Journal of Machine Learning Research 22 (2021) 1-30 Submitted 5/19; Revised 10/20; Published 1/21

Simple and Fast Algorithms for Interactive Machine Learning with Random Counter-examples

Jagdeep Singh Bhatia JAGDEEP@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA 02139

Editor: Manfred Warmuth

Abstract

This work describes simple and efficient algorithms for interactively learning non-binary concepts in the learning from random counter-examples (LRC) model. Here, learning takes place from random counter-examples that the learner receives in response to their *proper equivalence queries*, and the learning time is the number of counter-examples needed by the learner to identify the target concept. Such learning is particularly suited for online ranking, classification, clustering, etc., where machine learning models must be used before they are fully trained.

We provide two simple LRC algorithms, deterministic and randomized, for exactly learning concepts from any concept class H. We show that both these algorithms have an $\mathcal{O}(\log |H|)$ asymptotically optimal average learning time. This solves an open problem on the existence of an efficient LRC randomized algorithm while also simplifying previous results and improving their computational efficiency. We also show that the expected learning time of any Arbitrary LRC algorithm can be upper bounded by $\mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$, where ϵ and δ are the allowed learning error and failure probability respectively. This shows that LRC interactive learning is at least as efficient as non-interactive Probably Approximately Correct (PAC) learning. Our simulations also show that these algorithms outperform their theoretical bounds.

Keywords: interactive learning, active learning, online learning, PAC learning, algorithms, learning theory

1. Introduction

Machine learning has made great advances in fields such as image recognition and natural language processing. However, it currently requires large sets of training data upfront. This is a problem because often the amount of data available is small or sometimes machine learning models must be used before they are fully trained. In these cases, interactive learning, which involves training machine learning models through interaction between a learner and a teacher can be very beneficial. For example, in personalized learning with one-on-one interaction, teachers are better able to assess their students' strength and weakness and provide individualized instruction. Companies such as Netflix and Amazon also use these kinds of interactive algorithms to learn the preferences of their customers and provide them with engaging content.

There are numerous types of interactive learning frameworks, such as that of active learning (Settles, 2012). Our work, however, uses the recently proposed framework of exact

learning through random counter-examples (LRC) presented by Angluin and Dohrn (Angluin and Dohrn, 2017). In this environment, the learning process may be viewed as a game between a *teacher* and a *learner*, where the goal of the learner is to identify a *target concept* chosen by the teacher from a set of hypotheses H called a *concept class*. The learner learns through *proper equivalence queries* (Angluin and Dohrn, 2017), which means that in every round of the learning process, the learner queries the teacher by selecting a hypothesis from their concept class. The teacher either indicates that the learner has identified the target concept or randomly reveals one of the mistakes the learner's hypothesis makes, called a *counter-example*. This process continues until the learner correctly identifies the target concept. The learner's goal is to minimize the number of rounds needed to do this.

In interactive learning, it is known that a learner can always identify the target concept in $\mathcal{O}(\log |H|)$ rounds even if counter-examples are adversarial (Littlestone, 1988). However, this is only true if the learning is improper, meaning that the learner's queries may involve hypotheses from outside the concept class. By contrast, efficient proper learning with adversarial counter-examples is not possible (i.e. it may take up to $|H|$ rounds) and is the reason random counter-examples are crucial. In the Discussion Section (Section 6) we show that for some concept classes, an adversarial teacher can choose counter examples in a way that requires a proper learner query almost every single hypothesis before correctly identifying the target concept.

While other interactive learning models assume adversarial teachers and do worst-case analysis (Angluin, 1988; Barzdin and Freivald, 1972; Emamjomeh-Zadeh and Kempe, 2017; Littlestone, 1988), the LRC framework proposes a more helpful teacher for which average case analysis can be done. This is more realistic because in most practical applications, the teacher is not trying to hinder the learning process. Additionally, learning in the LRC model is proper, meaning that the learner picks their hypotheses from their concept class. This is also more realistic because in applications such as personalized learning, learners with different concept classes can represent learners with different levels of skill and prior knowledge. With improper learning, however, learners cannot be distinguished in this way.

The main contribution of our work is the improved computational efficiency, simplification, and solution to an open problem from previous research in the LRC model (Angluin and Dohrn, 2017). We also provide a mathematical proof that interactive LRC learning is at least as efficient as non-interactive Probably Approximately Correct (PAC) learning (Kearns and Vazirani, 1994) regardless of the learning algorithm used by the interactive learner. In particular, our results are as follows:

- **Majority learning algorithm:** We provide a simple interactive learning algorithm based on majority vote which learns in the asymptotically optimal $\mathcal{O}(\log |H|)$ expected number of rounds. This algorithm works by simply picking, in each round, a hypothesis in the concept class that has the highest "majority" score among the hypotheses that are consistent with, or do not contradict, the counter-examples seen so far. Thus, this algorithm can be seen as a generalization of the well known halving algorithm based on majority voting (Angluin, 1988; Barzdin and Freivald, 1972; Littlestone, 1988) to the LRC setting. The Majority algorithm requires lower computation time and is a significant simplification over the previously known Max-Min algorithm (Angluin and Dohrn, 2017) for learning binary hypotheses, which was also shown to be asymptotically optimal in the LRC setting, but is much more complex.

- **Randomized learning algorithm:** We solve the open problem posed by Angluin and Dohrn (Angluin and Dohrn, 2017) that asks if a randomized learning algorithm exists with the same asymptotically optimal $\mathcal{O}(\log |H|)$ bound on expected number of rounds, specifically when the teacher draws the target concept from a known probability distribution over all hypothesis in H at the beginning of the learning process. We show that such an algorithm does exist, and that the bound is achieved when the learner also draws consistent hypotheses from the same probability distribution conditioned on the sequence of previous learner's hypotheses and teacher's counter-examples. This algorithm also requires lower computation time than the Max-Min algorithm.

- **Upper bound on the learning time of an Arbitrary learning algorithm:** We prove that with probability greater than $1-\delta$ and with error less than ϵ, the expected number of rounds for an Arbitrary LRC algorithm is upper bounded by $\mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$. This result holds for any arbitrary small ϵ and δ values. It also establishes that LRC learning, regardless of the learning algorithm, is at least as efficient as non-interactive Probably Approximately Correct (PAC) learning (Kearns and Vazirani, 1994).

- **Generalization to non-binary hypotheses:** Our algorithms not only apply to binary concept classes, but extend more generally to concept classes with arbitrary values.

- **Performance evaluation:** We show with simulations that, in practice, the Majority and Arbitrary learning algorithms outperform their worst case bounds for one class of hypothesis spaces.

Our work mainly focuses on the number of rounds it takes for the learner to identify the target concept. However, another important consideration is the learner's computational complexity, or the time it takes for the learner to compute the next hypothesis to query. In the algorithms that we present, although the computational complexity is typically linear in $|H|$, it can still be exponential in the number of examples. As H is part of the input, this is inevitable. However, in some scenarios better computational complexity may be possible when H is, for instance, represented implicitly.

2. Related Work

Interactive learning is typically characterized by learning that takes place through query and response. One of the most common models of interactive learning is active learning (Settles, 2012), where the learner queries the teacher for the labels of sample points. Active learning is commonly used in settings where labeling costs are high and therefore it is more cost-effective to selectively label the samples as opposed to labeling them all upfront.

Our work, however, pertains to the interactive learning setting of learning with random counter-examples (Angluin and Dohrn, 2017) using equivalence queries. In this setting, the learner is not allowed to query the teacher directly for sample points. Instead, the teacher randomly selects which sample points to label (i.e. gives random counter-examples) based on where the learner's hypothesis is incorrect. Interactive learning involving more general

membership, equivalence, and related queries was initially proposed in the seminal work of Angluin (Angluin, 1988).

In a related setting, Littlestone (Littlestone, 1988) developed efficient learning algorithms for minimizing the number of mistakes when learning certain boolean functions including k-DNF. Additionally, there has been significant work on interactive learning with equivalence queries involving specific geometric classes such as hyperplanes or axis-aligned boxes (Maass and Turán, 1994). The work of Maass and Turán (Maass and Turán, 1990, 1992) is on the complexity of interactive learning including lower bounds on the number of membership and equivalence queries required for exact learning. Many complexity results can be found in the work of Angluin (Angluin, 2004).

Recent work in interactive learning with equivalence queries pertains to clustering, ranking, and classification (Emamjomeh-Zadeh and Kempe, 2017). For instance, one of the goals is to quickly learn the preferred ranking of a list of items in an online setting (Joachims, 2002; Emamjomeh-Zadeh and Kempe, 2017). This is motivated by applications in personalized web search and information retrieval systems where the learning algorithms learn their users' preferences through online feedback in the form of click behavior. Another goal, involving interactive learning for clustering, is to learn a user's preferred clustering of a set of objects in an online fashion (Awasthi et al., 2017; Balcan and Blum, 2008; Emamjomeh-Zadeh and Kempe, 2017). An application of this work includes identifying communities in social networks.

Learning in the LRC framework (Angluin and Dohrn, 2017), on which our work is based, is proper since the learner's equivalence queries are required to be from their concept class H. This makes LRC different from other recent work (Emamjomeh-Zadeh and Kempe, 2017) because in other settings the learner's hypotheses are not necessarily drawn from the concept class H. This is also what makes our work different from the previous work on halving algorithms based on majority vote (Littlestone, 1988; Barzdin and Freivald, 1972; Angluin, 1988). The LRC model (Angluin and Dohrn, 2017) that we use is also unique in that the teacher's counter-examples are not given arbitrarily, but are randomly drawn from a fixed probability distribution conditioned on the set of all possible counter-examples.

3. Learning Models

The primary learning model used in this work is that of learning through random counter-examples (LRC) (Angluin and Dohrn, 2017). As LRC is only defined for exact learning, we propose a new learning model called *PAC-LRC* for approximate learning in the LRC setting.

In LRC, the learner's concept class can be viewed as a $n \times m$ matrix H, whose rows (denoted by h) represent the set of learner's *hypotheses* and whose columns (denoted by X) represent the set of examples (samples). The *target concept* is one of the rows of the matrix, $h^* \in H$. Additionally, the entries of matrix H represent the values assigned by each hypothesis $h \in H$ to each example $x \in X$ (denoted by function $h(x)$). In this framework, the learner learns through *proper equivalence queries* (Angluin and Dohrn, 2017). This means that in every round of the learning process, the learner queries the teacher by selecting a hypothesis $h \in H$. The teacher either indicates that h is the target concept (and hence the learning is complete), or reveals a *counter-example* on x for $x \in X$ and the value of

$h^*(x)$ on which the target concept differs with h, i.e. $h(x) \neq h^*(x)$. Moreover, there is a known probability distribution $\mathbb{P}$ over X. The teacher's counter-example x is drawn from the probability distribution $\mathbb{P}(h, h^*)$, which is defined as $\mathbb{P}$ conditioned on the event $h(x) \neq h^*(x)$. Upon receiving the counter-example, the learner selects another hypothesis $h \in H$ for the next round until $h = h^*$. The learner's goal is to minimize the number of rounds needed for learning the target concept h^*. In this work, the LRC learning model is used for both the Majority and Randomized learning algorithms. However, for the Randomized learning algorithm it is additionally assumed that the *target concept* is randomly drawn by the teacher at the beginning of the learning process using a known probability distribution.

