Proceedings of Machine Learning Research vol 247:1–35, 2024 37th Annual Conference on Learning Theory

Active Learning with Simple Questions

Vasilis Kontonis VASILIS@CS.UTEXAS.EDU
The University of Texas at Austin
Mingchen Ma MINGCHEN@CS.WISC.EDU
University of Wisconsin-Madison
Christos Tzamos TZAMOS@WISC.EDU
University of Athens and Archimedes AI

Editors: Shipra Agrawal and Aaron Roth

Abstract

We consider an active learning setting where a learner is presented with a pool S of n unlabeled examples belonging to a domain $\mathcal{X}$ and asks queries to find the underlying labeling that agrees with a target concept $h^* \in \mathcal{H}$.

In contrast to traditional active learning that queries a single example for its label, we study more general *region queries* that allow the learner to pick a subset of the domain $T \subset \mathcal{X}$ and a target label y and ask a labeler whether $h^*(x) = y$ for every example in the set $T \cap S$. Such more powerful queries allow us to bypass the limitations of traditional active learning and use significantly fewer rounds of interactions to learn but can potentially lead to a significantly more complex query language. Our main contribution is quantifying the trade-off between the number of queries and the complexity of the query language used by the learner.

We measure the complexity of the region queries via the VC dimension of the family of regions. We show that given any hypothesis class $\mathcal{H}$ with VC dimension d, one can design a region query family Q with VC dimension $O(d)$ such that for every set of n examples $S \subset \mathcal{X}$ and every $h^* \in \mathcal{H}$, a learner can submit $O(d \log n)$ queries from Q to a labeler and perfectly label S. We show a matching lower bound by designing a hypothesis class $\mathcal{H}$ with VC dimension d and a dataset $S \subset \mathcal{X}$ of size n such that any learning algorithm using any query class with VC dimension less than $O(d)$ must make $\mathrm{poly}(n)$ queries to label S perfectly.

Finally, we focus on well-studied hypothesis classes including unions of intervals, high-dimensional boxes, and d-dimensional halfspaces, and obtain stronger results. In particular, we design learning algorithms that (i) are computationally efficient and (ii) work even when the queries are not answered based on the learner's pool of examples S but on some unknown superset L of S.

Keywords: Active Learning, Region Queries, Efficient Algorithms

1. Introduction

Acquiring labeled examples is often challenging in applications as querying either human annotators or powerful pre-trained models is time consuming and/or expensive. Active learning aims to minimize the number of labeled examples required for a task by allowing the learner to adaptively select for which examples they want to obtain labels. More precisely, in pool-based active learning, the learner has to infer all labels of a pool S of n unlabeled examples, and can adaptively select an example $x \in S$ and ask for its label.

Even though it is known that active learning can exponentially reduce the number of required labels, this is unfortunately only true in very idealized settings such as datasets labeled by one-dimensional thresholds or structured high-dimensional instances (e.g., Gaussian marginals) (Dasgupta et al., 2005; Balcan et al., 2007; Balcan and Long, 2013; Balcan and Zhang, 2017; Awasthi et al., 2017). It is well-known that without such distributional assumptions, even in 2 dimensions, linear classification active learning yields no improvement over passive learning (Dasgupta, 2004, 2005).

Active Learning with Queries To bypass the hardness results and establish learning without restrictive distributional assumptions (Balcan and Hanneke, 2012; Kane et al., 2017; Hopkins et al., 2020b, 2021; Yona et al., 2022; Bressan et al., 2022) introduce enriched queries, where the learner is allowed to make more complicated queries. In this work we follow this paradigm and aim to characterize the trade-off between the number of required queries and their complexity. For example, comparison queries that select two examples and ask which one is closer to the decision boundary Kane et al. (2017) are simple in the sense that they are very easy to implement but also do not improve over passive learning beyond 2-dimensional data. On the other extreme, mistake-based queries such as conditional-class queries (Balcan and Hanneke, 2012) and seed queries (Bressan et al., 2022), where the learner selects a set of examples from the dataset and requests an example with a proposed label, allow the learner to label the whole dataset with very few queries but are very complicated in the sense that each one requires transfering a lot of information from the learner to the labeler (essentially the learner has to transfer their whole dataset) making them impractical. Motivated by those gaps in the literature, we ask the following natural question.

