CSE599i: Online and Adaptive Machine Learning **Winter 2018**

Lecture 15: Binary Classification: Pool-based, Greedy Methods

Lecturer: Kevin Jamieson Scribes: G. Erion, S. Fatehi, K. Ehsani, E. Barzegary, O. Rafieian, N. Cano

1 Introduction

In this lecture, we focus on pool-based active learning methods for binary classification. In the previous lecture, we studied disagreement-based methods wherein unlabeled data comes as a stream and for each incoming data point the learner needs to decide whether to query its label or infer it. Pool-based methods, instead, are given a pool of unlabeled data points all at once, and the learner chooses the points to query from that pool. The active learner wants to identify the classifier with the lowest error with as few queries as possible. An example of pool-based active learning would be classifying images in a large repository into cats and dogs by only asking for labels for a small subset of images.

The simplest version of this problem is the *noiseless* case where each query always gives us the correct label. Among the most popular algorithms within this category we find generalized binary search (GBS henceforth) [1,2]. GBS, discussed in section 3, is a greedy algorithm for determining a binary-valued function through a sequence of strategically selected queries. In every step, the query that most evenly splits the hypotheses under consideration into two disjoint subsets is selected, i.e., a natural generalization of classic binary search.

In section 4 we consider a setting with *noisy* labels. For example, when using Mechanical Turk to manually annotate a dataset, the task may not be defined properly or Turkers may not understand the task. When label observations are noisy, noiseless GBS can perform very poorly as shown by Golovin et al. [3]. A modified version of GBS using a Bayesian model of noisy observations can handle noise in the *realizable* setting, where the true data-generating model lies within the hypothesis class being considered. Another approach to noisy realizable active learning is the Equivalence Class Edge Cutting algorithm (EC^2) of Golovin et al. [3] This algorithm reframes the problem as finding an equivalence class in which the correct hypothesis lies by cutting edges between hypothesis classes. This allows the use of adaptive submodularity, a natural diminishing returns property that generalizes the classical notion of submodularity to adaptive policies, to bound the number of queries needed to find the best hypothesis [3].

When the true data-generating model is not in the class of hypotheses being considered (e.g. nonlinear data being fit with linear classifiers), we are in the much more challenging *agnostic* case discussed in section 5. One approach in this setting is to transform the binary labeling task to a combinatorial bandit problem, which facilitates successful learning without dramatically increasing the sample complexity.

2 Preliminaries

The problem formulation in the setting of pool-based active learning is defined as follows: Suppose we have $\mathcal{X} = x_1, \ldots, x_m \in \mathbb{R}^d$ as input domain, and for each x_i there exists an unknown scalar μ_i influencing its label Y_i as follows. For $t = 1, 2 \ldots$:

- Player chooses $x_i \in \mathcal{X}$,
- Nature reveals $Y_i \in \{-1, 1\}; P(Y_i = 1|x_i) = (1 + \mu_i)/2$, where $\mu_i \in [-1, 1]$ and $E(Y_i) = \mu_i$

The objective is to identify $\text{sign}(\mu_i)$ as soon as possible, with the magnitude of μ_i impacting the level of error in our observations. When $|\mu_i| = 1$ for all μ_i, we are in the noiseless case with no uncertainty about labels.

As μ_i tends to zero, then $P(Y_i|x_i) = 1/2$, and the error rate becomes very high. If this is the case, we can ask for a label multiple times because of the pool-based, non-streaming setting. For example, in scenarios like Mechanical Turk crowdsourcing, μ_t may be close to 0 if each individual Turker does not produce reliable labels. In this case we may want to ask multiple Turkers for the same label.

2.1 Effective Hypothesis Class

The goal is to find the best hypothesis function h to label our data from a broader hypothesis class $\mathcal{H}$ (for example, linear classifiers). This can be seen as learning the true μ_i that generate the data labels with as few queries as possible. Note that if there is a hypothesis $h_* \in \mathcal{H}$ such that $h_*(x_i) = \text{sign}(\mu_i) \forall i$, we are in the *realizable* case where the true data-generating model is in our hypothesis class. If we have no such guarantee, we are in the harder *agnostic* case. While the hypothesis class may be an infinite set of functions, if we assume the data are finite, we can represent each hypothesis by its labeling of the data x_i and consider hypotheses which produce equivalent labelings to be the same hypothesis. This gives us a finite effective hypothesis class $\hat{\mathcal{H}}$ defined as

$$\hat{\mathcal{H}} \approx \{(h(x_1), \ldots, h(x_m)) : h \in \mathcal{H}\} \tag{1}$$

where we identify a given element of $\hat{\mathcal{H}}$ as $\hat{h} = (h(x_1), \ldots, h(x_m))$ for some $h \in \mathcal{H}$.

The size of the effective hypothesis class can be quantified in two ways. First, since there are m points with binary labels, there may be no more than 2^m possible labelings, and thus no more than 2^m effective hypotheses. Second, we can use the VC dimension d from Vapnik–Chervonenkis theory, defined as the maximum number of points for which we can always find a perfect classifier in $\mathcal{H}$ for any labeling of the d points.

Sauer's lemma [4] relates VC dimension d of a hypothesis space $\mathcal{H}$ over m points to the size of the effective hypothesis class $\hat{\mathcal{H}}$ as follows:

$$|\hat{\mathcal{H}}| \leq \sum_{i=0}^{d} \binom{m}{i} \leq (1+m)^d = O(m^d)$$

where the first inequality is the statement of Sauer's lemma and the second inequality holds for any sum of binomial coefficients. This result implies that we can classify a pool of unlabeled data points in at most $O(m^d)$ ways, which bounds the size of the effective hypothesis space $\hat{\mathcal{H}}$ at $O(m^d)$.

2.2 Search strategy

Any deterministic search strategy can be represented as a binary *query tree* T where internal nodes are queries and leaves are elements of $\hat{\mathcal{H}}$, as depicted in Figure 1.

This idea was first introduced as *extended teaching dimension* by Hegeds [5] and later studied by Hanneke [6] with the aim of investigating the query complexity of learning a concept class over a finite domain with membership queries and quantifying the minimum number of queries we have to ask in order to reduce the hypothesis class by half.

In a noiseless setting, the process goes as follows: we choose a point and see its label, then we choose another point and see its label, and so on. We can view this process as building a tree, where each internal node in the tree is a query point, and the leaves are hypotheses $h_1, h_2, \ldots$. If we uniformly draw a hypothesis from our class $\mathcal{H}$, the expected path length down the tree representing $\mathcal{H}$ is the number of labels we need. If we can build an "optimal" tree deterministically offline (i.e., before any querying) that minimizes the expected path length, we could then achieve the optimal label complexity. However, building and searching this tree is NP-hard and intractable to compute. Despite this intractability, there exists a heuristic algorithm called Generalized Binary Search (GBS) that can get a constant approximation ratio in favorable conditions.