We also define the *PAC-LRC* model, an extension of LRC, for approximate learning with random counter-examples. For this we introduce two additional parameters: ϵ, the allowed error, and δ, the allowed failure probability. In PAC-LRC, the goal is to approximately learn the target concept with probability at least $1 - \delta$ and with error no more than ϵ. We call hypotheses that differ with the target concept in a region with total probability at least ϵ, ϵ-*bad* hypotheses. As in LRC, learning in PAC-LRC proceeds in rounds. In each round that the learner presents a ϵ-*bad* hypothesis, a randomly drawn counter-example is returned to the learner and the learning continues. Otherwise, the learner's hypothesis is accepted and the learning ends. Learning in PAC-LRC model may also end when all ϵ-*bad* hypotheses in concept class H have been eliminated with probability at least $1 - \delta$. The PAC-LRC model is inspired by the well known non-interactive Probably Approximately Correct or PAC learning model (Kearns and Vazirani, 1994), where generally $\delta, \epsilon \ll \frac{1}{2}$. However, unlike in the LRC and PAC-LRC models, in the PAC model the probability distribution on examples is unknown to the learner.

4. Algorithmic Results

In Section 4.1, some definitions used throughout the paper are explained. In Section 4.2, the Majority learning algorithm is presented and shown to be asymptotically optimal in the LRC model. In Section 4.3, the Randomized learning algorithm is presented and is also shown to be asymptotically optimal in the LRC model. In Section 4.4, an upper bound is derived for the learning time of an Arbitrary learning algorithm in the PAC-LRC setting.

4.1 Preliminaries

Definition 1 *The learner's concept class H is a $n \times m$ matrix, for positive integers n and m, with no duplicated rows or columns. The entries of H are non-negative integers. H can also be thought of as a set of hypotheses where $|H|$ denotes the number of hypotheses in H.*

Definition 2 *X denotes the set of columns of matrix H.*

Definition 3 *$h \in H$ denotes a hypothesis or row in matrix H. For $x \in X$, the function $h(x)$ denotes the value of row h at column x in H.*

Definition 4 *$\mathbb{P}$ denotes a probability distribution over X. For $x \in X$, $\mathbb{P}[x]$ denotes the probability that x is drawn from $\mathbb{P}$, or $x \sim \mathbb{P}$, and $\mathbb{P}[S] = \sum_{x \in S} \mathbb{P}[x]$. $\mathbb{P}[x] > 0$ for all $x \in X$.*

Definition 5 *Define $D(h_1, h_2)$ to be the set of all columns on which $h_1 \in H$ and $h_2 \in H$ have different values. More formally, $D(h_1, h_2) = \{x \in X \mid h_1(x) \neq h_2(x)\}$.*

Definition 6 *For $h_1 \neq h_2$, $\mathbb{P}(h_1, h_2)$ is defined as a probability distribution over $D(h_1, h_2)$, and is the result of conditioning $\mathbb{P}$ on the event $h_1(x) \neq h_2(x)$. $\mathbb{P}[x \mid h_1(x) \neq h_2(x)]$ is defined as the individual probability of drawing element $x \in D(h_1, h_2)$ from distribution $\mathbb{P}(h_1, h_2)$. For $x \notin D(h_1, h_2)$, $\mathbb{P}[x \mid h_1(x) \neq h_2(x)] = 0$. Let $\mathbb{P}[S \mid h_1(x) \neq h_2(x)] = \sum_{x \in S} \mathbb{P}[x \mid h_1(x) \neq h_2(x)]$.*

4.2 Majority Learning Algorithm

In this section, the Majority learning algorithm is proposed and mathematically analyzed. It is shown that the Majority learning algorithm has an expected learning time of $\mathcal{O}(\log |H|)$ and is asymptotically optimal in the LRC learning model.

Definition 7 MAJ_H *is a hypothesis constructed by setting the value in each of its columns to the most frequent element in the corresponding column of matrix H. Ties are broken in favor of the smaller element. Note that it is possible that* $\text{MAJ}_H \notin H$.

Definition 8 *The best majority hypothesis $\hat{h}$ of H is any hypothesis in H that maximizes $\mathbb{P}[h(x) = \text{MAJ}_H(x)]$. Ties are broken in favor of the smallest h in a lexicographic sort by the values of $h(x)$. We consider $h_1 \in H$ lexicographically smaller than $h_2 \in H$, if there exists a column i such that $h_1(x_i) < h_2(x_i)$ and $h_1(x_j) = h_2(x_j)$ for all columns $j < i$.*

while *true* **do**
  Pick $\hat{h}$ to be a best majority hypothesis in H.
  Let x be the counter-example returned by the teacher for $\hat{h}$.
  if *there is no such counter-example* **then**
    Output $\hat{h}$.
  end
  else
    Eliminate the set of hypotheses $\{h \in H \mid h(x) \neq h^*(x)\}$ from H.
  end
end

Algorithm 1: Majority Learning Algorithm

The analysis for the Majority learning algorithm (Algorithm 1) is as follows. Lemma 9 and Lemma 10 establish the performance of the algorithm for any particular round. In these lemmas, H denotes the set of consistent hypotheses (that do not contradict the counter-examples seen so far) and $\hat{h}$ denotes the learner's choice of hypothesis for the round being considered. Lemma 9 shows that the probability that the teacher's counter-example x is a majority element in $\hat{h}$, or $\hat{h}(x) = \text{MAJ}_H(x)$, is at least $\frac{1}{2}$. Lemma 10 shows that a counter-example drawn by the teacher will eliminate at least $\frac{1}{4}$ of the remaining hypotheses in expectation. Through an example we show that $\frac{1}{4}$ is the best possible per round fraction that can be guaranteed for the Majority algorithm in general. Theorem 11 shows that the

Majority learning algorithm has an $\mathcal{O}(\log |H|)$ expected learning time which is asymptotically the best possible in the LRC model (Angluin and Dohrn, 2017). Finally, Theorem 14 bounds the algorithm's per round computation time.

Lemma 9 *Let $\hat{h}$ be the best majority hypothesis selected by the Majority algorithm, the teacher's hypothesis $h^* \neq \hat{h}$ be any hypothesis in H, and A be the event the teacher returns counter example x from the set $\{x : \hat{h}(x) \neq h^*(x) \wedge x \in X\}$. Then, $\mathbb{P}[h^*(x) \neq \text{MAJ}_H(x) \mid A] \geq \frac{1}{2}$.*

Proof Assume for sake of contradiction that the lemma is not true. Then,

$$\mathbb{P}[h^*(x) \neq \text{MAJ}_H(x) \mid A] < \frac{1}{2} \text{ which implies } \mathbb{P}[h^*(x) = \text{MAJ}_H(x) \mid A] > \frac{1}{2}.$$

By definition of A we have $\hat{h}(x) \neq h^*(x)$ for all $x \in A$. Thus given event A, for any x for which $h^*(x) = \text{MAJ}_H(x)$, we must have $\hat{h}(x) \neq \text{MAJ}_H(x)$. Thus,

$$\mathbb{P}[\hat{h}(x) \neq \text{MAJ}_H(x) \mid A] \geq \mathbb{P}[h^*(x) = \text{MAJ}_H(x) \mid A] > \frac{1}{2}.$$

However,

$$\mathbb{P}[\hat{h}(x) \neq \text{MAJ}_H(x) \mid A] > \frac{1}{2} \text{ implies } \mathbb{P}[\hat{h}(x) = \text{MAJ}_H(x) \mid A] < \frac{1}{2}.$$

From $\mathbb{P}[h^*(x) = \text{MAJ}_H(x) \mid A] > \frac{1}{2}$ and $\mathbb{P}[\hat{h}(x) = \text{MAJ}_H(x) \mid A] < \frac{1}{2}$, it follows that

$$\mathbb{P}[h^*(x) = \text{MAJ}_H(x) \mid A] > \mathbb{P}[\hat{h}(x) = \text{MAJ}_H(x) \mid A].$$

Since $\hat{h}(x) = h^*(x)$ for any $x \in X$ not in A, we have

$$\mathbb{P}[h^*(x) = \text{MAJ}_H(x)] > \mathbb{P}[\hat{h}(x) = \text{MAJ}_H(x)].$$

Since $h^* \in H$, it follows that

$$\max_{h \in H} \mathbb{P}[h(x) = \text{MAJ}_H(x)] > \mathbb{P}[\hat{h}(x) = \text{MAJ}_H(x)].$$

This contradicts the definition of $\hat{h}$ (Definition 8). ■

Lemma 10 *Fix any hypothesis $h^* \in H$, and let $\hat{h}$ be a best majority hypothesis selected by the Majority algorithm. For counter-example $x \sim \mathbb{P}(\hat{h}, h^*)$, the expected number of hypotheses $h \in H$ with $h(x) \neq h^*(x)$ is at least $|H|/4$. Thus, in expectation at least $|H|/4$ hypotheses are eliminated by the counter-example.*

Proof Consider counter-example $x \sim \mathbb{P}(\hat{h}, h^*)$ chosen in response to $\hat{h}$. By definition of MAJ_H (Definition 7) it follows that

$$|\{h \mid h(x) = \text{MAJ}_H(x)\}| \geq |\{h \mid h(x) = h^*(x)\}|.$$

Consider a counter-example for which $h^*(x) \neq \text{MAJ}_H(x)$. For such an x we have that

$$\{h \mid h(x) = \text{MAJ}_H(x)\} \subseteq \{h \mid h(x) \neq h^*(x)\}$$

and therefore

$$|\{h \mid h(x) \neq h^*(x)\}| \geq |\{h \mid h(x) = \text{MAJ}_H(x)\}| \geq |\{h \mid h(x) = h^*(x)\}|.$$

Since, $H = \{h \mid h(x) \neq h^*(x)\} \bigcup \{h \mid h(x) = h^*(x)\}$ it follows that

$$|\{h \mid h(x) \neq h^*(x)\}| \geq \frac{|H|}{2}.$$

Note that $\{h \mid h(x) \neq h^*(x)\}$ are exactly the hypotheses in H that get eliminated by counter-example x. We conclude that a counter-example x for which $h^*(x) \neq \text{MAJ}_H(x)$ eliminates $|H|/2$ of the hypotheses in H.