Can we design simple query classes that simultaneously lead to active learning algorithms with low query complexity?

Example: 2-d Halfspaces Consider the 2-dimensional halfspace learning problem shown in Figure 1. A learner is given a complicated unlabeled dataset $S \subset \mathbb{R}^2$ labeled by some unknown halfspace h^* and wants to learn the labels of examples in S. Consider the shadowed region T in Figure 1. There is a significant fraction of examples contained in T and all of them have the same label. If one can verify this fact, then a huge progress is made for the learning task. However, if the learner can only use label queries, then to verify this fact, every example in this region has to be queried once. This is why vanilla active learning has a high query complexity. On the other hand, the region T is independent on the dataset S. The structure of T is so simple that to describe T for the labeler, the learner only needs to send information about the two halfspaces that define T. Once the labeler describes the region T for the labeler, the labeler can easily respond to the learner and the verification problem can be solved in a single round of simple interaction. This implies that a simple query language may help a lot in learning and motivates the following learning model.

Definition 1 (Active Learning with Region Queries) *Let $\mathcal{H}$ be a class of binary hypotheses over a domain $\mathcal{X}$ and let $h^* \in H$ be a true hypothesis that labels the examples in $\mathcal{X}$ as positive or negative. Given a set of n examples $S \subseteq \mathcal{X}$, a learner $\mathcal{A}$ wants to learn the labels of examples in S by adaptively submitting* ***region queries*** *to a labeler from a query family Q. In particular, a region query $q = (T, z) \in Q$ consists of a subset T of $\mathcal{X}$ and a label $z \in \{\pm 1\}$. The labeler has a (possibly unknown) labeling domain L such that $S \subseteq L \subseteq X$ and after receiving a query (T, z), answers whether all examples in $L \cap T$ have label z under the true hypothesis h^*.*[1]

We remark that an important feature of region queries is that the query family Q is defined independently on the dataset S. Such an additional requirement not only captures the feature that checking whether an example with a given label exists in a region could be much easier than labeling every example in the region but also captures the features of many other applications such as human learning (Shanks and John, 1994), theorem proving (Davis et al., 1962) and learning via language models (Polu and Sutskever, 2020).

[1] If $L \cap T = \emptyset$, then the labeler can output an arbitrary answer.