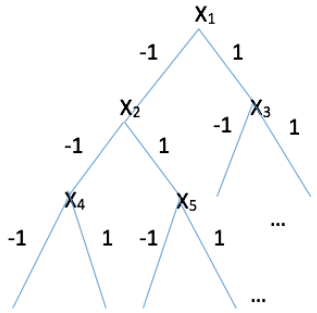


Figure 1: Deterministic Search Tree

2.3 Distribution over hypotheses

In many cases it will be useful to think about the *probability of a hypothesis* h, which we will refer to as $P(h)$. In the average-case analysis of generalized binary search in section 3, we want to find the average runtime of an algorithm with respect to the distribution over hypotheses $P(h)$. In the Bayesian analysis of noisy generalized binary search in 4.1, we want to start with a prior $P(h)$ over the correct hypothesis, and update it to generate a new posterior $P_t(h)$ at each time step t.

3 Generalized Binary Search

To begin, we discuss the simplest case of active learning for binary classification. In this noiseless case, $|\mu_i| = 1$ for all data x_i and every query produces a completely accurate label, and. We follow the work of Dasgupta [1] and start with a 1-dimensional example illustrating the promise of active learning, followed by a 2-dimensional example illustrating why the worst-case performance of active learning can be quite poor. We then develop the *generalized binary search* (GBS) algorithm and bound the number of samples it needs on average.

3.1 Motivation: 1D Binary Search

A very simple example of why active learning might help is in binary classification of linearly separable points in 1 dimension. Suppose we have points $x_1, \ldots, x_m \in \mathbb{R}^1$ and we know there exists a threshold w^* such that $Y(x) = \text{sgn}(x - w^*)$, as in Figure 2.

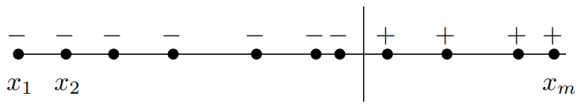


Figure 2: 1D Binary Search (Dasgupta [1])

All points left to the threshold are labeled negative, and all to the right are labeled positive. As shown in the previous lecture, the number of points queried in a fixed, nonadaptive strategy like uniform sampling to achieve error ϵ is of $O(\frac{1}{\epsilon})$. However, we can do substantially better than this if we employ an active learning scheme that intelligently chooses which labels to query. In particular, the problem of finding the point at which the transition occurs can be achieved by binary searching on the m points, which results in a sample complexity of $O(\log m)$ labels. The reason for this efficient search is that we can use the ordering of points on the real line to cut the search space in half at every step.

3.2 Worst-case Intractability of Higher Dimensions

Unfortunately, when the number of dimensions is 2 or greater, there are target hypotheses in $\hat{\mathcal{H}}$ that can not be identified without querying every data point in the dataset. An example comes from Dasgupta [1], which discusses the worst-case complexity of generalized binary search and introduces an average-case analysis.

We show that active learning may provide no benefit even in a two-dimensional case, let alone higher dimensions, in Proposition 1.

Proposition 1. *Let $\mathcal{H}$ be the hypothesis class of half spaces such that $\forall h \in \mathcal{H}; |h^+ \cap \mathcal{X}| \leq 1$, where $h \subseteq \mathbb{R}^2$ and $h^+ = \{x \in \mathcal{X} : h(x) = 1\}$. For any set of m distinct data points on the perimeter of the unit circle, there is always some labeling such that the correct hypothesis in $\mathcal{H}$ cannot be identified without querying all m labels.*

Proposition 1 considers half spaces in $\mathbb{R}^2$, which are simply linear classifiers that split the space in half on either side of a line as in the 1-dimensional example we saw before. The number of classifiers that label exactly 1 or 0 points positive is $|\hat{\mathcal{H}}| = m + 1$. Any active learning method will, at worst, require querying all m points in this scenario. To see this, call the noiseless labelings L_0 and $L_1, \ldots, L_m$, where L_0 is the hypothesis that labels all points negative, whereas L_i labels x_i positive and all other points negative.

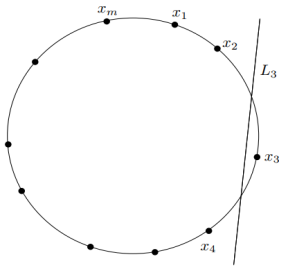


Figure 3: 2D worst-case for generalized binary search. [1] The labeling L_3 labels x_i negative for all $i \neq 3$, and labels x_3 positive. Suppose we tested every point except for x_3; we would still have to query x_3 to determine whether L_3 or L_0 was the correct labeling.

In Figure 3, L_3 labels all points negative except for x_3, so sampling x_3 is crucial in order to choose the true hypothesis between L_3 and L_0. A similar argument holds for all the data points x_i. In other words, we have to sample every single point on the circle in order to find the optimal hypothesis. The one-dimensional case in Figure 2, on the real line, was easier because labeling a point on the line also fixes the labels of all points to its left or right. However, in the worst case of a unit circle like Figure 3, even labeling every single other point may not necessarily force the label of x_i.

3.3 Average-case Model

We have shown that in ideal cases like 1-dimensional binary search, active learning can achieve $\log_2 |\hat{\mathcal{H}}|$ queries. In subsection 2.1 $|\hat{\mathcal{H}}|$ was shown to be $O(m^d)$, which gives us $d \log m$ queries. However, in the worst case, such as the 2-dimensional unit circle, active learning may require up to order m queries.

Even though there are worst-case examples where active learning fails, we may hope that the optimal h can be identified using close to $\log_2 |\hat{\mathcal{H}}|$ queries *on average* with respect to some distribution $P(h)$ over target hypotheses in $\hat{\mathcal{H}}$. The intuition is that each label reveals at most 1 bit of information and since it only takes $\log_2 |\hat{\mathcal{H}}|$ bits to encode a hypothesis, there should exist a strategy that can query labels and find the hypothesis $\hat{\mathcal{H}}$ using just $\log_2 |\hat{\mathcal{H}}|$ queries.

The main result by Dasgupta [1], which we walk through in subsection 3.4, shows that if the distribution $P(h)$ is uniform over $\hat{\mathcal{H}}$, and we greedily ask for the labels which most evenly divide the current effective

version space, then the expected number of labels needed to identify a target hypothesis drawn from $P(h)$ is at most $O(\log |\hat{\mathcal{H}}|)$ times any other strategy.

3.4 Greedy Strategy

In this section, we describe the greedy strategy, then analyze its effectiveness by putting upper and lower bounds on the number of queries it requires.

Consider our effective hypothesis class $\hat{\mathcal{H}}$ as defined in (1) and let P be any distribution over $\hat{\mathcal{H}}$. Recall that each search strategy can be represented as a binary tree where internal nodes are queries and leaves are hypotheses of $\hat{\mathcal{H}}$. Consider any tree T whose leaves (hypotheses) are chosen from P. The *quality* of this tree is its expected depth with respect to P:

$$Q(T, P) = \sum_{h \in \hat{\mathcal{H}}} P(h) \cdot (\#\text{ labels needed for } h) = \sum_{h \in \hat{\mathcal{H}}} P(h) \cdot \text{leaf\_depth}(h) \tag{2}$$

We are interested in determining whether there is always a tree T of average depth $o(m)$. Unfortunately, as shown in Proposition 2, we cannot always achieve an average depth of $o(m)$. We refer the interested reader to Dasgupta [1] for the proof.