Let A be the event the teacher returns counter example x from the set $\{x : \hat{h}(x) \neq h^*(x) \wedge x \in X\}$. By Lemma 9, $\mathbb{P}[h^*(x) \neq \text{MAJ}_H(x) \mid A] \geq \frac{1}{2}$. Thus, with probability at least $1/2$, the counter-example chosen in response to $\hat{h}$ satisfies $h^*(x) \neq \text{MAJ}_H(x)$. It follows that, in expectation, at least $1/2 \cdot |H|/2 = |H|/4$ hypotheses are eliminated by a counter-example chosen in response to $\hat{h}$. ■

Theorem 11 *The Majority learning algorithm (Algorithm 1) only needs to see an expected* $\log_{\frac{4}{3}} |H|$ *or* $\mathcal{O}(\log |H|)$ *counter-examples to learn* h^* *in the general case.*

Proof This proof by induction follows along the line of the proof of Theorem 21 in (Angluin and Dohrn, 2017). Define $T(n)$ for the Majority algorithm to be the worst case expected number of queries required to learn any any concept class H with $|H| = n$ hypotheses. We re-define predicate $P(n) : T(n) \leq \log_{\frac{4}{3}} n$. Note that the base case $P(1) : T(1) \leq 0$ trivially holds since no queries are needed when the learner's concept class has only one hypothesis. Also, $P(2) : T(2) \leq \log_{\frac{4}{3}} 2$ since at most one query is needed when the learner's concept class has two hypotheses and therefore $T(2) \leq 1 < \log_{\frac{4}{3}} 2$. Assume $P(r)$ is true for all positive integers $r \leq n$ and for some $n \geq 2$. Let H be any concept class with $|H| = n+1$ hypotheses. Let R be the number of remaining consistent hypotheses that do not get eliminated by the teacher's counter-example in a round of the Majority algorithm applied to H. Then,

$$T(n+1) \leq 1 + \sum_{r=1}^{n} \left(Pr(R = r) \cdot T(r) \right).$$

Applying the inductive hypothesis,

$$T(n+1) \leq 1 + \sum_{r=1}^{n} \left(Pr(R = r) \cdot \log_{\frac{4}{3}} r \right).$$

Applying Jensen's Inequality,

$$T(n+1) \leq 1 + \log_{\frac{4}{3}} \mathbb{E}[R].$$

Here $\mathbb{E}[R]$ is the expected number of remaining hypotheses in H that are consistent with the teacher's counter-example. By Lemma 10, the counter-example that is chosen eliminates at least $\frac{1}{4}$ of the hypotheses in expectation. Thus, $\mathbb{E}[R] \leq \frac{3}{4}(n+1)$. Thus,

$$T(n+1) \leq 1 + \log_{\frac{4}{3}} \frac{3}{4}(n+1) = \log_{\frac{4}{3}}(n+1),$$

concluding the inductive step. It follows that Majority learning algorithm can learn the target concept in $\log_{\frac{4}{3}} |H| = \mathcal{O}(\log |H|)$ expected rounds. ■

Theorem 11 establishes that the expected learning time of the Majority algorithm is asymptotically optimal. However its expected learning time is $\log_{\frac{4}{3}} 2 = 2.41$ times more than the $\log_2 |H|$ expected learning time of the Max-Min algorithm (Angluin and Dohrn, 2017). In addition there is a gap between the learning time of the Majority algorithm and the $\log_2 |H| - 1$ lower bound on the expected learning time for any LRC algorithm (Angluin and Dohrn, 2017). Whether the analysis in Theorem 11 for the Majority algorithm can be improved remains an interesting open question. One key difficulty for answering this question, as we show with Lemma 13, is that the bound in Lemma 10 is tight.

Theorem 12 *Given a δ such that $0 < \delta < 1$, the Majority learning algorithm will terminate in $\mathcal{O}(\log \frac{|H|}{\delta})$ rounds with probability at least $1 - \delta$.*

Proof We follow the same line of reasoning given in (Angluin and Dohrn, 2017). Let R_i be the number of consistent hypotheses remaining after i rounds of the Majority algorithm. We first show by induction that $\mathbb{E}[R_i] \leq \left(\frac{3}{4}\right)^i \cdot |H|$. By Lemma 10, the counter-example in a round of the Majority algorithm eliminates at least $\frac{1}{4}$ of the hypotheses in expectation. Thus $\mathbb{E}[R_1] \leq \frac{3}{4} \cdot |H|$ and the base case holds. Applying the inductive step for i we get:

$$\mathbb{E}[R_{i+1}] = \mathbb{E}[\mathbb{E}[R_{i+1} \mid R_i]] \leq \frac{3}{4} \cdot \mathbb{E}[R_i] \leq \left(\frac{3}{4}\right)^{i+1} \cdot |H|,$$

thus completing the induction. The inequality $\mathbb{E}[R_{i+1} \mid R_i] \leq \frac{3}{4} \cdot R_i$ used above follows from Lemma 10.

After the i-th round the identity of the target concept is still unknown iff there are two or more remaining consistent hypotheses or $R_i \geq 2$. As R_i is a non-negative random variable, we can apply Markov inequality to bound the probability of this event.

$$Pr(R_i \geq 2) \leq \frac{E[R_i]}{2} \leq \frac{1}{2} \cdot \left(\frac{3}{4}\right)^i \cdot |H|.$$

Thus, for $i \geq \log_{\frac{4}{3}} \frac{|H|}{\delta}$ we have $Pr(R_i \geq 2) < \delta$ or $Pr(R_i \leq 1) \geq 1 - \delta$. Hence with probability $\geq 1 - \delta$, by round $\log_{\frac{4}{3}} \frac{|H|}{\delta}$, there is at most one consistent hypotheses remaining and the Majority algorithm, having identified the target concept, must terminate. ■

Lemma 13 *The bound in Lemma 10 on the fraction of hypotheses eliminated by the teacher's counter-example in one round of the Majority algorithm is tight.*

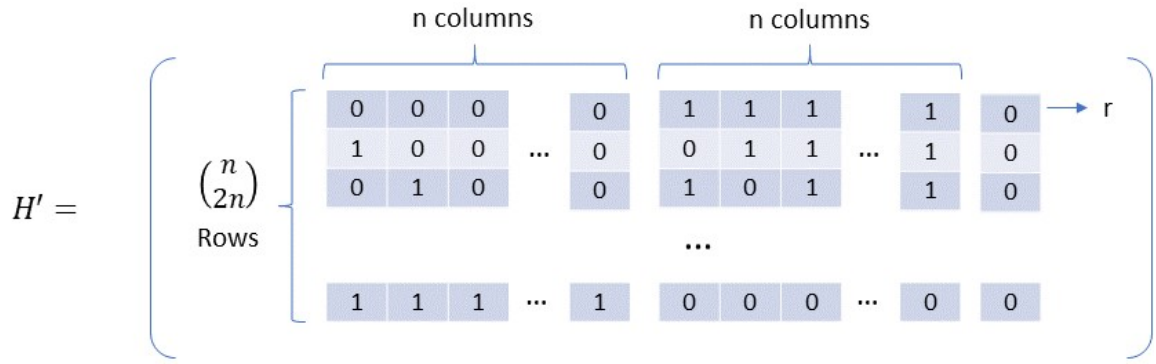

Figure 1: Structure of concept class H'

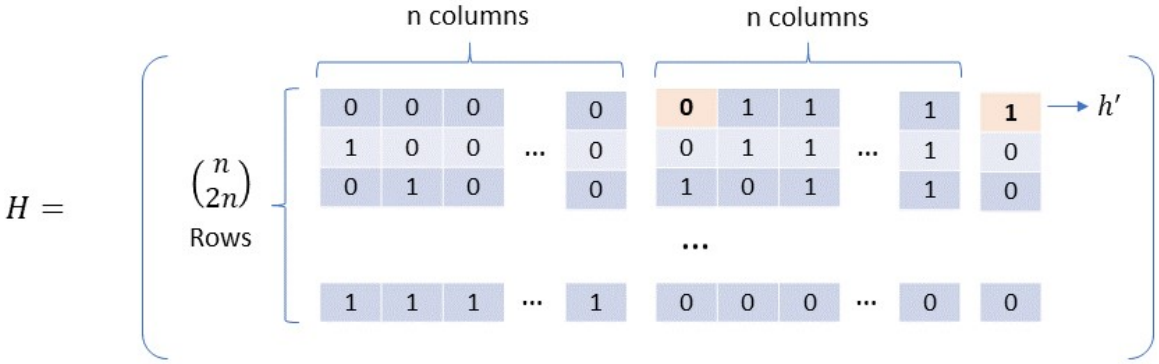

Figure 2: Structure of concept class H

Proof We present an example to show that the per round bound is tight. Consider a binary matrix H' over $2n+1$ columns in which the last column has only zeros while the values in the first $2n$ columns represent all possible combinations of an equal number of zeros and ones. In other word, H' has $\binom{2n}{n}$ distinct rows each consisting of a unique combination of n zeros and n ones in the first $2n$ columns and a zero in the last column. Note that H' also has an equal number of zeros and ones in each of the first $2n$ columns as seen in Figure 1.

Consider the row r of H' that has all zeros in the first n columns and has all ones in the next n columns. We construct a new matrix H from H' by modifying the values in row r of H' as follows: toggle the value in the $n+1$-th column from one to zero and toggle the value in the last column from zero to one. Note that this new matrix H has $n+1$ zeros and n ones in every row as seen in Figure 2.

Let H be the learner's concept class. Let $\mathbb{P}$ be the uniform probability distribution over X, the columns of H. Let h' denote the hypothesis in H that corresponds to the modified row r of H'. Consider the first round of the Majority algorithm. By Definition 7, the majority hypothesis MAJ_H has all zeros. This is because in each column of H either the number of zeros and ones is the same and the tie breaking rule favors zeros over ones, or there are more zeros (i.e. in $n+1$-th column and last column) in which case the column majority is zero. Therefore, all hypotheses $h \in H$ have the same probability $\mathbb{P}[h(x) = \text{MAJ}_H(x)]$ as

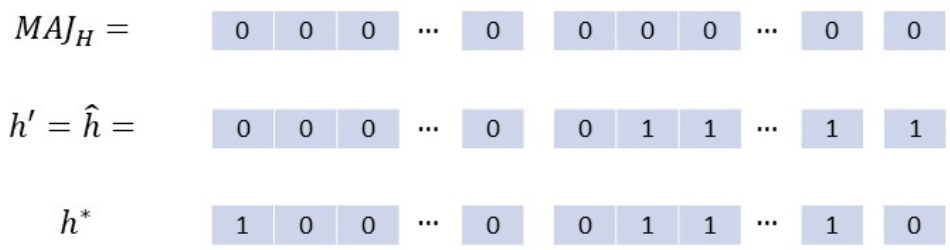

Figure 3: Majority, Learner's, and Teacher's Hypotheses

they all have $n+1$ zeros. For this reason, by Definition 8, the majority hypothesis $\hat{h} \in H$ is h'. This is because ties for the majority hypothesis are broken in favor of the smallest one in a lexicographic sort. h' has all zeros in its first $n+1$ columns while every other hypothesis $h \in H$ has at least one one in the first $n+1$ columns. Therefore, h' is smaller than every other hypotheses in a lexicographic sort. Let the teacher's hypothesis $h^* \in H$ be the hypothesis that differs from $\hat{h} = h'$ in two columns: the first column (where it has a one) and the last column (where it has a zero). Note that such h^* is in H since H was constructed from all possible $\binom{2n}{n}$ combinations of equal number of ones and zeros.

Note that the teacher's counter-example will either be returned on the first column or the last column, each with probability 1/2. If it is the former, half the hypothesis in H will be eliminated, since there is an equal number of ones and zeros in the first column. If it is the latter, only one hypothesis will be eliminated—the learner's hypothesis $\hat{h}$. Thus the expected fraction of hypotheses that get eliminated in the first round by the teacher's counter-example is

$$\frac{1}{|H|}\left(\frac{1}{2}\cdot\frac{|H|}{2}+\frac{1}{2}\cdot 1\right)=\frac{1}{4}+\frac{1}{2|H|},$$

which approaches 1/4 as $|H|$ becomes large. ■

Theorem 14 *Each round of the majority algorithm can be implemented in time $O(|H||X|)$*

Proof By Definition 7, computation of $\mathrm{MAJ}_H(x)$ requires finding maximum values in each column of matrix H, which can be done in time $O(|H|)$ per column. Since there are $O(|X|)$ columns in H, $\mathrm{MAJ}_H(x)$ can be computed in time $O(|H||X|)$. The computation of $\mathbb{P}[h(x) = \mathrm{MAJ}_H(x)]$ for any $h \in H$ takes time $O(|X|)$. Therefore, by Definition 8, the learner's hypothesis $\hat{h}$ can be computed in time $O(|H||X|)$. ■

One of the primary advantages of the Majority algorithm (and also the Randomized algorithm described in the next section) over the Max-Min algorithm is the per round running time for computing the next hypothesis. In the case of the Max-Min algorithm, the computation of the weights of all the edges in the elimination graph (Angluin and Dohrn, 2017) entails a computation time of $O(|H|^2|X|)$. Compared to this, the $O(|H||X|)$ per round computation time requirement of the Majority algorithm is a significant improvement, particularly when $|H|$ is large.