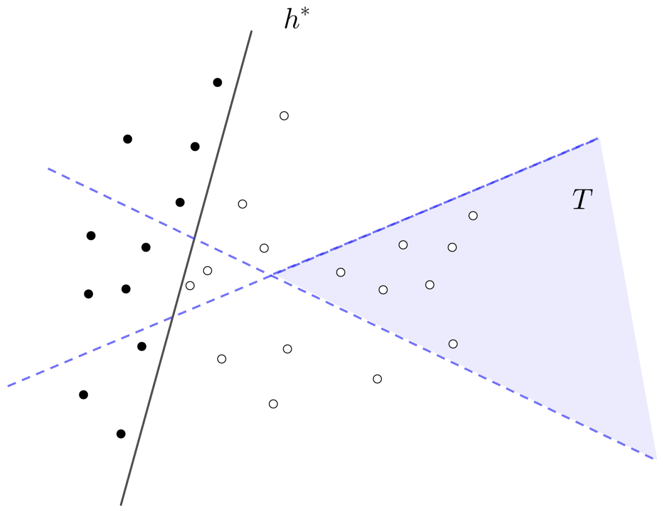

Figure 1: Learning 2-dimensional Halfspaces with Region Queries

2-d Halfspaces (cont.) We now revisit the previous example where a set of n points $S \subseteq \mathbb{R}^2$ are labeled by some hidden halfspace h^* to illustrate how region queries can be used to efficiently obtain the labels of all examples. It is well-known that (Megiddo, 1985), for any set of n points in $\mathbb{R}^2$, one can compute in $O(n)$ time, two lines that partition these points into 4 regions, each of which contains at least $\lfloor n/4 \rfloor$ points, see Figure 1. We notice that h^* can have at most 2 intersections with the two lines, which implies at least one of the four regions lies on the one side of h^*. Now, if we make region queries over these four regions, then with at most 8 region queries, we can identify a region T as in Figure 1, which contains only points with the same label and thus label a quarter of S. If we repeat this process over the remaining examples $O(\log n)$ rounds, we successfully infer the label of every example with $O(\log n)$ region queries. In particular, the algorithm used here does not rely on the label of a single example in the dataset to make an update, and every query used by the algorithm is binary. Furthermore. the query family $Q = \{(T, z) \mid z \in \{\pm 1\}, T \text{ is an intersection of two halfspaces}\}$ is predetermined before the learner sees the dataset S. Thus, no matter how complicated the dataset S is, in a single round of interaction, the learner only needs to describe the two halfspaces to the labeler and let the labeler check the answer to the query. This requires sending just 4 numbers plus a binary label. Motivated by our success in this example we ask:

Given a hypothesis class $\mathcal{H}$, can we design a region query family Q, where the region used in a query comes from a simple set family, such that $O(\log n)$ queries suffice to perfectly label every set of n examples? If this is true, how complicated should the set family be?

1.1. Our results

Characterizing the Complexity of Learning with Region Queries We measure the complexity of the region query class using the VC dimension of the family of regions. VC dimension characterizes the capacity of a set family and is one of the most well-studied complexity measures in learning theory. Queries from a query family with bounded VC dimension can be communicated using few bits: for a finite domain $\mathcal{X}$, and a set family C of VC dimension d, communicating a set $c \in C$ only requires $O(d \log(|\mathcal{X}|))$ bits.

Our first main result shows that if the hypothesis class $\mathcal{H}$ has a VC dimension d, we can always design a simple query family Q with VC dimension at most $O(d)$ and use it to perfectly label any set of n examples with $O(d \log n)$ regions queries. Formally, we have the following theorem.

Theorem 2 *Let $\mathcal{X}$ be a space of example and $\mathcal{H}$ be a hypothesis class over $\mathcal{X}$ with VC dimension d. There is a region query family Q over $\mathcal{X}$ with VC dimension at most $6d$ and a learning algorithm $\mathcal{A}$ such that for any set of n examples $S \subseteq \mathcal{X}$ labeled by any true hypothesis $h^* \in \mathcal{H}$, $\mathcal{A}$ makes $O(d \log n)$ region queries from Q and correctly label every example in S, if the labeling domain $L = S$.*

In particular, the $O(d \log n)$ query complexity in Theorem 2 matches the lower bound for the query complexity of active learning with arbitrary binary-valued queries in Kulkarni et al. (1993) and thus is essentially information-theoretically optimal. Given Theorem 2, an immediate question is in general, whether it is possible to quickly learn $\mathcal{H}$ with an even simpler query class (with VC dimension $o(d)$). Our next main result gives a negative answer firmly. We give a matching lower bound showing that unless the hypothesis class $\mathcal{H}$ has a good structure, a region query family with VC dimension $\Omega(d)$ is necessary to achieve query complexity $O(\log n)$. Formally, we have the following theorem.

Theorem 3 *For every $d \in N^+$ and $n \geq d$ large enough, there exists a space of examples $\mathcal{X}$ and a hypothesis class $\mathcal{H}$ over $\mathcal{X}$ with VC dimension d such that there exists a set of n example S such that for every region query family Q over $\mathcal{X}$ with $VC \dim(Q) \leq (d-2)/3$ and every active learning algorithm $\mathcal{A}$, there exists a true hypothesis $h^* \in \mathcal{H}$, such that if $\mathcal{A}$ makes less than $\mathrm{poly}(n)$ region queries from Q, then with probability at least $1/3$, some example $x \in S$ is labeled incorrectly by $\mathcal{A}$. In particular, this even holds when $\mathcal{A}$ knows the labeling domain $L = S$.*

Theorem 2 and Theorem 3 together give a perfect trade-off between the complexity of the query family and the query complexity and thus show that the VC dimension is a good measure for the performance of region queries. We want to remark that Theorem 3 not only holds in our learning model where queries are binary but also holds in the stronger model studied in (Balcan and Hanneke, 2012; Bressan et al., 2022), where a counter-example is also returned in each round of interaction. We also remark that Theorem 3 gives an optimal lower bound that matches Theorem 2 only in a minimax perspective. In general, it could be the case where for a very special hypothesis class $\mathcal{H}$, we can design a query family with a much smaller VC dimension than the one of $\mathcal{H}$ but still achieve the information-theoretically optimal query complexity. These examples will be shown later. Furthermore, given a pair of hypothesis classes and query class $(\mathcal{H}, Q)$, we actually come up with a combinatorial characterization of the query complexity of learning $\mathcal{H}$ using Q. However, since the result is far from the central theme of this paper, we leave it for Appendix D.