Proposition 2. *Pick any $d \geq 2$ and any $m \geq 2d$. There is an input space $\mathcal{X}$ of size m and a hypothesis class $\mathcal{H}$ of VC dimension d, defined on domain $\mathcal{X}$, with the following property: if $P(h)$ is chosen to be uniform over $\mathcal{H} = \hat{\mathcal{H}}$, then any query tree T has $Q(T, P) \geq \frac{m}{8}$.*

So far, the benefit of pool-based active learning for binary classification has varied based upon the particular hypothesis class and the pool of unlabeled data points. Thus, we are interested in a querying strategy that makes close to the minimum number of queries, whatever that minimum is. A natural approach is the following greedy strategy. We maintain a *version space* $S \subseteq \hat{\mathcal{H}}$ which contains the hypotheses that are being considered. $S_0 = \hat{\mathcal{H}}$, while $S_1 \subset \hat{\mathcal{H}}$, and at the end of the algorithm we should have that $S_t = \{h_*\}$, a singleton set containing the target hypothesis. Points are queried by always choosing the x_i whose label is most uncertain among hypotheses in $\mathcal{H}$:

> *Let $S \subseteq \hat{\mathcal{H}}$ be the current version space. For each unlabeled x_i, let S_i^+ be the hypotheses that label it positive and S_i^- those that label it negative. Query the label of x_i for which these sets are nearest to being equal in mass with respect to $P(h)$, that is, query the x_i for which $\min\{P(S_i^+), P(S_i^-)\}$ is largest.*

The key step of Algorithm 1, Generalized Binary Search, is step 3. The sum in this step is minimized to 0 when half of the hypotheses by probability mass label x_i positive, and half negative. Thus the x_t produced is the point whose label will greedily rule out as many hypotheses as possible.

Algorithm 1 Generalized Binary Search Algorithm (GBS)

1: Initialize: $S_0 = \hat{\mathcal{H}}$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad x_t = \arg\min_x |\sum_{h \in S_t} P(h)h(x)|$
4: $\quad$ Observe $h_*(x_t)$ for unknown $h_* \in \hat{\mathcal{H}}$
5: $\quad$ Update S_t
6: **end for**

The rest of this section will show that the simple greedy strategy is almost as good at minimizing queries as any other strategy. For the GBS algorithm shown in Algorithm 1, in the noiseless setting, we will show the number of queries required to obtain h_* is at most $(\log |\hat{\mathcal{H}}|)OPT$, where OPT is the expected number of queries performed by the optimal strategy.

If $\exists c < 1$ such that $\min_x \max_{S \subseteq \mathcal{H}} |\sum_{h \in S} h(x)|/|S| < c$, then $|S_t| \leq c|S_{t-1}| \leq c^t|S_0|$ where $S_0 = \hat{\mathcal{H}}$, which gives us an upper bound on the total number of queries we need to ask, $t \leq \frac{\log |\hat{\mathcal{H}}|}{\log_2 c}$. However, sometimes it

is hard to prove that a constant $c < 1$ exists. Instead, we can only say that $c = 1/|\hat{\mathcal{H}}|$, which is very slow. This is why we look to the average-case model and the expected number of queries.

3.4.1 Lower Bounds

In this section we want to derive lower bounds on the greedy scheme. The greedy approach builds a query tree top-down, trying to reduce the number of hypotheses being considered as quickly as possible. However, this may not be optimal because each query looks only one step ahead; it does not consider how a query may reshape the search space and reduce the quality of later queries. This could result in slower elimination of hypotheses than alternative learning strategies. Proposition 3 shows a lower bound on a greedy active learner assuming P is uniform. For instance, $\hat{H}$ might consist of several tight clusters, where binary search is rapid within a given cluster. However, it is possible that the optimal strategy must first slowly cut down the version space to one of these subregions. This process, though ultimately optimal, could initially be slower at eliminating hypotheses than greedy alternatives. A concrete example of this type gives rise to the following lower bound; we refer the interested reader to the proofs in Dasgupta [1].

Proposition 3. *For any integer $n \geq 2^4$, there is a concept class $\hat{\mathcal{H}}_n$ of size n with the following property: under uniform P, the optimal tree has average height at most $q_n = \Theta(\log n)$, but the greedy active learning strategy produces a tree of average height $\Omega(q_n \cdot \frac{\log n}{\log \log n})$*

The above lower bound is derived assuming π is uniform. Proposition 4 follows a similar analysis, but assumes π is not uniform.

Proposition 4. *Pick any $n \geq 2$. There exists a hypothesis class $\hat{\mathcal{H}}$ with $2n + 1$ elements and a distribution P over $\hat{\mathcal{H}}$ such that (a) P ranges in value from $\frac{1}{2}$ to $\frac{1}{2^{n+1}}$; (b) the optimal query tree has average depth less than 3; (c) the greedy query tree has average depth at least $\frac{n}{2}$.*

3.4.2 Upper Bounds

We next seek to upper bound the number of queries required by the greedy algorithm. We follow [1] and relate the quality of a tree as defined in 3.4 to the amount Δ by which it shrinks the version space.

Assume the candidate hypotheses have shrunk to some version space $S \subseteq \hat{\mathcal{H}}$ and the next possible query is x_j. Now we want to analyze what probability mass P is going to be eliminated on average by this query. Let S^+ be the subset of S that labels x_j positive and S^- be the subset that labels x_j negative. On average the following probability mass is eliminated:

$$\Delta = \frac{P(S^+)}{P(S)}P(S^-) + \frac{P(S^-)}{P(S)}P(S^+) = \frac{2P(S^+)P(S^-)}{P(S)}.$$

It can be easily shown that the shrinkage is monotonically decreasing; we refer the interested reader to the proofs in Dasgupta [1].

From the definition of shrinkage, one might think that if all queries result in at most Δ shrinkage, then we need at least $P(S)/\Delta$ more queries. However, this is not completely true because if there are only two hypotheses left in the current version space S, then only one query is needed, regardless of $P(S)$.

If there are many hypotheses with significant mass left in the current space S, then the shrinkage effect $P(S)/\Delta$ dominates; afterwards, the second effect does. The concept of *collision probability* helps us address this transition. Let v be a distribution over support Z, then the collision probability defines the chance that two random draws from v are identical:

$$CP(v) = \sum_{z \in Z} v(z)^2$$

We can now write a single bound that relates the quality of a tree with the amount of shrinkage Δ, and thus, with the collision probability.