4.3 Randomized Learning Algorithm

In this section, an open problem posed by Angluin and Dohrn (Angluin and Dohrn, 2017) regarding the existence of an efficient randomized algorithm is solved. In this version of the LRC model, the teacher's target concept h^* is drawn from the learner's concept class H according to a known probability distribution $\mathbb{Q}$. A Randomized learning algorithm is presented and mathematically analyzed. It is shown that the Randomized learning algorithm has an expected learning time of $\mathcal{O}(\log |H|)$ and is asymptotically optimal in the LRC learning model.

Definition 15 *$\mathbb{Q}$ denotes a known teacher's probability distribution of drawing the target concept h^* from H.*

This learning algorithm works as follows. In the first round of the Randomized learning algorithm (Algorithm 2), the learner draws a hypothesis randomly from H according to the distribution $\mathbb{Q}$. When presented with a counter-example, the learner updates H by removing the hypotheses that disagree with the counter-example. The learner also updates the teacher's distribution $\mathbb{Q}$ to match the new H. The learner draws the hypothesis for the next round from the updated H according to the updated distribution $\mathbb{Q}$. This process continues until the learner correctly learns the target concept.

Definition 16 *In the Randomized learning algorithm (Algorithm 2), the set of consistent hypotheses evolves over time by the sequence denoted by $H_1, H_2, H_3, \ldots$. Here $H_1 = H$ and $H_1 \supset H_2 \supset H_3 \ldots$. The corresponding evolution of the teacher's probability distribution $\mathbb{Q}$ by the learner over this set is denoted by the sequence $\mathbb{Q}_1, \mathbb{Q}_2, \mathbb{Q}_3, \ldots$, where $\mathbb{Q}_i$ is a distribution on the hypothesis set H_i. Furthermore, the sequence of learner's hypotheses is denoted by $h_1, h_2, h_3, \ldots$ and the corresponding sequence of counter-examples is denoted by $x_1, x_2, x_3, \ldots$.*

Definition 17 *For $i \geq 1$, the pair (h_i, x_i) denotes that for the learner's hypothesis h_i, the counter-example x_i is returned in round i of the Randomized algorithm. For $i \geq 1$, $R_i = \{(h_1, x_1), (h_2, x_2) \ldots (h_i, x_i)\}$ denotes the sequence of these pairs in the first i rounds of the Randomized algorithm. $R_0 = \{\}$ denotes the empty sequence of pairs. For $i \geq 1$, the notation $R_i = R_{i-1} + \{(h_i, x_i)\}$ is used to indicate that in the sequence R_i is made up of the set of pairs in the sequence R_{i-1} followed by the pair (h_i, x_i).*

The probability distributions $\mathbb{Q}_i$ are defined as follows. $\mathbb{Q}_1 = \mathbb{Q}$ is set to the known teacher's distribution over H. For $i \geq 1$, $\mathbb{Q}_{i+1}$ is set to the teacher's posterior distribution over H_{i+1} given R_i. Specifically, for $i \geq 1$, denote q_j^i to be the probability the learner draws $h_j \sim \mathbb{Q}_i$. Then for $h_j \in H_{i+1}$,

$$q_j^{i+1} = Pr(h^* = h_j \mid R_i).$$

We now show how these probabilities can be recursively computed.

Lemma 18 *For $i \geq 1$, the teacher's posterior distribution over H_{i+1} given R_i, or probability distribution $\mathbb{Q}_{i+1}$, can be recursively computed as*

$$q_j^{i+1} = \frac{q_j^i \cdot \mathbb{P}[x_i \mid h_j(x) \neq h_i(x)]}{\sum_{h_k \in H_{i+1}} q_k^i \cdot \mathbb{P}[x_i \mid h_k(x) \neq h_i(x)]}. \tag{1}$$

Proof

Applying Bayes' theorem:

$$q_j^{i+1} = Pr(h^* = h_j \mid R_i) = \frac{Pr(R_i \mid h^* = h_j) \cdot Pr(h^* = h_j)}{Pr(R_i)}. \tag{2}$$

Consider $Pr(R_i \mid h^* = h_j)$ for $i \geq 1$. This can be written as

$$Pr(R_{i-1} + \{(h_i, x_i)\} \mid h^* = h_j) = Pr(R_{i-1} \mid h^* = h_j) \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_j). \tag{3}$$

In other words it is the product of 1) the conditional probability that R_{i-1} is the sequence of pairs in the first $i-1$ rounds of the Randomized algorithm given that teacher's hypothesis is h_j and 2) the conditional probability that for learner's hypothesis h_i in round i the counter-example returned is x_i given teacher's hypothesis h_j and the sequence of pairs R_{i-1} in the first $i-1$ rounds of the Randomized algorithm. Applying Bayes' theorem to $Pr(R_{i-1} \mid h^* = h_j)$ we get

$$Pr(R_{i-1} \mid h^* = h_j) = \frac{Pr(h^* = h_j \mid R_{i-1}) \cdot Pr(R_{i-1})}{Pr(h^* = h_j)} = \frac{q_j^i \cdot Pr(R_{i-1})}{Pr(h^* = h_j)}.$$

The last equality follows from the definition of q_j^i. By substituting in Equation (3) we get

$$Pr(R_i \mid h^* = h_j) = \frac{q_j^i \cdot Pr(R_{i-1}) \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_j)}{Pr(h^* = h_j)}. \tag{4}$$

Substituting in Equation (2) we get:

$$q_j^{i+1} = \frac{q_j^i \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_j) \cdot Pr(R_{i-1})}{Pr(R_i)}. \tag{5}$$

Note that

$$Pr(R_i) = \sum_{h_k \in H_{i+1}} Pr(R_i \wedge h^* = h_k) = \sum_{h_k \in H_{i+1}} Pr(R_i \mid h^* = h_k) \cdot Pr(h^* = h_k).$$

Substituting in Equation (5) and applying Equation (4) we get:

$$q_j^{i+1} = \frac{q_j^i \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_j) \cdot Pr(R_{i-1})}{\sum_{h_k \in H_{i+1}} q_k^i \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_k) \cdot Pr(R_{i-1})}.$$

By simplifying we get

$$q_j^{i+1} = \frac{q_j^i \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_j)}{\sum_{h_k \in H_{i+1}} q_k^i \cdot Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_k)}. \tag{6}$$

Note that $Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_k)$ is independent of R_{i-1} as the probability of getting a counter-example x_i in round i only depends on the learner's hypothesis h_i in round

i and the teacher's hypothesis $h^* = h_k$. In particular these probabilities are distributed as $\mathbb{P}(h_i, h_k)$ (as defined in Definition 6). That is

$$Pr(\{(h_i, x_i)\} \mid R_{i-1} \wedge h^* = h_k) = \mathbb{P}[x_i \mid h_k(x) \neq h_i(x)].$$

Substituting in Equation (6), the result follows. Thus, the teacher's posterior distribution over H_{i+1} given R_i, which is also the probability distribution $\mathbb{Q}_{i+1}$ used by the Randomized algorithm, can be recursively computed from the prior probability distribution $\mathbb{Q}_i$ and the probability distributions $\mathbb{P}(h_i, h_k)$ for all hypotheses $h_k \in H_{i+1}$. ■

$r = 1$, $H_1 = H$, $\mathbb{Q}_1 = \mathbb{Q}$
while *true* **do**
  Draw the learner's hypothesis $h_r \in H_r$ randomly from $\mathbb{Q}_r$.
  Let x_r be the counter-example returned by the teacher.
  if *there is no such counter-example* **then**
    Output h_r.
  end
  else
    $H_{r+1} = H_r - \{h \in H_r \mid h(x_r) \neq h^*(x_r)\}$.
    Calculate $\mathbb{Q}_{r+1}$ as described in Lemma 18.
    $r = r + 1$.
  end
end

Algorithm 2: Randomized Learning Algorithm

The analysis for Algorithm 2 works as follows. A fact is proven in Lemma 22 using which Lemma 23 proves that the expected fraction of hypotheses eliminated by the counter-example given in any round is at least $\frac{1}{2}$. In other words, $\mathbb{E}\left[|H_{i+1}|\right] \leq |H_i|/2$. Using this result, Theorem 24 proves that the expected learning time of Algorithm 2 is at most $\log_2 |H|$. The following analysis is for the i-th round of the algorithm and omits the index i wherever possible.

Definition 19 *For $h \in H_i$, define $V(h, x)$ as the fraction of hypotheses in H_i that disagree with h on example x. More formally,*

$$V(h, x) = \frac{|\{h' \in H_i \mid h'(x) \neq h(x)\}|}{|H_i|}.$$

Lemma 20 *In any i-th round of the algorithm ($i \geq 1$), for any two hypotheses $h_1,\ h_2 \in H_i$ where $h_1(x) \neq h_2(x)$, $V(h_1, x) + V(h_2, x) \geq 1$.*

Proof Note that $V(h_1, x) =$

$$\frac{|\{h' \in H_i \mid h'(x) \neq h_1(x)\}|}{|H_i|} =$$

$$\frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) = h_2(x)\}|}{|H_i|} + \frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) \neq h_2(x)\}|}{|H_i|}.$$

Likewise $V(h_2, x) =$

$$\frac{|\{h' \in H_i \mid h'(x) \neq h_2(x) \wedge h'(x) = h_1(x)\}|}{|H_i|} + \frac{|\{h' \in H_i \mid h'(x) \neq h_2(x) \wedge h'(x) \neq h_1(x)\}|}{|H_i|}.$$

Therefore $V(h_1, x) + V(h_2, x) =$

$$= \frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) = h_2(x)\}|}{|H_i|} + \frac{|\{h' \in H_i \mid h'(x) \neq h_2(x) \wedge h'(x) = h_1(x)\}|}{|H_i|}$$

$$+ \frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) \neq h_2(x)\}|}{|H_i|} + \frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) \neq h_2(x)\}|}{|H_i|}.$$

Note that the numerator of the first three fractions adds up to exactly $|H_i|$. Hence we have

$$V(h_1, x) + V(h_2, x) = \frac{|H_i|}{|H_i|} + \frac{|\{h' \in H_i \mid h'(x) \neq h_1(x) \wedge h'(x) \neq h_2(x)\}|}{|H_i|} \geq 1.$$

■

Definition 21 *Define $E(h_1, h_2)$ to be the expected fraction of hypotheses that are eliminated from H_i when the learner's hypothesis is $h_1 \in H_i$ and the target concept is $h_2 \in H_i$.*

$$E(h_1, h_2) = \sum_{x \in D(h_1, h_2)} V(h_2, x) \cdot \mathbb{P}[x \mid h_1(x) \neq h_2(x)].$$

Lemma 22 *In any i-th round of the algorithm ($i \geq 1$), for any two hypotheses $h_1,\ h_2 \in H_i$ where $h_1 \neq h_2$, $E(h_1, h_2) + E(h_2, h_1) \geq 1$.*