Efficient Learning Algorithms for Natural Hypothesis Classes Although Theorem 2 gives an algorithm that can perfectly label every subset of n examples with an optimal query complexity, the algorithm itself is not efficient, as it needs to solve optimization problems over the hypothesis class, which is usually exponentially large with respect to the input. In this work, we also focus on designing query families and learning algorithms for some natural hypothesis classes and obtain stronger results. Specifically, our learning algorithms are computationally efficient and work even when queries are not answered based on the dataset S but on any unknown superset L of S. These results are summarized as follows.

Hypothesis Class $\mathcal{H}$	VC-dim(Q)	Query Complexity	Efficient?	Labeling Domain
General	$O(d)$	$O(d \log n)$	No	$L = S$
Union of d Intervals	$O(1)$	$O(d \log n)$	Yes	$L \supseteq S$
Axis Parallel Boxes	$O(\log d)$	$O(d \log n)$	Yes	$L \supseteq S$
Halfspaces	$\tilde{O}(d^3)$	$\tilde{O}(d^3 \log n)$	Yes	$L \supseteq S$

Table 1: Summary of the algorithmic results of Theorem 2 and Theorem 4 for a hypothesis class $\mathcal{H}$ of VC dimension $\Theta(d)$ and a dataset S of size n.

Theorem 4 *There is a computationally efficient algorithm $\mathcal{A}$ and a query class Q such that for any set S of n examples, $\mathcal{A}$ learns the labels of S perfectly by making region queries to a labeler with labeling domain an unknown set $L \supseteq S$:*

1. *For unions of d intervals, Q has VC dimension 2, and $\mathcal{A}$ makes $O(d \log n)$ queries.*
2. *For axis parallel boxes in $\mathbb{R}^d$, Q has VC dimension $O(\log d)$, and $\mathcal{A}$ makes $O(d \log n)$ queries.*
3. *For halfspaces in $\mathbb{R}^d$, Q has VC dimension $\tilde{O}(d^3)$, and $\mathcal{A}$ makes $\tilde{O}(d^3 \log n)$ queries.*

We note that for the first two cases, the VC dimension of the query class is significantly smaller than the VC dimension of the hypothesis class which is $\Theta(d)$. In the case of halfspaces, the VC dimension of Q and the query complexity is worse than that given in Theorem 2 but applies in a significantly more general setting and is computationally efficient. The cubic dependence on d can be improved to quadratic if the learner provides counter-examples instead of binary answers to our region queries. We leave the detail discussion on this improvement to Appendix C.3.

1.2. Connection with Other Learning Models and Related Work

Active Learning with Enriched Queries The study of active learning with enriched queries can be traced to the literature of exact learning (Angluin, 1988; Balcázar et al., 2001, 2002; Chase and Freitag, 2020). More recently, the focus has been shifted from general queries to more problem-dependent queries such as mistake-based queries (Balcan and Hanneke, 2012; Bressan et al., 2022), clustering-based queries (Ashtiani et al., 2016; Mazumdar and Saha, 2017; Bressan et al., 2021; Del Pia et al., 2022; Xia and Huang, 2022), comparison-based queries (Kane et al., 2017, 2018; Xu et al., 2017; Hopkins et al., 2020b,c,a), separation-based queries (Har-Peled et al., 2021) and derivative-based queries (Ben-Eliezer et al., 2022). In this work, we study active learning with region queries for both general hypothesis classes and concrete learning problems.