Lemma 1. *Suppose that every query shrinks $(\hat{\mathcal{H}}, P)$ by at most some $\Delta > 0$. Then for any $S \subseteq \mathcal{H}$, and any query tree T whose leaves include S*

$$Q(T, P_S) \geq \frac{P(S) \cdot (1 - CP(P_S))}{\Delta} \tag{3}$$

Proof. We prove this proposition using mathematical induction on $|S|$. First consider $|S| = 1$: we can easily verify $CP(P_s) = 1$ and thus the claim holds for $|S| = 1$. Next consider any $S \subset \hat{H}$ with $|S| > 1$. Let the next query for the optimal tree under P_s be at x_j. Querying here splits the space S into S^+ and S^- with masses $pP(S)$ and $(1-p)P(S)$, respectively. Our inductive hypothesis says that the lemma holds for the two subtrees, thus the average number of queries needed for S^+ and S^- are $P(S^+)(1 - CP(P_{S^+}))/\Delta$ and $P(S^-)(1 - CP(P_{S^-}))/\Delta$, respectively. Thus, the expected number of queries for P_S is at least:

$$\begin{aligned} & (1+p).P(S^+)(1 - CP(P_{S^+}))/\Delta + (1-p).P(S^-)(1 - CP(P_{S^-}))/\Delta \\ = \;& 1 + \frac{P(S)}{\Delta}\left\{p^2 + (1-p)^2 - p^2 CP(P_{S^+}) - (1-p)^2 CP(P_{S^-})\right\} \\ = \;& 1 + \frac{P(S)}{\Delta}\left\{1 - 2p(1-p) - CP(P_S)\right\} \end{aligned}$$

We know $2p(1-p)P(S) \leq \Delta$, as any query shrinks (S, P) by at most Δ. From this and the above equality we conclude the expected number of queries needed for S is at least $P(S)(1 - CP(P(S)))/\Delta$, which completes the proof. □

If $P(S)$ has small collision probability, then we expect to exist some query that can eliminate a reasonable amount of space S. By rearranging terms in (3), we get Corollary 1, which shows that such a query exists:

Corollary 1. *Pick any $S \subseteq \hat{\mathcal{H}}$ and any tree T whose leaves include all of S. Then there must exist a query which shrinks (S, P_S) by at least $\frac{(1 - CP(P_S))}{Q(T, P_S)}$.*

The next lemma attempts to use this result to bound the sample complexity of GBS. The only problem is that if P_S has high collision probability, then Corollary 1 doesn't guarantee much shrinkage of the sample space: S could shrink an insignificant amount during the next greedy queries. However, the following proof explicitly addresses the case where there is a hypothesis h_0 causing high collision probability by occupying $\geq \frac{1}{2}$ of the probability mass. Within roughly the number of iterations that the optimal tree needs to identify h_0, the greedy active learner will either reject h_0 or identify it as the target. If h_0 is the target hypothesis, we are done. Otherwise, by the time h_0 is rejected, the remaining search space S will have shrunk by enough that a logarithmic bound on the number of queries still exists.

Lemma 2. *Let T^* denote any particular query tree for P (for instance, the optimal tree), and let T be the greedily-constructed query tree. For any $S \subseteq \hat{\mathcal{H}}$ which corresponds to a subtree T_S of T*

$$Q(T_S, P_S) \leq 4Q(T^*, P_S) \ln \frac{P(S)}{\min_{h \in S} P(h)}$$

Proof. We use mathematical induction for completing the proof. First consider $|S| = 1$: the claim holds trivially. Next consider $S \subset \hat{H}$ where $|S| > 1$. We do the proof for the following two cases:

Case i) $CP(P_S) < 1/2$ Assume greedy query S splits S into S^+ and S^- where $p = P(S^+)/P(S)$. In the remainder of the proof we use $Q^*(A)$ as a shorthand for $Q(T^*, P_A)$ and $\min(A)$ as a shorthand for $\min_{h \in A} P(h)$ where A is some subset of the hypothesis space. The number of queries needed for tree T_s is

$$Q(T_S, P_S) \quad = 1 + pQ(T_{S^+}, P_{S^+}) + (1-p)Q(T_{S^-}, P_{S^-})$$

From the induction hypothesis we have the following inequality:

$$\begin{aligned} Q(T_S, P_S) &\leq 1 + 4pQ^*(S^+) \ln \frac{P(S^+)}{\min(S^+)} + 4(1-p)Q^*(S^-) \ln \frac{P(S^-)}{\min(S^-)} \\ &\leq 1 + 4\left(pQ^*(S^+) + (1-p)Q^*(S^-)\right) \ln \frac{\max\{P(S^+), P(S^-)\}}{\min(S)} \\ &= 1 + 4Q^*(S) \ln \frac{P(S)}{\min(S)} - 4Q^*(S) \ln \max\{p, 1-p\} \end{aligned}$$

From inequality $\ln(1-x) \leq -x$, we obtain the following inequality:

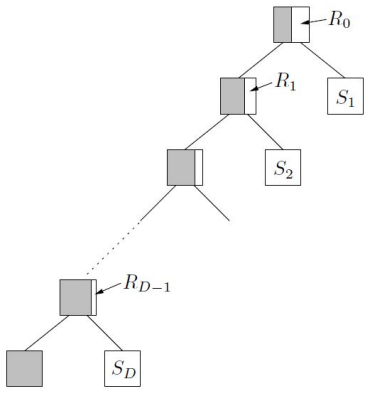
$$Q(T_S, P_S) \leq 1 + 4Q^*(S) \ln \frac{P(S)}{\min(S)} - 4Q^*(S) \min\{p, 1-p\}$$

As $CP(P_S) \leq 1/2$, from Corollary 1 we have

$$\min\{p, 1-p\} \geq p(1-p) \geq \frac{1 - CP(P_S)}{2Q^*(S)} \geq \frac{1}{4Q^*(S)}.$$

If we input the above inequality to the inequality on $Q(T_S, P_S)$ we get $Q(T_S, P_S) \leq 4Q^*(S) \ln \frac{P(S)}{\min(S)}$ and this completes the proof for case i.

Case ii) $CP(P_S) > 1/2$: As $CP(P_S) \leq \max_{h \in S} P_S(h)$, we can conclude there is a single hypothesis h_0 that covers more than half of the probability mass S. Now consider the figure below which assumes that



hypothesis h_0 is located at depth D in the greedy tree T_S [1]. Assume at depth $d < D$ of T_S, R_d is the set of hypotheses that have not been distinguished from h_0. Thus, at depth d, the greedy tree is trying to remove as much probability mass of R_d as possible. At the next step, the greedy tree will remove some S_{d+1} and therefore we have $R_d = R_{d+1} \cup S_{d+1}$.

Note that, by definition hypothesis h_0 can be identified using $Q^*(h_0)$ queries and as a result, it must always be possible to cut off $1/Q^*(h_0)$ of R_d:

$$P(R_{d+1}) = P(R_d) - P(S_{d+1}) \leq P(R_d) - \frac{P(R_d)}{Q^*(h_0)}$$

From the above inequality we conclude

$$P(R_d) \leq (1 - 1/Q^*(h_0))^d \, P(R_0) \leq P(R_0)\mathbf{e}^{-d/Q^*(h_0)}.$$

From the above inequality we conclude $D \leq Q^*(h_0) \ln (P(R_0)/\min(R_0))$.