Proof Recall from Definition 5 that $D(h_1, h_2) = \{x \in X \mid h_1(x) \neq h_2(x)\}$. Therefore, $D(h_1, h_2) = D(h_2, h_1)$. We can write

$$\begin{aligned} &E(h_1, h_2) + E(h_2, h_1) \\ &= \sum_{x \in D(h_1, h_2)} V(h_2, x) \cdot \mathbb{P}[x \mid h_1(x) \neq h_2(x)] + \sum_{x \in D(h_2, h_1)} V(h_1, x) \cdot \mathbb{P}[x \mid h_2(x) \neq h_1(x)] \\ &= \sum_{x \in D(h_1, h_2)} (V(h_2, x) \cdot \mathbb{P}[x \mid h_1(x) \neq h_2(x)] + V(h_1, x) \cdot \mathbb{P}[x \mid h_1(x) \neq h_2(x)]). \end{aligned} \tag{7}$$

It follows from Lemma 20 that for any $x \in D(h_1, h_2)$, since $h_1(x) \neq h_2(x)$, $V(h_1, x) \geq 1 - V(h_2, x)$. Substituting in Equation (7) we get

$$\begin{aligned} E(h_1, h_2) + E(h_2, h_1) &\geq \sum_{x \in D(h_1, h_2)} (V(h_2, x) + 1 - V(h_2, x)) \cdot \mathbb{P}[x \mid h_1(x) \neq h_2(x)] \\ &= \sum_{x \in D(h_1, h_2)} \mathbb{P}[x \mid h_1(x) \neq h_2(x)] = 1. \end{aligned}$$

The last equality follows from $D(h_1, h_2) = \{x \in X \mid h_1(x) \neq h_2(x)\}$. ■

Lemma 23 *In any i-th round of the algorithm ($i \geq 1$) the expected fraction of hypotheses eliminated by the counter-example given is at least $\frac{1}{2}$. In other words, $\mathbb{E}\left[|H_{i+1}|\right] \leq |H_i|/2$.*

Proof Note that in round i of the Randomized algorithm the learner draws a hypothesis $h \sim \mathbb{Q}_i$, for $i \geq 1$. Note that $\mathbb{Q}_i$ is also the teacher's posterior distribution over H_i in round i of the Randomized algorithm (Lemma 18). Let $n = |H_i|$, and let q_j denote the probability that $h_j \sim \mathbb{Q}_i$ (we drop the superscript i in q_j^i for ease of exposition). Thus, for hypotheses $h_j, h_k \in H_i$, q_j is the probability of learner drawing hypothesis h_j and q_k is the probability of h_k being the teacher's hypothesis in round i of the Randomized algorithm. Therefore, the expected fraction of hypotheses eliminated by the counter-example in round i of the Randomized algorithm is

$$\sum_{j=1}^{n}\sum_{k=1}^{n} q_j q_k E(h_j, h_k).$$

By Lemma 22, and using the fact that $E(h,h) = 1$ for any $h \in H_i$ the expected fraction of hypotheses eliminated by the counter-example in round i of the Randomized algorithm,

$$\geq \sum_{j=1}^{n} {q_j}^2 + \sum_{j=1}^{n}\sum_{k>j}^{n} q_j q_k = (q_1 + q_2 + q_3 + ... + q_n)^2 - \sum_{j=1}^{n}\sum_{k>j}^{n} q_j q_k$$

$$\geq (q_1 + q_2 + q_3 + ... + q_n)^2 - \frac{(q_1 + q_2 + q_3 + ... + q_n)^2}{2} = (1 - \frac{1}{2}) = \frac{1}{2}.$$

■

Theorem 24 *In the setting where the target concept is drawn from $\mathbb{Q}$ and the counter-examples are drawn from $\mathbb{P}$, Algorithm 2 only needs to see an expected $\log_2 |H|$ counter-examples to learn h^*.*

Proof From Lemma 23 it follows that in any i-th round of the Randomized algorithm ($i \geq 1$), at least $\frac{1}{2}$ of the remaining hypotheses get eliminated in expectation. Thus, by Theorem 21 of Angluin and Dohrn (Angluin and Dohrn, 2017) it follows that the Randomized algorithm (Algorithm 2) can learn the teacher's hypothesis in $\log_2 |H|$ expected rounds. ■

Theorem 25 *Each round of the Randomized learning algorithm can be implemented in time $O(|H||X|)$.*

Proof The most expensive per-round operation in the Randomized learning algorithm is the computation of posterior distribution Q_{i+1}. In round $i+1$, for each of the $|H_{i+1}|$ hypotheses h_j, the quantity q_j^{i+1} needs to be computed from Equation (1) in Lemma 18. The numerator of Equation (1) requires calculating the conditional probability of x_i with respect to $h_j(x) \neq h_i(x)$, which takes time $O(|X|)$. The denominator of Equation (1) is simply a normalization of the of q_j^{i+1} values, which is achieved by dividing them by their sum. Therefore, the overall per-round computation time of the Randomized learning algorithm is $O(|H||X|)$. ■

4.4 Upper Bound on the Learning Time of Arbitrary Learning Algorithm

In this section, an Arbitrary learning algorithm (Algorithm 3) is analyzed in the PAC-LRC model where the learner is allowed to pick any consistent hypothesis in every round. Similar to LRC, in the PAC-LRC model, a randomly drawn counter-example is returned to the learner in each round. However, the two differ in their termination conditions. Recall that in the PAC-LRC model with parameters ϵ and δ, learning ends if all ϵ-*bad* hypotheses (hypothesis that differs with the target concept in a region with total probability at least ϵ) in H have been eliminated with probability at least $1-\delta$. The only exception is if the learner presents a hypothesis that is not ϵ-*bad* in any round, in which case the teacher does not return a counter-example and learning ends immediately.

The main result of this section is that with probability at least $1-\delta$, the Arbitrary learning algorithm terminates within $\mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$ rounds. In other words, the learning time of the Arbitrary learning algorithm is $\mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$ with probability at least $1-\delta$.

$i = 1$.
while *true* **do**
  Pick h_i to be any arbitrary hypothesis in H.
  Let x_i be the counter-example returned by the teacher.
  if *there is no such counter-example* **then**
    Output h_i.
  end
  else
    Eliminate the set of hypotheses $\{h \in H \mid h(x_i) \neq h^*(x_i)\}$ from H.
  end
  $i = i + 1$.
end

Algorithm 3: Arbitrary Learning Algorithm

Definition 26 *Let the target hypothesis be h^*. Let h_i denote the learner's hypotheses in round i. In other words, the sequence of learner's hypotheses is written as $h_1, h_2 \ldots$. Let the sequence of counter-examples received by the learner be denoted by $x_1, x_2 \ldots$.*

Definition 27 *Let the weight of a column $x \in X$ at the start of any round $i \geq 1$ be denoted as $W_i(x)$, and let $W_i(S) = \sum_{x \in S} W_i(x)$. For hypothesis h, $W_i(h) = W_i(D(h, h^*))$ denotes the total weight on all the columns on which h differs from h^* in round i. For all $x \in X$, define $W_1(x) = 0$, and allow the weight of a column to be incremented in each round by the probability that the column is chosen as a counter-example. More formally for $n \geq 2$,*

$$W_n(x) = W_{n-1}(x) + \mathbb{P}[x \mid h_{n-1}(x) \neq h^*(x)].$$

Or

$$W_n(x) = \sum_{i=1}^{n-1} \mathbb{P}[x \mid h_i(x) \neq h^*(x)].$$

Note that $\mathbb{P}[x \mid h_i(x) \neq h^*(x)] = 0$ if $x \notin D(h_i, h^*)$.

Also note that in each round $i \geq 1$, $W_{i+1}(X) = W_i(X) + 1$. This is because

$$\sum_{x \in X} \mathbb{P}[x \mid h_i(x) \neq h^*(x)] = 1$$

Definition 28 *For $i \geq 1$, let $E_i(h)$ be the probability that on counter-example x_i, hypothesis h contradicts h^*. More formally,*

$$E_i(h) = \mathbb{P}[h(x) \neq h^*(x) \mid h_i(x) \neq h^*(x)].$$

Note that for any $h \in H$ and any round $n \geq 1$, by Definition 27 and Definition 28,

$$\sum_{i=1}^{n} E_i(h) = \sum_{i=1}^{n} \quad \mathbb{P}[h(x) \neq h^*(x) \mid h_i(x) \neq h^*(x)]$$

$$= \sum_{x \in D(h,h^*)} \quad \sum_{i=1}^{n} \mathbb{P}[x \mid h_i(x) \neq h^*(x)] = W_{n+1}(D(h, h^*)) = W_{n+1}(h).$$

Lemma 29 *Let $\theta = \ln(\frac{|H|}{\delta})$. At the beginning of any round $n \geq 1$, consider a hypothesis $h \in H$ for which $W_n(h) > \theta$. The probability that h has not already been eliminated, in other words it does not contradict any of the counter-examples $x_1, x_2, \ldots, x_{n-1}$, is at most $\frac{\delta}{|H|}$.*

Proof Note that for $n = 1$ the lemma holds trivially because $W_1(h) = 0$ for all $h \in H$. Therefore, in the following we assume $n > 1$.

In each i-th round of the algorithm $i \geq 1$, the counter-example x_i is drawn with the probability distribution $\mathbb{P}[x \mid h_i(x) \neq h^*(x)]$. The probability that on this counter-example hypothesis h is inconsistent with h^* is therefore $\mathbb{P}[h(x) \neq h^*(x) \mid h_i(x) \neq h^*(x)]$, which by Definition 28 equals $E_i(h)$. Therefore, with probability $1 - E_i(h)$, hypothesis h is not eliminated in round i. Thus, the probability that hypothesis $h \in H$ with $W_n(h) > \theta$ has not been eliminated in the first $n - 1$ rounds is:

$$\prod_{i=1}^{n-1} (1 - E_i(h)) \leq \prod_{i=1}^{n-1} (1 - \frac{\sum_{i=1}^{n-1} E_i(h)}{n-1}) = \prod_{i=1}^{n-1} (1 - \frac{W_n(h)}{n-1}) \leq e^{-W_n(h)} < e^{-\ln(\frac{|H|}{\delta})} = \frac{\delta}{|H|}.$$

■

Definition 30 *Let a threshold $\theta^*(x) = \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}$ be defined over every $x \in X$ based on the probability distribution $\mathbb{P}$. A column is considered light at the start of any round i if $W_i(x) \leq \theta^*(x)$ and heavy otherwise. Let $L_i \subset X$ denote the set of light columns and $B_i \subset X$ denote the set of heavy (or bulky) columns at the start of round i.*

Lemma 31 *Let the learner's hypothesis h_i in round $i \geq 1$ be ϵ-bad. That is, $\mathbb{P}[h_i(x) \neq h^*(x)] \geq \epsilon$. Let h_i also satisfy that at the start of round i its total weight $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$. Then, the weight of light columns L_i should increase by at least half in round i. In other words*

$$W_{i+1}(L_i) \geq W_i(L_i) + \frac{1}{2}.$$

Proof Assume for the sake of contradiction that

$$W_{i+1}(L_i) < W_i(L_i) + \frac{1}{2}.$$

$W_{i+1}(X) = W_i(X) + 1$ and $B_i + L_i = X$ together imply that

$$W_{i+1}(B_i) > W_i(B_i) + \frac{1}{2}. \tag{8}$$

Note that only the weights of the columns in the set $D(h_i, h^*)$ change in this round as the counter-example x_i is drawn from the probability distribution $\mathbb{P}[x \mid h_i(x) \neq h^*(x)]$. Additionally, all heavy columns whose weights change in this round therefore belong to the set $D(h_i, h^*) \cap B_i$. Applying Definition 27 we therefore have,

$$W_{i+1}(B_i) - W_i(B_i) = \sum_{x \in D(h_i, h^*) \cap B_i} \mathbb{P}[x \mid h_i(x) \neq h^*(x)]. \tag{9}$$

