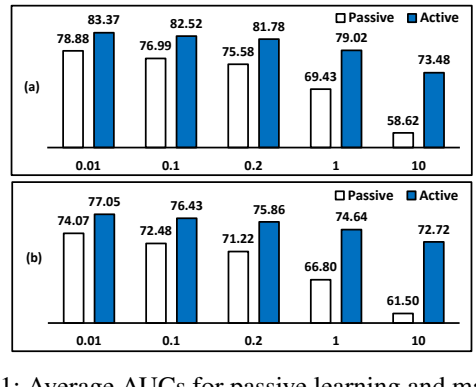

Figure 1: Average AUCs for passive learning and maximum Gibbs error AL algorithms with $1/\sigma^2 = 0.01$, 0.1, 0.2, 1, and 10 on the 20 Newsgroups (a) and UCI (b) data sets.

t and $\sum_h p_t^i[h] = 1$ for all i and t. The algorithm returns the final weights and posteriors that can be used to make predictions on 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 1 does not require the hypotheses to be deterministic. In fact, the algorithm can be used with probabilistic hypotheses where $\mathbb{P}[h(x) = y \mid h] \in [0, 1]$. We also note that computing $p_t^i[y; x]$ for a posterior p_t^i can be expensive. In this work, we approximate it using the MAP hypothesis. In particular, we assume $p_t^i[y; x] \approx p_{\text{MAP}}^i[y; x]$, the probability that x has label y according to the MAP hypothesis of the posterior p_t^i. This is used to approximate both the AL criterion and the predicted label of a new example.

6 Experiments

We report experimental results with different priors and the mixture prior. We use the logistic regression model with different L_2 regularizers, which are well-known to impose a Gaussian prior with mean zero and variance σ^2 on the parameter space. Thus, we can consider different priors by varying the variance σ^2 of the regularizer. We consider two experiments with the maximum Gibbs error algorithm. Since our data sets are all binary classification, this algorithm is equivalent to the least confidence and the maximum entropy algorithms. In our first experiment, we compare models that use different priors (equivalently, regularizers). In the second experiment, we run the uniform mixture prior model and compare it with models that use only one prior. For AL, we randomly choose the first 10 examples as a seed set. The scores are averaged over 100 runs of the experiments with different seed sets.

6.1 Experiment With Different Priors

We run maximum Gibbs error with $1/\sigma^2 = 0.01, 0.1, 0.2, 1, 10$ on tasks from the 20 Newsgroups and UCI data sets (Joachims 1996; Bache and Lichman 2013) shown in the first column of Table 1. Figure 1 shows the average areas under the

accuracy curves (AUC) on the first 150 selected examples for the different regularizers. Figure 1a and 1b give the average AUCs (computed on a separate test set) for the 20 Newsgroups and UCI data sets respectively. We also compare the scores for AL with passive learning.

From Figure 1, AL is better than passive learning for all the regularizers. When the regularizers are close to each other (e.g., $1/\sigma^2 = 0.1$ and 0.2), the corresponding scores tend to be close. When they are farther apart (e.g., $1/\sigma^2 = 0.1$ and 10), the scores also tend to be far from each other. In some sense, this confirms our results in previous sections.

6.2 Experiment With Mixture Prior

We investigate the performance of the mixture prior model proposed in Algorithm 1. For AL, it is often infeasible to use a validation set to choose the regularizers beforehand because we do not initially have any labeled data. So, using the mixture prior is a reasonable choice in this case.

We run the uniform mixture prior with regularizers $1/\sigma^2 = 0.01,\ 0.1,\ 1,\ 10$ and compare it with models that use only one of these regularizers. Table 1 shows the AUCs of the first 150 selected examples for these models on the 20 Newsgroups and the UCI data sets.

From the results, the mixture prior model achieves the second best AUCs for all tasks in the 20 Newsgroups data set. For the UCI data set, the model achieves the best score on Ionosphere 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 model performs reasonably well given the fact that we do not know which regularizer is the best to use for the data. We also note that if a bad regularizer is used (e.g., $1/\sigma^2 = 10$), AL may be even worse than passive learning with mixture prior.

7 Conclusion

We proved new robustness bounds for AL with perturbed priors that can be applied to various AL algorithms used in practice. We showed that if the utility is not Lipschitz, an optimal algorithm on perturbed priors may not be robust. Our results suggest that we should use a Lipschitz utility for AL if robustness is required. We also proved novel robustness bounds for a uniform mixture prior and showed experimentally that this prior is reasonable in practice.

Acknowledgement. We gratefully acknowledge the support of the Australian Research Council through an Australian Laureate Fellowship (FL110100281) and through the Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS), and of QUT through a Vice Chancellor's Research Fellowship. We also gratefully acknowledge the support of Singapore MOE AcRF Tier Two grant R-265-000-443-112.

References

Bache, K., and Lichman, M. 2013. UCI machine learning repository. *University of California – Irvine*.

Chen, Y., and Krause, A. 2013. Near-optimal batch mode active learning and adaptive submodular optimization. In *ICML*.

Culotta, A., and McCallum, A. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*.

Cuong, N. V.; Lee, W. S.; Ye, N.; Chai, K. M. A.; and Chieu, H. L. 2013. Active learning for probabilistic hypotheses using the maximum Gibbs error criterion. In *NIPS*.

Cuong, N. V.; Lee, W. S.; and Ye, N. 2014. Near-optimal adaptive pool-based active learning with general loss. In *UAI*.

Dasgupta, S. 2004. Analysis of a greedy active learning strategy. In *NIPS*.

Forsyth, R. 1990. PC/Beagle user's guide. *BUPA Medical Research Ltd*.

Golovin, D., and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*.

Gorman, R. P., and Sejnowski, T. J. 1988. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*.

Guillory, A., and Bilmes, J. 2010. Interactive submodular set cover. In *ICML*.

Hoi, S. C.; Jin, R.; Zhu, J.; and Lyu, M. R. 2006. Batch mode active learning and its application to medical image classification. In *ICML*.

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

Kohavi, R. 1996. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *KDD*.

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

Lim, Z. W.; Hsu, D.; and Lee, W. S. 2015. Adaptive informative path planning in metric spaces. *International Journal of Robotics Research*.

McCallum, A., and Nigam, K. 1998. Employing EM and pool-based active learning for text classification. In *ICML*.

Nowak, R. 2008. Generalized binary search. In *Annual Allerton Conference on Communication, Control, and Computing*.

Schlimmer, J. C. 1987. Concept acquisition through representational adjustment. *University of California – Irvine*.

Settles, B., and Craven, M. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*.

Settles, B. 2009. Active learning literature survey. *University of Wisconsin – Madison*.

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

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

Wei, K.; Iyer, R.; and Bilmes, J. 2015. Submodularity in data subset selection and active learning. In *ICML*.

Wolberg, W. H., and Mangasarian, O. L. 1990. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *PNAS*.