If we consider h_0 separately from other hypothesis in space S we can split the expected tree depth into a weighted sum of the depth of h_0 and the depth of all other subtrees:

$$Q(T_S, P_S) \quad = \frac{1}{P(S)} \left\{ P(h_0)D + \sum_{d=1}^{D} P(S_d) \left(d + Q(T_s, P_{S_d^+}) \right) \right\}$$

By applying the inductive hypothesis to the rightmost sum, we get the following inequality:

$$\begin{aligned} Q(T_S, P_S) \quad &= \frac{1}{P(S)} \left\{ P(h_0)D + \sum_{d=1}^{D} P(S_d)d + \sum_{d=1}^{D} P(S_d) 4Q^*(S_d) \ln \frac{P(S_d)}{\min(S_d)} \right\} \\ &\leq \frac{1}{P(S)} \left\{ P(h_0)D + \sum_{d=1}^{D} P(S_d)d + 4P(R_0)Q^*(R_0) \ln \frac{P(R_0)}{\min(R_0)} \right\} \end{aligned}$$

where the last step follows because the mass cut out at any given split of the tree must be less than or equal to R_0, the total non-h_0 mass. Note that at each depth d at least $1/Q^*(h_0)$ fraction of R_d is cut off. Thus we have $\sum_d P(S_d)d \leq P(R_0)Q^*(h_0)$. Combining this with the above upper bound on $Q(T_S, P_S)$ we have:

$$\begin{aligned} Q(T_S, P_S) \quad &\leq \frac{1}{P(S)} \left\{ P(h_0)D + P(R_0)Q^*(h_0) + 4P(R_0)Q^*(R_0) \ln \frac{P(R_0)}{\min(R_0)} \right\} \\ &\leq \frac{1}{P(S)} \left\{ P(h_0)Q^*(h_0) \ln \frac{P(R_0)}{\min(R_0)} + P(R_0)Q^*(h_0) + 4P(R_0)Q^*(R_0) \ln \frac{P(R_0)}{\min(R_0)} \right\} \\ &\leq \frac{1}{P(S)} \left\{ P(h_0)Q^*(h_0) \ln \frac{P(S)}{\min(S)} + P(R_0)Q^*(h_0) + 4P(R_0)Q^*(R_0) \ln \frac{P(S)}{\min(S)} \right\} \end{aligned}$$

It must be true that $P(S) \geq 2\min(S)$, which lets us conclude

$$\begin{aligned} Q(T_S, P_S) \quad &\leq \frac{1}{P(S)} \left\{ 4P(h_0)Q^*(h_0) + 4P(R_0)Q^*(R_0) \right\} \ln \frac{P(S)}{\min(S)} \\ &\leq 4Q^*(S) \ln \frac{P(S)}{\min(S)} \end{aligned}$$

where the last inequality comes from $P(h_0) \geq P(R_0)$ and this completes the proof. □

4 Noisy Realizable Active Learning

In this section, we introduce the problem of active learning with *noisy* observations. As before, we want to select the smallest number of expensive tests required to identify a hypothesis. However, in this case the tests may not be perfectly reliable and may mislabel points.

One way to formalize this problem is Bayesian experimental design [7], where one assumes a prior on the hypotheses as well as probabilistic assumptions on the outcomes of tests. The goal then is to determine the correct hypothesis while minimizing the cost of the experiments. In such setting, finding the optimal policy is NP-hard and also hard to approximate [8].

4.1 Bayesian Posterior Inference

We start by discussing how we can apply ideas from Bayesian posterior inference to adapt GBS to noisy settings. In the realizable setting, as discussed above, we make the rather strong assumption that the true data-generating hypothesis h_* is in our hypothesis class $\mathcal{H}$ and that the labels are noisy realizations of the predictions of h_*, Let the noise model be

$$Y_i = \begin{cases} h_*(x_i), & \text{w.p. } 1-\alpha \\ -h_*(x_i), & \text{w.p. } \alpha \end{cases}$$

where α is the noise rate. From the posterior inference literature, if we know we have this noise model, we can then infer the probability of h_* given values of Y_i. Thus, by doing posterior inference we can build a probability distribution, which results in $P_t(h)$ to be the probability that the hypothesis h is equal to h_* given all the noisy observations up to the current time. We show GBS for noisy observations in Algorithm 2, which is an adaptation of Algorithm 1 used in the noise-free setting.

Rather than starting with a set of hypotheses and eliminating them through queries, we start with a uniform prior P_0 over hypotheses. We choose the point x_t to query at each time t according to the exact same rule as GBS, maximizing disagreement among hypotheses weighted by the probability assigned to them by P_t. We then update the distribution P_t based on which hypotheses fit the observed label and continue. Eventually, we will have $P_t(h) \geq \frac{1}{2}$ for some $h \in \hat{\mathcal{H}}$, at which point we will report h as the best hypothesis.

Algorithm 2 Generalized Binary Search Algorithm for Noisy Observations

$P_0 =$ uniform over $\hat{\mathcal{H}}$
for $t = 1, 2, \ldots$ **do**
  $x_t = \arg\min_x |\sum_{h \in S_t} h(x) P_t(h)|$
  Observe $h_*(x_t)$ for unknown $h_* \in \hat{\mathcal{H}}$
  Update P_t
end for

4.2 Adaptive Submodularity

Another approach to noisy, realizable binary classification is to use the framework of adaptive submodularity. We start the section with some preliminaries, introduce the adaptive submodularity framework, then make some connections to GBS in the noiseless case. Finally, we address the noisy case by introducing the Equivalence Class Determination problem and the EC^2 algorithm proposed by Golovin et al. [3].

4.2.1 Preliminaries

Recall we want to distinguish among a set of hypothesis $\mathcal{H} = \{h_1, ..., h_n\}$. The adaptive submodularity framework introduces a cost $c(i)$ when we query each label Y_i; if this is uniform, we have the exact same goal as above of minimizing the total number of queries. Note also that our query policies, written as trees T, function as maps from the labels seen so far (a likely-incomplete vector of labels $\mathbf{y_t} = \{Y_{t=0} ... Y_t\}$ to a new index i to query. As before, we assume a distribution $P(h)$ over the hypotheses.

4.2.2 Framework

At first glance, the objective of eliminating as many hypotheses as possible with as few tests seems like a natural submodular optimization problem. Recall that a submodular optimization problem refers to optimizing a submodular function.

Let N be a finite ground set and $f : 2^N \to \mathbb{R}$. Then f is submodular if for all $A, B \subset N$

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

Another definition commonly used in the literature has to do with marginals. Let $f_A(i) = f(A+i) - f(A)$ as the marginal value of i with respect to A. Then f is submodular if for all $A \subset B \subset N$ and $i \in N \notin B$, $f_A(i) \geq f_B(i)$.

In particular, the submodular approximation for the max-cover problem (find k sets whose union covers as many elements as possible) seems analogous to the task of finding a small number of tests whose union rules out as many hypotheses as possible. The key difference is that in active learning, we don't know which hypotheses a test will rule out before we choose it—if 99% of our hypotheses predict $x_i = 1$, testing and

finding $x_i = 1$ rules out almost none of our hypotheses, but finding $x_i = 0$ rules out almost all of them, and we don't know in advance which it will be.