From Equations (8) and (9) we get,

$$\sum_{x \in D(h_i, h^*) \cap B_i} \mathbb{P}[x \mid h_i(x) \neq h^*(x)] = \mathbb{P}[D(h_i, h^*) \cap B_i \mid h_i(x) \neq h^*(x)] > \frac{1}{2}. \tag{10}$$

The total weight on h_i at the beginning of round i satisfies

$$W_i(h_i) = W_i(D(h_i, h^*)) \geq W_i(D(h_i, h^*) \cap B_i).$$

Since the weights of all $x \in B_i$, at the beginning of round i, is at least $\theta^*(x) = \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}$ (Definition 30), the total weight on h_i at the beginning of round i satisfies

$$W_i(h_i) \geq W_i(D(h_i, h^*) \cap B_i) > \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \mathbb{P}[D(h_i, h^*) \cap B_i]}{\epsilon}. \tag{11}$$

Since $\mathbb{P}[h_i(x) \neq h^*(x)] \geq \epsilon$, and since $\mathbb{P}[D(h_i, h^*) \cap B_i \mid h_i(x) \neq h^*(x)] > \frac{1}{2}$ (Equation 10) we have

$$\mathbb{P}[D(h_i, h^*) \cap B_i)] > \frac{\epsilon}{2}.$$

Combining with Equation (11) we get, the total weight on h_i at the beginning of round i satisfies

$$W_i(h_i) > \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \mathbb{P}[D(h_i, h^*) \cap B_i]}{\epsilon} > \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \epsilon}{\epsilon \cdot 2} = \ln(\frac{|H|}{\delta}).$$

This is a contradiction because we assumed $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$. ■

Lemma 32 *If for all $i \geq 1$ the learner's hypotheses h_i satisfy $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$, then within $\mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$ rounds either all $x \in X$ become heavy, or the algorithm terminates because the learner presents a hypothesis that is not ϵ-bad.*

Proof Note that it is enough to bound the number of rounds it takes for all $x \in X$ to become heavy under the assumption that in every round the learner's hypothesis is *ϵ-bad*. Thus, in each round i we assume the learner's hypothesis is *ϵ-bad* and has weight at most $\ln(\frac{|H|}{\delta})$. Therefore, by Lemma 31, the total weight of light columns increases by at least $\frac{1}{2}$ in each round $i \geq 1$. We assume this in the proof below.

Additionally, we assume the number of rounds is at least 2, because otherwise the result trivially holds.

Note that in each round $i \geq 1$, since h_i is *ϵ-bad*, $\mathbb{P}[h_i(x) \neq h^*(x)] \geq \epsilon$ and therefore $\mathbb{P}[x \mid h_i(x) \neq h^*(x)] \leq \frac{\mathbb{P}[x]}{\epsilon}$. Thus, we can write

$$W_{i+1}(x) - W_i(x) \leq \frac{\mathbb{P}[x]}{\epsilon}. \tag{12}$$

We now provide some intuition motivating the rest of the proof. We know that each column $x \in X$ has threshold $\theta^*(x) = \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}$. After acquiring $\theta^*(x)$ weight, x transitions from light to heavy. However, in the round j that x transitions from a light to a heavy column, its final weight $W_{j+1}(x)$ exceeds threshold $\theta^*(x)$ by the amount $s(x) = W_{j+1}(x) - \theta^*(x)$. Call this difference $s(x)$ spillover weight. Note that $s(x) \leq W_{j+1}(x) - W_j(x)$. Accounting for spillover weight, the total weight that a column can accumulate while it is light is at most $\theta^*(x) + s(x)$. At the same time, the weight of light columns increases by at least $\frac{1}{2}$ in each round. Therefore, during any arbitrary round i where light columns remain, we can see that the total weight accumulated on the light columns, which is at least $\frac{i}{2}$, cannot exceed the sum of $\theta^*(x) + s(x)$ over all columns $x \in X$. In other words,

$$\frac{i}{2} \leq \sum_{x \in X} [\theta^*(x) + s(x)]$$

By Equation (12), it follows that $s(x) \leq W_{j+1}(x) - W_j(x) \leq \frac{\mathbb{P}[x]}{\epsilon}$. This in conjunction with the definition of $\theta^*(x)$ yields

$$\frac{i}{2} \leq \sum_{x \in X} \left[\ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon} + \frac{\mathbb{P}[x]}{\epsilon}\right]$$

$$\frac{i}{2} \leq \ln(\frac{|H|}{\delta}) \cdot \frac{2}{\epsilon} + \frac{1}{\epsilon}$$

$$i \leq O(\frac{\log\frac{|H|}{\delta}}{\epsilon}).$$

Thus the number of rounds i where light columns remain is bounded by $O(\frac{\log\frac{|H|}{\delta}}{\epsilon})$. A formal argument follows below.

For any particular $x \in X$, and round $i \geq 1$ we define values $f_i(x)$ as follows. Note that all columns start out being light initially (since $W_1(x) = 0$). Let j be the round in which

x turns from light to heavy. If, $i > j$, which happens when $x \in B_i$, then $f_i(x) = j + 1$. Otherwise if $i \leq j$, which happens when $x \in L_i$, then $f_i(x) = i$.

Let $i \geq 1$ be any round in which the set L_i is not empty. We claim that

$$\sum_{x \in X} W_{f_i(x)}(x) \geq \frac{i-1}{2}.$$

We continue by induction on i. The base case, $i = 1$, holds because all columns are initially light. Therefore, for all $x \in X$, $f_1(x) = 1$ and $W_1(x) = W_{f_1(x)}(x) = \frac{1-1}{2} = 0$. Assume (for the sake of induction), the induction hypothesis that

$$\sum_{x \in X} W_{f_k(x)}(x) \geq \frac{k-1}{2}.$$

for some round $k \geq 1$ where the set L_k is not empty. We want to prove that

$$\sum_{x \in X} W_{f_{k+1}(x)}(x) \geq \frac{k}{2}.$$

We know that for all $x \in L_k$, $f_{k+1}(x) = k + 1$ and $f_k(x) = k$. Therefore,

$$\sum_{x \in L_k} [W_{k+1}(x) - W_k(x)] = \sum_{x \in L_k} \left[W_{f_{k+1}(x)}(x) - W_{f_k(x)}(x)\right]. \tag{13}$$

We also know by Lemma 31, that the total weight of light columns increases by at least $\frac{1}{2}$ in round k. Thus,

$$\sum_{x \in L_k} [W_{k+1}(x) - W_k(x)] \geq \frac{1}{2}. \tag{14}$$

Combining Equations (13) and (14) we see that,

$$\sum_{x \in L_k} W_{f_{k+1}(x)}(x) \geq \sum_{x \in L_k} W_{f_k(x)}(x) + \frac{1}{2}.$$

Also, because both $W_k(x)$ and $f_k(x)$ are non-decreasing functions,

$$\sum_{x \in B_k} W_{f_{k+1}(x)}(x) \geq \sum_{x \in B_k} W_{f_k(x)}(x).$$

Adding the last two inequalities we get

$$\sum_{x \in X} W_{f_{k+1}(x)}(x) \geq \sum_{x \in X} W_{f_k(x)}(x) + \frac{1}{2}.$$

Using the induction hypothesis we arrive at

$$\sum_{x \in X} W_{f_{k+1}(x)}(x) \geq \frac{k-1}{2} + \frac{1}{2} \geq \frac{k}{2}.$$

This concludes the proof that

$$\sum_{x \in X} W_{f_i(x)}(x) \geq \frac{i-1}{2}. \tag{15}$$

In any particular round $i \geq 2$, let $x \in B_i$ be a heavy column. By definition of $f_i(x)$, $x \in L_{f_i(x)-1}$. In other words, x must have been a light column in round $f_i(x) - 1$. By definition 30, when a column x is light its weight is at most $\theta^*(x) = \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}$. Thus,

$$W_{f_i(x)-1}(x) \leq \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}.$$

Summing over all columns $x \in B_i$ we get:

$$\sum_{x \in B_i} W_{f_i(x)-1}(x) \leq \sum_{x \in B_i} \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}. \tag{16}$$

From Equation (12), it follows that in any round $i \geq 2$:

$$(W_{f_i(x)}(x) - W_{f_i(x)-1}(x)) \leq \frac{\mathbb{P}[x]}{\epsilon}.$$

Considering all $x \in B_i$, it follows that in any round $i \geq 2$,

$$\sum_{x \in B_i} (W_{f_i(x)}(x) - W_{f_i(x)-1}(x)) \leq \sum_{x \in B_i} \frac{\mathbb{P}[x]}{\epsilon} \leq \sum_{x \in X} \frac{\mathbb{P}[x]}{\epsilon} = \frac{1}{\epsilon}. \tag{17}$$

Therefore,

$$\sum_{x \in B_i} W_{f_i(x)}(x) \leq \sum_{x \in B_i} W_{f_i(x)-1}(x) + \frac{1}{\epsilon}.$$

Combining with Equation (16) it follows:

$$\sum_{x \in B_i} W_{f_i(x)}(x) \leq \sum_{x \in B_i} \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon} + \frac{1}{\epsilon}. \tag{18}$$

In round $i \geq 2$, consider a light column $x \in L_i$. By definition, $f_i(x) = i$. Therefore,

$$W_{f_i(x)}(x) \leq \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon}.$$

Summing over all such light columns and combining with Equation (18) we get

$$\sum_{x \in X} W_{f_i(x)}(x) = \sum_{x \in B_i} W_{f_i(x)}(x) + \sum_{x \in L_i} W_{f_i(x)}(x) \leq \sum_{x \in X} \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon} + \frac{1}{\epsilon}. \tag{19}$$

Thus it follows that

$$\sum_{x \in X} W_{f_i(x)}(x) \leq \sum_{x \in X} \ln(\frac{|H|}{\delta}) \cdot \frac{2\mathbb{P}[x]}{\epsilon} + \frac{1}{\epsilon} \leq \ln(\frac{|H|}{\delta}) \cdot \frac{2}{\epsilon} + \frac{1}{\epsilon}. \tag{20}$$

Combining with Equation (15) we get that

$$\frac{i-1}{2} \leq \sum_{x \in X} W_{f_i(x)}(x) \leq \ln(\frac{|H|}{\delta}) \cdot \frac{2}{\epsilon} + \frac{1}{\epsilon}. \tag{21}$$

It follows that

$$i \leq \frac{4 \cdot \log \frac{|H|}{\delta} + 2}{\epsilon} + 1.$$

Thus, at the end of i rounds if all learner's hypotheses h_i are ϵ-*bad* and satisfy $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$, every column must be heavy. Here

$$i = \frac{4 \cdot \log \frac{|H|}{\delta} + 2}{\epsilon} + 1 = \mathcal{O}(\frac{1}{\epsilon} \log \frac{|H|}{\delta}).$$

This establishes the Lemma. ■

Theorem 33 *With probability at least $1-\delta$ the Arbitrary learning algorithm terminates in $\mathcal{O}(\frac{1}{\epsilon} \log \frac{|H|}{\delta})$ rounds.*

Proof Let

$$N = \frac{4 \cdot \log \frac{|H|}{\delta} + 2}{\epsilon} + 1.$$

We first show that the Arbitrary algorithm cannot run for more than $N+1$ rounds assuming that in every round i except the last, $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$. We then show with probability at least $1-\delta$ that in every round i except the last, $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$, completing the proof.

Assume for the sake of contradiction that the algorithm runs for $m+1 > N+1$ rounds, where in rounds $1 \leq i \leq m$, $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$. Note that in rounds $1 \leq i \leq m$ the learner's hypotheses h_i are all ϵ-*bad*. Otherwise, the algorithm would have terminated before round $m+1$. Consider the $N+1$-th round (because $N+1 \leq m$). By our assumptions, Lemma 32 applies and it follows that all the columns must be heavy both by the end of the N-th round and in the beginning of round $N+1$.

By Definition 27,

$$W_{N+1}(h_{N+1}) = W_{N+1}(D(h_{N+1}, h^*)).$$

Since every column in the beginning of round $N+1$ is heavy, by Definition 30,

$$W_{N+1}(D(h_{N+1}, h^*)) > \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \mathbb{P}[D(h_{N+1}, h^*)]}{\epsilon}.$$

Also, since h_{N+1} must be ϵ-*bad*, we have $\mathbb{P}[D(h_{N+1}, h^*)] \geq \epsilon$. Thus,

$$W_{N+1}(h_{N+1}) > \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \mathbb{P}[D(h_{N+1}, h^*)]}{\epsilon} \geq \ln(\frac{|H|}{\delta}) \cdot \frac{2 \cdot \epsilon}{\epsilon} > \ln(\frac{|H|}{\delta}).$$

This is a contradiction because $W_{N+1}(h_{N+1}) \leq \ln(\frac{|H|}{\delta})$ (since $N+1 \leq m$). Thus, we must have that the number of rounds, $m+1$, cannot exceed $N+1$ and the Arbitrary algorithm must terminate in $m+1 \leq N+1 = \mathcal{O}(\frac{1}{\epsilon}\log\frac{|H|}{\delta})$ rounds.

We now show with probability at least $1-\delta$, our previous assumption holds. That is, in every round $1 \leq i \leq m$, $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$.

Each of the learner's hypotheses $h_1, h_2 \ldots h_m$ has to be ϵ-*bad* and is consistent in the beginning of the i-th round in which it got selected by the algorithm. From Lemma 29 it follows for any of these learner's hypothesis h_i:

$$Pr(\ [W_i(h_i) > \ln(\frac{|H|}{\delta})]\ ) < \frac{\delta}{|H|}.$$

By applying the union bound we get:

$$Pr(\bigvee_{i=1}^{m}\ [W_i(h_i) > \ln(\frac{|H|}{\delta})]\ ) < m \cdot \frac{\delta}{|H|} \leq \delta.$$

Here we used the fact that $m \leq |H|$, as any PAC-LRC learning algorithm must always terminate within $|H|$ rounds. Thus:

$$Pr(\bigwedge_{i=1}^{m}\ [W_i(h_i) \leq \ln(\frac{|H|}{\delta})]\ ) > 1-\delta. \tag{22}$$

Thus with probability at least $1-\delta$, each of the learner's hypotheses h_i satisfy $W_i(h_i) \leq \ln(\frac{|H|}{\delta})$. This concludes the proof. ■

5. Simulation Results

In this section we present the results of testing the performance of our interactive LRC algorithms on randomly generated learner's concept classes H which are chosen to be binary matrices.

We generate H with a particular structure—with g distinct groups of 100 rows each, for some integer g—so that $|H| = 100g$. The groups are generated so that two hypotheses from the same group only differ in a few places. At the same time, hypotheses from different groups differ a lot. The process for generating $|H|$ is described in detail below.

To generate H, we first generate g random rows, one for each group. From each of these g rows we derive 99 additional random rows. H consists of these $99g$ rows plus the original g rows for a total of $100g$ rows. Figure 4 illustrates this structure of the learner's concept class.

The generation of the g rows in the first step is as follows. We construct a binary matrix H' with g rows and the same set of columns X as H. For each column x_j of matrix H', $1 \leq j \leq |X|$, we draw a real number p_j independently and uniformly from the interval $[0,1]$. For each row $h_i \in H'$ and j-th column $x_j \in X$, we independently set $h_i(x_j)$ to 1 with probability p_j and 0 with probability $1-p_j$. Note that in H' the expected number of ones in column j is $p_j \cdot g$.

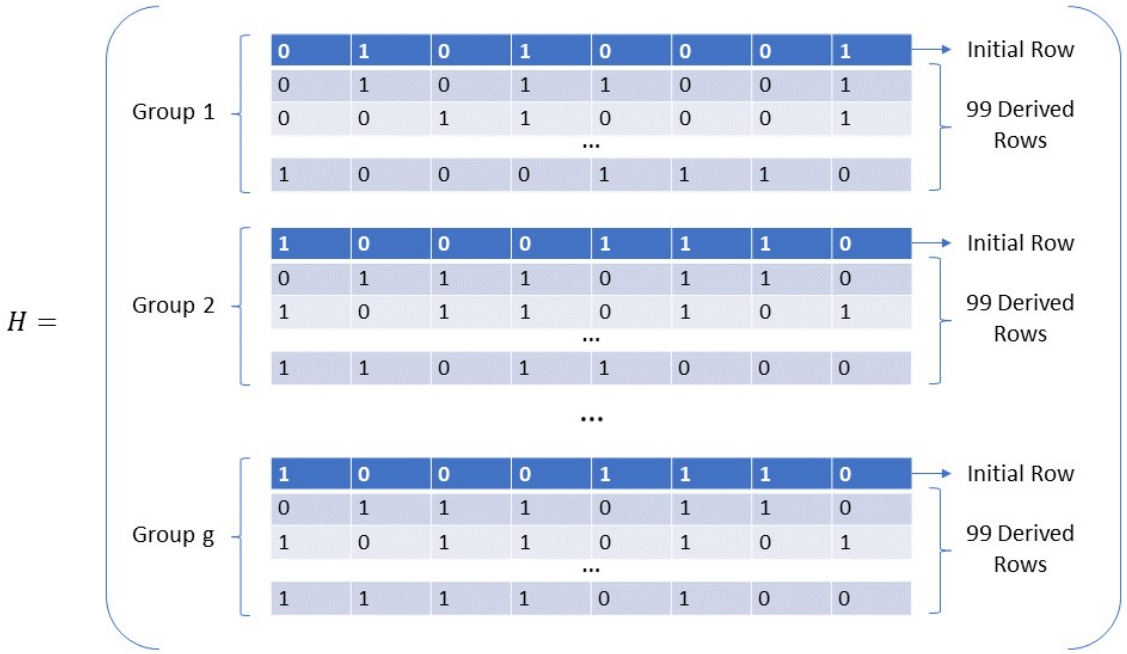

Figure 4: Structure of learner's concept class H

Note that the probability that any two rows of H' differ on column j is $2p_j(1-p_j)$. Since p_j is uniformly distributed in the interval $[0,1]$, the expected value of $2p_j(1-p_j)$ is $1/3$. This implies that any two rows of H' are likely to differ in $|X|/3$ columns. We use the rows of H' as the g rows of H in the first step.

Let $h \in H'$ be the row generated for a group in the first step. In the second step, the additional 99 rows for this group are generated as follows. Let $q_1, q_2, \ldots q_{99}$ be 99 different, equally spaced, values that span the range $[0.01, 0.04]$. The i-th row h_i for the group is generated by toggling each value in row h independently with probability q_i. In other words, $Pr(h(x) \neq h_i(x)) = q_i$ for all $x \in X$. Assuming a uniform distribution over the samples X, in expectation, rows h and h_i differ on $q_i|X|$ columns.

Some additional details regarding our implementation are as follows. The teacher's concept is randomly drawn from H. In each of our tests, the performance of a learning algorithm is computed by taking an average over at least 100 randomly generated binary matrices H of a given size. We also make some simplifications in our implementation. We assume a uniform distribution over the samples X. Thus $\mathbb{P}[x]$ is the same for all columns $x \in X$. Our implementation of the Arbitrary learning algorithm, called the Anti-majority learning algorithm, chooses in each round the hypothesis that has the lowest majority score. Ideally we would want to implement an adversarial learning algorithm that would maximize the expected number of counterexamples needed for learning. However, as such an algorithm can be hard to design, we approximate it by following the opposite of the Majority strategy as that can can be beneficial to the adversary. The Anti-majority learning algorithm terminates when there are no more ϵ-*bad* hypotheses left in H or when the learner proposes a hypothesis that is not ϵ-*bad*. In other words, we set $\delta = 0$ for ease

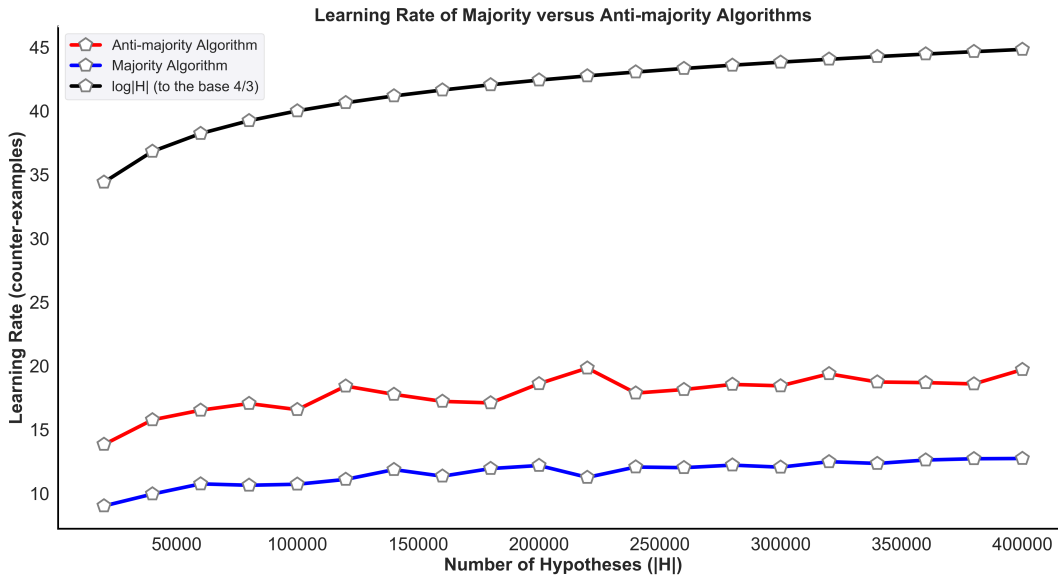

Figure 5: Performance of Majority versus Anti-majority Learning Algorithm

of implementation. It is worth noting however that for higher values of δ, the algorithms' performance would not be any worse as they would have a higher chance of terminating earlier.

5.1 Majority versus Anti-majority Algorithm Performance Analysis

When comparing the performance of the Majority and the Anti-majority LRC algorithms, we generated random binary matrices H with 500 columns using the previously described procedure.

For these matrices the number of rows ($|H|$) ranged from 20000 to 400000, in increments of 20000. We also set $\epsilon = 0.0$, requiring the Anti-majority learning algorithm to exactly learn the target concept in order to compare the algorithms on equal terms. Figure 5 shows the performance of these algorithms.

Here, the X-axis shows the number of hypotheses ($|H|$) in H and the Y-axis shows the expected number of rounds or the number of mistakes made by the learner in learning the target concept. In Figure 5 we also plot the function $\log_{\frac{4}{3}} |H|$, the theoretical upper bound for the Majority algorithm.

As seen in Figure 5, on this data set the Majority learning algorithm outperforms its Anti-majority counterpart. In addition, for all sizes of H tested, the learning time of the Majority algorithm is approximately one third of its $\log_{\frac{4}{3}} |H|$ theoretical bound.