Instead of blindly applying a greedy approximate algorithm, we have to create an extension of submodularity for the case where the marginal increase given by a set is not fixed but is a random variable. A way to implement this extension is by considering the *expected* increase when a set is added, which gives us the concept of "adaptive submodularity".

The adaptive submodularity framework, introduced by Golovin et al. [9], is used in the design and analysis of many active learning algorithms, and allows us to borrow many results from the submodular function optimization setting to the adaptive one.

Call $\mathbf{x}_\mathcal{A}$ a subvector of $\mathbf{x}_\mathcal{B}$ if $\mathcal{A} \subseteq \mathcal{B}$ and $P(\mathbf{x}_\mathcal{B}|\mathbf{x}_\mathcal{A}) > 0$. We denote this as $\mathbf{x}_\mathcal{A} \prec \mathbf{x}_\mathcal{B}$. A function $f : 2^\mathcal{T}$ x $\mathcal{H}$ is called *adaptive submodular* with respect to a distribution P, if for any $\mathbf{x}_\mathcal{A} \prec \mathbf{x}_\mathcal{B}$ and any new element q we have that $\Delta(q|\mathbf{x}_\mathcal{A}) \geq \Delta(q|\mathbf{x}_\mathcal{B})$, where

$$\Delta(q|\mathbf{x}_\mathcal{A}) = \mathbb{E}_H[f(\mathcal{A} \cup \{q\}, H) - f(\mathcal{A}, H)|\mathbf{x}_\mathcal{A}] \tag{4}$$

is the expected marginal benefit of adding a new element q. In other words, f is adaptive submodular if the expected marginal benefits $\Delta(q|\mathbf{x}_\mathcal{A})$ of a new element q can only decrease as we add more elements.

f is *strongly adaptively monotone* w.r.t. P if observations never "hurt" with respect to the reward. More formally, for all $\mathcal{A}$, all $q \notin \mathcal{A}$, and all $x \in \mathcal{X}$, $\mathbb{E}_H[f(\mathcal{A}, H)|\mathbf{x}_\mathcal{A}] \leq \mathbb{E}_H[f(\mathcal{A} \cup \{q\}, H)|\mathbf{x}_\mathcal{A}, X_q = x]$.

In the language of active learning, q is a new query. For an adaptive submodular objective, the expected marginal benefit of adding a query q decreases with each query. An adaptively monotone objective means that, in expectation, each query moves the search closer to the optimal hypothesis, or at worst does not move it farther away.

A greedy algorithm with an adaptive submodular and strongly adaptive monotone objective function can play as follows:

$$q^* = \arg\max_{q \in \mathcal{T}} \Delta\{q|\mathbf{x}_\mathcal{A}\}/c(q) \tag{5}$$

The work by Golovin et al. [9] shows that if f is adaptive monotone and adaptive submodular, then the adaptive greedy algorithm achieves a $(1 - 1/e)$ approximation to the expected reward of the best policy, i.e.,

$$f(\pi^{\text{greedy}}) \geq (1 - 1/e) \max_\pi f(\pi).$$

4.2.3 Connection to Generalized Binary Search

For the noise-free setting, the GBS objective of reducing the version space probability mass is adaptive submodular and strongly adaptive submonotone. Thus by choosing queries according to (5), and by using adaptive submodular analysis (Theorem 1 from [3]) we can directly get the near-optimal expected cost bound of $c(T_{GBS}) \leq c(T^*)(ln(1/p_{min}) + 1)$, where $p_{min} = min_{h \in \mathcal{H}} P(h)$. Note that this removes the factor of 4 from the upper bound in the GBS analysis of equation 2.

Unfortunately, the noisy GBS objective isn't adaptive submodular because queries do not rule out hypotheses, only make them less likely. Since we can't use submodular analysis directly we have to take a different approach.

4.2.4 Reducing to Noiseless Classification

We can reduce the noisy problem to noiseless classification by generating extra "noisy" hypotheses. Recall that an infinite hypothesis class $\mathcal{H}$ can be reduced to a finite effective hypothesis class $\hat{\mathcal{H}}$ by considering hypotheses which predict the same label for every data point to be the same hypothesis, represented as $\hat{h} = \{h(x_1)...h(x_m)\}$ for any $h \in \hat{\mathcal{H}}$. That is, each hypothesis is specified by its vector of predictions on the data seen so far. In the noisy case, we assume that the data is generated by a true hypothesis h_* with

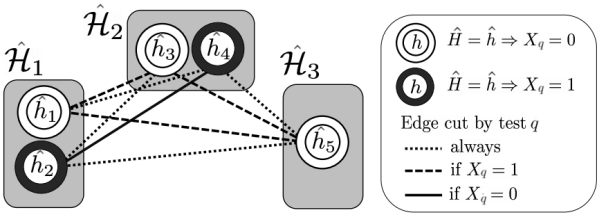


Figure 4: Equivalent Class Determination problem, where distinguishing hypotheses is viewed as disconnecting groups in a graph. Figure adapted from [3].

added noise, and introduce a random noise parameter $\theta \in \Theta$ that denotes exactly what realization of noise is added. We can now represent the hypothesis space as $\{\hat{h}(\theta) \forall (h, \theta) \in \hat{\mathcal{H}} \times \Theta\}$ that is, each noiseless $\hat{h}$ turns into a set of new "noisy" $\hat{h}'$, one for each possible addition of noise to $\hat{h}$'s predictions.

This process substantially increases the number of effective hypotheses being considered. We need some way to account for the fact that many of these hypotheses are *equivalent* in some way, i.e., we can separate them into *equivalence classes*, where all elements of an equivalence class are noisy versions of the same original hypothesis. Since we are not trying to find a single best noisy hypothesis but the best equivalence class, we can use the structure of the equivalence classes to narrow down to the correct answer much faster.

4.2.5 The Equivalence Class Determination Problem and the EC^2 Algorithm

In this section we introduce the general formulation of Bayesian active learning with noisy observations used by Golovin et al. [3] called *Equivalence Class Determination*. The core idea of this framework, developed by Dasgupta in [10], is to reframe differentiating hypotheses as cutting edges between them. We will use this model as a way to apply adaptive submodularity to a problem with noisy hypotheses; it has also been used to analyze noisy GBS by Bellala et al in [11].

The Equivalence Class Determination Problem (ECD) is formulated as follows (see Figure 4). Suppose the set of hypotheses $\hat{\mathcal{H}}$ is partitioned into a set of m equivalence classes such that $\hat{\mathcal{H}} = \bigcup_{i=1}^{m} \hat{\mathcal{H}}_i$. The goal is to determine in which class the true hypothesis lies.

In the Equivalence Class Edge Cutting (EC^2) algorithm, we define a set of edges $\mathcal{E} = \bigcup_{1 \le i < j \le m} \{(\hat{h}, \hat{h}') : \hat{h} \in \hat{\mathcal{H}}_i, \hat{h}' \in \hat{\mathcal{H}}_j\}$. These are the edges that must be cut, i.e., for any instance in this set at least one hypothesis must be ruled out. A query q under true hypothesis $\hat{h}_*$ is said to cut the edges $\mathcal{E}_q(\hat{h}_*) = \{(\hat{h}', \hat{h}'') : \hat{h}'(q) \neq \hat{h}(q)$ or $\hat{h}''(q) \neq \hat{h}(q)\}$. Figure 4, borrowed from [3], shows an example with the set of edges that must be cut, and depicts the effects of test i under different outcomes. We also define a weight function to prioritize which edges we cut $w : \mathcal{E} \to \mathbb{R}_{\ge 0}$ as $w(\hat{h}, \hat{h}') = P(\hat{h})\, P(\hat{h}')$.

The objective function that will then be greedily maximized is defined as follows:

$$f_{EC}(T, \hat{h}) = w\Big(\bigcup_{q : x_q \in \mathcal{X}} \mathcal{E}_q(\hat{h}) \Big) \tag{6}$$

In other words, our objective increases as more and more edge weight is removed with cuts. The objective f_{EC} is adaptive submodular and strongly adaptive submonotone. We can compute the marginal benefits for query q upon observations $\mathbf{y}_t$ using (4), where $f = f_{EC}$ and $\mathbf{x}_{\mathcal{A}} = \mathbf{y}_t$. For the proofs of why f_{EC} is adaptive submodular and strongly adaptive submonotone please refer to Appendix A of [3].

Now recall that in our noisy version of the active learning problem, each equivalence class is a set of noisy copies of a single hypothesis. We then draw edges between hypotheses with weights $w(\hat{h}, \hat{h}') = P(\hat{h})\, P(\hat{h}')$, and pick tests that disconnect as much of this edge weight at each step as possible. The goal is no longer to reduce the number of viable hypotheses to one, but to disconnect all edges connecting to an equivalence class. Once this process is complete, the original hypothesis corresponding to this equivalence class must

be the best one. The EC^2 algorithm uses the adaptive policy T_{EC} that, after observing $\mathbf{y}_t$, greedily selects the next query with (5). Finally, by using adaptive submodular analysis and the same theorem as before from [3], we can get the cost bound for EC^2:

$$c(T_{EC}) \leq c(T^*)(2\ln(1/p_{min}) + 1)$$

where $p_{min} = min_{h\in\mathcal{H}}P(h)$. Note that the constant factor here is only 2, not 4 as in the noiseless GBS analysis.

5 Agnostic Setting – Reduction to Bandits

In this section, we show how the noisy case can be transformed to a multi-armed bandit problem and thus solved even in the *agnostic* case, where the true data-generating model is not in the set of hypotheses being considered. Recall the noise model introduced in Section 2, with datums $x_1...x_m$ and degree of noise parameterized by μ_i. In the agnostic setting, the function sign(μ_i) is not in the hypothesis class $\mathcal{H}$, which makes the problem quite difficult. Our goal is to cast the task as a bandit problem and leverage the rich literature on bandits. We start by defining a set Π:

$$\pi \in \Pi,\ \pi \subseteq [m] : \forall h \in \mathcal{H}, \exists \pi \in \Pi : h(x_i) = 1 \iff i \in \pi\ \forall i \in [m]$$

In other words, for every h there exists π_h which includes the indices i for which $h(x_i) = 1$. In order to frame the problem as a bandit one, we will try to move away from h and start using π.

Recall the risk of a classifier h:

$$R(h) = P_{\substack{X\sim P_x \\ Y\sim P(Y|X)}}(h(X) \neq Y)$$

If P_x is uniform, which we will assume for the rest of the analysis, then we can write $R(h)$ as

$$R(h) = \frac{1}{m}\sum_{i=1}^{m} P(h(x_i) \neq Y_i) \tag{7}$$

where Y_i follows the noise model outlined above.

Let's first analyze when $P(h(x_i) \neq Y_i)$:

$$\begin{aligned}
P(h(x_i) \neq Y_i) &= E[\mathbf{1}\{Y_i = -1, h_i(x) = 1\}, \mathbf{1}\{Y_i = 1, h_i(x) = -1\}] \\
&= E[\mathbf{1}\{Y_i = -1, i \in \pi_h\}, \mathbf{1}\{Y_i = 1, i \notin \pi_h\}] \\
&= E[\mathbf{1}\{Y_i = -1\}\mathbf{1}\{i \in \pi_h\}, \mathbf{1}\{Y_i = 1\}\mathbf{1}\{i \notin \pi_h\}] \\
&= \frac{1-\mu_i}{2}\mathbf{1}\{i \in \pi_h\} + \frac{1+\mu_i}{2}\mathbf{1}\{i \notin \pi_h\} \\
&= \frac{1-\mu_i}{2}\mathbf{1}\{i \in \pi_h\} + \frac{1+\mu_i}{2}(1 - \mathbf{1}\{i \in \pi_h\}) \\
&= \frac{1+\mu_i}{2} + \mathbf{1}\{i \in \pi_h\}(-\mu_i)
\end{aligned}$$

Using the above equation we can write (7) as:

$$\begin{aligned}
R(h) &= \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1+\mu_i}{2} + \mathbf{1}\{i \in \pi_h\}(-\mu_i)\right) \\
&= \frac{1}{m}\sum_{i=1}^{m}(\frac{1+\mu_i}{2}) + \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}\{i \in \pi_h\}(-\mu_i)
\end{aligned}$$

Thus, we can conclude the following:

$$\arg\min_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} P\left(h(x_i)\neq Y_i\right) = \arg\max_{\pi\in\Pi}\sum_{i\in\pi}\mu_i = \pi_* \tag{8}$$

The formulation in (8) converts the active learning problem into a bandits problem. For the i^{th} arm, we observe some random variable Y_i with expectation μ_i and we are trying to find a set π that has the highest mean value. In other words, the active learning algorithm for binary classification that rules out hypotheses has now became a bandits algorithm, which rules out arms.

For clarity consider Figure 5, where $\mu_i = \mu$ and we are trying to find the set π that best matches μ; therefore we get $\arg\max_{\pi\in\Pi}\sum_{i\in\pi}\mu_i = \pi_* = \pi$

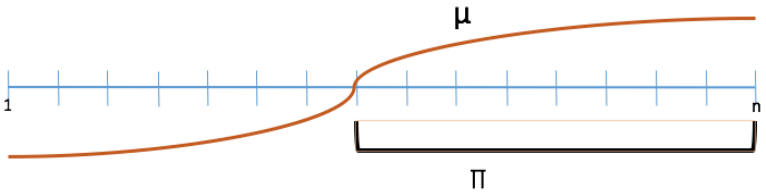


Figure 5: Set π that best matches μ.