5.2 Anti-majority LRC versus PAC Learning Performance Analysis

For the sake of performance analysis between interactive learning and non-interactive PAC learning (Kearns and Vazirani, 1994), we implemented an equivalent interactive version of the PAC algorithm. Just as in LRC, in each round, this PAC algorithm returns a random

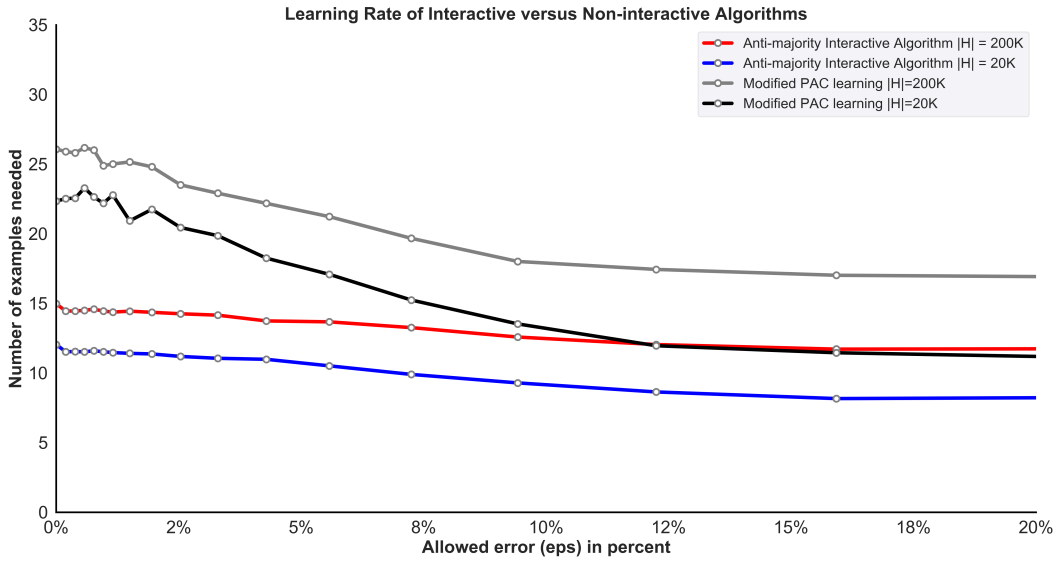

Figure 6: Anti-majority Learning Algorithm versus PAC Learning

consistent hypothesis. However, instead of receiving a counter-example as is the case in LRC, the PAC algorithm receives any random example, which eliminates all hypotheses in H that disagree with it. The PAC algorithm terminates either when it returns a hypothesis that is not ϵ-*bad* or when all the ϵ-*bad* hypotheses are eliminated from H. Note that just as in our implementation of the Arbitrary algorithm, the parameter δ does not play any role in this PAC algorithm.

In Figure 6, we compare the performance of the Anti-majority interactive learning algorithm with the interactive PAC algorithm. For this evaluation, as shown on the X-axis, we varied the error ϵ from 0.0 to 0.2 (0% to 20%). We used randomly generated binary concept classes H with 500 columns and of two different sizes: $|H| = 20000$ and $|H| = 200000$. The Y-axis plots the average number of rounds of interactions for these algorithms. As expected, both the Anti-majority and PAC algorithms require less counter-examples (or examples) when the error tolerance (ϵ) is increased. It can also be seen that the Anti-majority algorithm outperforms the PAC learner particularly when the error tolerance (ϵ) is small. This contrasts with the fact that the $\frac{4 \cdot \log \frac{|H|}{\delta} + 2}{\epsilon} + 1$ theoretical bound on the learning time of any arbitrary interactive learning algorithm is approximately 4 times that that of the $\frac{\log \frac{|H|}{\delta}}{\epsilon}$ theoretical bound of non-interactive PAC learning (Kearns and Vazirani, 1994). Note that non-interactive PAC learning has even higher learning time than interactive PAC learning used in our simulation.

5.3 Majority versus Max-Min Algorithm Performance Analysis

We also implemented the Max-Min algorithm (Angluin and Dohrn, 2017) and compared its performance with the Majority algorithm. Figure 7 shows the results of comparing the performance of the algorithms on randomly generated matrices H with 100 columns and with rows ($|H|$) that ranged from 100 to 2000, in increments of 100. Since the per

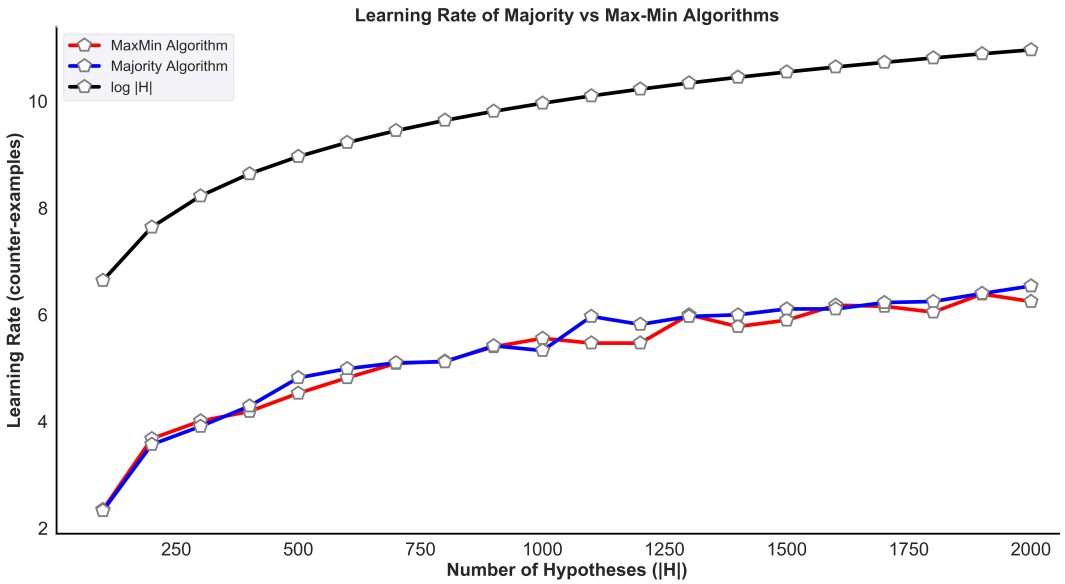

Figure 7: Majority versus Max-Min Learning algorithm

round computation time of the Max-Min algorithm has a quadratic dependence on $|H|$, we were only able to run our tests on these smaller matrices. As can be seen, both the algorithms perform equally well. This contrasts with the fact that the theoretical bound on the Majority algorithm's expected learning time is $\log_{\frac{4}{3}} |H|$ which is 2.41 times more than the $\log_2 |H|$ expected learning time theoretical bound for the Max-Min algorithm. It suggests that the Majority algorithm's theoretical upper bound may not be tight.

6. Discussion

These results demonstrate the potential benefits and limitations of using random counter-examples as a form of feedback in interactive learning. When learners are intelligent, and use the Majority learning algorithm (Algorithm 1), they can learn by seeing an expected $\mathcal{O}(\log |H|)$ counter-examples. As was previously shown (Angluin and Dohrn, 2017), when counter-examples are not chosen randomly, it is difficult for the learner to learn some concept classes. This happens, for instance, when H is simply the $n \times n$ identity matrix. In this case, for any target concept h^* and consistent hypothesis $h \neq h^*$, the teacher will have a choice between exactly two counter-examples. One counter-example will eliminate all hypothesis but h^*, and the other *bad* counter-example will just eliminate h. An adversarial teacher could simply pick the *bad* counter-example every single round and thus it would take $\Omega(|H|)$ time to learn h^* without random counter-examples. While this problem of needing random counter-examples was previously solved by a well known halving algorithm based on majority vote (Littlestone, 1988; Barzdin and Freivald, 1972; Angluin, 1988), that algorithm required that the learner be allowed to make improper queries. On the other hand, our work shows that a teacher who gives random counter-examples solves this problem without having to make any such sacrifices. The $\mathcal{O}(\frac{1}{\epsilon} \log \frac{|H|}{\delta})$ upper bound on the adversarial learner shows that the interactive LRC learner performs no worse than,

and achieves the same asymptotic bound as, the non-interactive PAC learner (Kearns and Vazirani, 1994). This is somewhat surprising because intuitively, providing specific feedback in the form of counter-examples would seem more valuable to the learner than providing randomly sampled examples as is done in the PAC model (Kearns and Vazirani, 1994). It seems that the reason that the bound was not improved was that the learner's ability to be adversarial, or impede the learning process was much more pronounced in the LRC setting than the PAC setting. The reason for this was that in the LRC setting, the learner could choose specific hypothesis on which the teacher's random counter-example would generally make little progress in eliminating ϵ-*bad* hypotheses.

7. Conclusion and Future Work

In this work we provided simple and efficient algorithms for interactively learning non-binary concepts in the recently proposed setting of exact learning from random counter-examples (LRC). One such algorithm is based on majority vote and the other works by randomly selecting hypotheses from a probability distribution over target concepts that is evolved over time. Both these algorithms are shown to have the fastest possible $\mathcal{O}(\log |H|)$ expected learning time and entail significantly lower computation time than previously known algorithms. We also provided an analysis that shows that interactive LRC learning, regardless of the learning algorithm, is at least as efficient as non-interactive Probably Approximately Correct (PAC) learning.

Our future goal is to improve the efficiency of these algorithms on other cost measures. Throughout this paper, our focus has been on minimizing the learning complexity of the algorithms, which is the number of counter-examples needed by the learner in order to correctly identify the target concept. However, calculating the majority hypothesis or updating the teacher's distribution $\mathbb{Q}$ to match the new H, can require iterating over every remaining consistent hypothesis in H, in every round. This can take time $O(|H||X|)$, thus making it computationally expensive, especially since the number of hypothesis in H may grow exponentially in the number of examples (the set of columns X of H). To address the high computational overhead of these tasks, we plan to explore alternative approaches for carrying out these tasks such as by using sampling to trade accuracy for efficiency.

Acknowledgments

I would like to thank my mentor Professor Daniel Hsu who guided my research in the right direction by providing me with background material in the field of computational machine learning, validating the correctness of my proofs, and assisting me in using mathematical notation when writing this paper. Professor Hsu also helped me formulate theoretical models such as the PAC-LRC interactive learning model which served as the framework for my analysis of the upper bound of an Arbitrary LRC algorithm.

References

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

Dana Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

Dana Angluin and Tyler Dohrn. The power of random counterexamples. In *International Conference on Algorithmic Learning Theory*, pages 452–465, 2017.

Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *J. Mach. Learn. Res.*, 18(1):75–109, January 2017. ISSN 1532-4435.

Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *International Conference on Algorithmic Learning Theory*, pages 316–328. Springer, 2008.

J. M. Barzdin and R. V. Freivald. On the prediction of general recursive functions. *Soviet Math. Doklady*, 13:1224–1228, 1972.

Ehsan Emamjomeh-Zadeh and David Kempe. A general framework for robust interactive learning. In *Advances in Neural Information Processing Systems*, pages 7082–7091, 2017.

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

Michael J Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

Wolfgang Maass and György Turán. On the complexity of learning from counterexamples and membership queries. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 203–210. IEEE, 1990.

Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9(2-3):107–145, 1992.

Wolfgang Maass and György Turán. Algorithms and lower bounds for on-line learning of geometrical concepts. *Machine Learning*, 14(3):251–269, 1994.

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