Note that we do not assume that μ_i is matched to our policy class and therefore μ_i could be arbitrarily bad, up to the point where we can no longer do binary search. The natural next question then becomes how to solve the problem for arbitrary policy classes. We will analyze an algorithm similar to successive elimination. To obtain a confidence bound for a successive elimination approach at a time step t, let $C(\pi,\pi',t,\delta)$ satisfy

$$P\left(\cup_{t=1}^{\infty}\cup_{\pi\in\Pi\setminus\pi'}^{\infty}|\hat{\mu}_{\pi,t}-\hat{\mu}_{\pi',t}|>C(\pi,\pi',t,\delta)\right)\leq\delta$$

where $\hat{\mu}_{\pi,t}=\sum_{i\in\pi}\hat{\mu}_{i,t}$ and $\hat{\mu}_{i,t}=\frac{1}{t}\sum_{i=1}^{t}Y_{i,j}$ (i.e., $\hat{\mu}_{i,t}$ is the empirical mean of the i^{th} arm if pulled t times), and $\mu_\pi=\sum_{i\in\pi}\mu_i$. We have two sets π and π', which agree on their overlaps. The total disagreement between the two is the value of the arms falling in the symmetric set difference between them:

$$\xi=\hat{\mu}_{\pi,t}-\hat{\mu}_{\pi',t}=\sum_{i\in\pi\setminus\pi'}\hat{\mu}_{i,t}-\sum_{i\in\pi'\setminus\pi}\hat{\mu}_{i,t}$$

Recall that the goal is to identify a set with a classifier that makes the fewest errors or, equivalently, has the highest empirical mean, and to output that set as the empirical risk minimizer. We want to build a confidence interval on $\hat{\mu}_{\pi,t}-\hat{\mu}_{\pi',t}$. First consider $P(t\xi > t\epsilon)$ for some ϵ, where $t\xi$ is a sum of $t|\pi\Delta\pi'|$ mean-zero random variables bounded in $[-1,1]$, and where Δ indicates the symmetric set difference between π and π'.

We can apply Hoeffding's inequality to $P(t\xi > t\epsilon)$ as we have a sum of independent random variables all in $[-1,1]$:

$$P\left((\hat{\mu}_{\pi,t}-\hat{\mu}_{\pi',t})>(\mu_\pi-\mu_{\pi'})+\epsilon\right)\leq\exp\left(\frac{-t\epsilon^2}{2|\pi\Delta\pi'|}\right)$$

As a result, with probability at least $1-\delta$, we have the following confidence interval:

$$|(\hat{\mu}_{\pi,t}-\hat{\mu}_{\pi',t})-(\mu_\pi-\mu_{\pi'})|\leq\sqrt{\frac{2|\pi\Delta\pi'|\log(2/\delta)}{t}}$$

We can use this confidence interval to implement Algorithm 3 for agnostic active learning. The approach is modeled on successive elimination. In step 3, the first step of the loop, we set S_t equal to all i for which there is some disagreement about how to label x_i. Steps 4 and 5 construct $\hat{\mu}_{i,k}$ as an unbiased estimator of the true μ_i. In particular, we only query points in S_t, and for every arm $\hat{\mu}_{i,t}$ we sum only over samples from Y_i. Finally, in step 6 we eliminate hypotheses π whenever there exists some other $\hat{\pi}$ that is statistically significantly better according to the constructed confidence interval.

Algorithm 3 Agnostic Bandit Classification

1: Input δ, $\pi_1 = \pi$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad S_t = (\cup_{\pi \in \Pi_t} \pi) - (\cap_{\pi \in \Pi_t} \pi)$
4: $\quad$ Draw $I_t \in [m]$ randomly and uniformly, if $I_t \in S_t$ let Y_{I_t} be a query, otherwise set Y_{I_t} arbitrarily.
5: $\quad$ Let $\hat{\mu}_{i,t} = \frac{m}{t} \sum_{\ell=1}^{t} Y_{I_\ell} \mathbf{1}\{I_\ell = i\}$
6: $\quad \Pi_{t+1} = \Pi_t - \{\pi \in \Pi_t : \exists \hat{\pi} \in \Pi_t, \hat{\mu}_{\hat{\pi},h} - \hat{\mu}_{\pi,h} > \tilde{C}(\hat{\pi}, \pi, t, \delta)\}$
7: **end for**

Finally, we would like to introduce an upper bound on the total number of samples needed. We can directly apply results from combinatorial bandits like the work of Cao et al. [12] to obtain this bound. In particular, [12] shows that not only will the optimal π_* never be eliminated, if we construct the confidence bounds as above, every other candidate π will eventually be significantly worse and will be eliminated.

Theorem 1. *(Cao et al. [12]) Let* $D(\pi, \pi^*) = \max(\log(|\mathcal{B}(|\pi \Delta \pi^*|, \pi)|), \log(|\mathcal{B}(|\pi \Delta \pi^*|, \pi^*)|)$*, where* $\mathcal{B}(k, \pi) = \{\pi' \in \Pi : (|\pi \Delta \pi'|) \leq k\}$*. For any data distribution and any* $\delta > 0$*, Algorithm 3 guarantees that* $P(\pi \neq \pi^*) \leq \delta$*. Moreover, it runs in polynomial time, and the total number of samples is at most*

$$64 \sum_{i=1}^{m} H_i^{(1)} \left(2 \log(64 H_i^{(1)}) + \log(m\pi^2/\delta)\right) + H_i^{(2)}$$

where $H_i^{(1)} = \max_{\pi : i \in \pi \Delta \pi^*} \frac{|\pi^* \Delta \pi_i|}{<\mu, \pi^* - \pi_i>^2}$ *and* $H_i^{(2)} = \max_{\pi : i \in \pi \Delta \pi^*} \frac{|\pi^* \Delta \pi_i| D(\pi, \pi^*)}{<\mu, \pi^* - \pi>^2}$.

The full proof is very detailed; we refer the interested reader to the original paper. The fact that we can convert an active binary classification problem into a bandits one, and that the result holds even in the agnostic case, is a proof of the power of the bandits framework; that is, bandits allow us to handle cases in which our hypothesis class poorly matches the data.

References

[1] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. *Advances in neural information processing systems.*, pages 337–344, 2005.

[2] Robert Nowak. Generalized binary search. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 568–574. IEEE, 2008.

[3] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010.

[4] Ofer Dekel and Andrew Guillory. Lecture notes in cse522 winter 2011, learning theory, February 2011.

[5] Tibor Hegeds. Generalized teaching dimensions and the query complexity of learning. *In Proceedings of the eighth annual conference on Computational learning theory. ACM*, pages 108–117, 1995.

[6] Steve Hanneke. Teaching dimension and the complexity of active learning. *International Conference on Computational Learning Theory. Springer, Berlin, Heidelberg*, pages 66–81, 2007.

[7] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[8] Venkatesan T Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 53–62. ACM, 2007.

[9] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. volume 42, pages 427–486, 2011.

[10] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in neural information processing systems*, pages 235–242, 2006.

[11] Gowtham Bellala, Suresh K Bhavnani, and Clayton Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Transactions on Information Theory*, 58(1):459–478, 2012.

[12] Tongyi Cao and Akshay Krishnamurthy. Disagreement-based combinatorial pure exploration: Efficient algorithms and an analysis with localization. *arXiv preprint arXiv:1711.08018*, 2017.