Mistake-Based Query and Self-Directed Learning The region queries we study in this paper fall into the category of mistake-based queries (Angluin, 1988; Maass and Turán, 1992; Balcan and Hanneke, 2012; Bressan et al., 2022). The study of mistake-based queries can be traced to the study of learning with equivalence or partial equivalence queries (Angluin, 1988; Maass and Turán, 1992). Though named differently, a typical mistake-based query can be understood as follows. A learner selects a subset of examples $T \subset \mathcal{X}$, proposes a possible labeling for examples in T, and submits the information to a labeler. The labeler will return an example $x \in T$ labeled incorrectly by the learner or return nothing when every example in T is labeled correctly. We will discuss in Appendix C.3.1, if an arbitrary complicated subset T and any possible labeling are allowed to be used, a learner could use mistake-based queries to implement online learning algorithms or self-directed learning algorithms (Goldman and Sloan, 1994) and easily obtain active learning algorithms with low query complexity. Our query model has several differences from the previous work. (i) Unlike all previous work on mistake-based queries, a region query is a binary query and does not require a counter-example to be returned. (ii) Unlike (Angluin, 1988; Maass and Turán, 1992), a region query is not answered based on the example space $\mathcal{X}$ but based on some labeling domain $S \subseteq L \subseteq \mathcal{X}$ (usually $L = S$). In general, we should not hope to obtain useful information from examples not in the dataset. (iii) Unlike (Balcan and Hanneke, 2012; Bressan et al., 2022), we require the learner to design a query family Q with finite VC dimension before seeing the dataset S and thus we cannot simply design an active learning algorithm by reducing it to online/self-directed learning.

Learning Halfspace with the Power of Adaptivity The class of halfspaces is one of the most well-studied hypothesis classes under active learning. Dasgupta (2004) shows that to perfectly learn the labels of a set of n points in $\mathbb{R}^2$ labeled by some halfspace, vanilla active learning needs to make $\Omega(n)$ label queries. Since then, a large body of works (Dasgupta et al., 2005; Balcan et al., 2007; Balcan and Long, 2013; Balcan and Zhang, 2017; Awasthi et al., 2017) have been done to understand under which distribution vanilla active learning can learn a halfspace with few queries. On the other hand, the query complexity of learning halfspaces in the distribution-free setting is much less understood. Kane et al. (2017) points out that with the help of comparison queries, one can efficiently learn a 2-dimensional halfspace with a query complexity $O(\log n)$. However, in the same work, they point out that such an improvement disappears in $\mathbb{R}^3$. Recently, two remarkable results have been done to understand the query complexity of learning halfspaces in the distributional free setting. The first one is Hopkins et al. (2020c), where they show that if one can query the label of any point in $\mathbb{R}^d$, then $\tilde{O}(d \log n)$ queries are sufficient to perfectly label n examples. The second one is Bressan et al. (2022), in which they show without restriction on the complexity of the mistake-based query, they can efficiently learn a γ-margin halfspace with $\tilde{O}(d \log(1/\gamma))$ queries. Our halfspace learning algorithm does not rely on acquiring additional information from $\mathcal{X} \setminus S$ or using very complicated query classes but is still able to achieve a query complexity of $\mathrm{poly}(d, \log n)$.

Organization of the Paper In Section 2, we discuss our results for general hypothesis classes. We give proof sketches for Theorem 2 and Theorem 3 in Section 2.1 and Section 2.2. In Section 3, we discuss how to design query classes and efficient active learning algorithms for natural classes. In Section 3.1 and Section 3.2, we study the class of the union of k-intervals and the class of high dimensional boxes. In Section 3.3, we discuss our main results on efficient active learning algorithms for halfspaces. Due to the limited space, we present the notations and detailed proofs in the Appendix.

2. Active Learning for General Hypothesis Class Using Simple Region Query

2.1. Construction of Simple Query Classes for General Hypothesis Classes

In this section, we give an overview of the proof of Theorem 2 and leave the full proof and detailed discussion of Theorem 2 to Appendix B.1. Before diving into the proof, we first give an overview of why the previous works on mistake-based queries result in using query families with unbounded query complexity. Previous work such as (Maass and Turán, 1992; Balcan and Hanneke, 2012) design learning algorithms based on the fact that it is possible to use region queries to implement the Halving algorithm. An algorithm of this style predicts a label for each example in S via majority voting, submits examples with positive predictions, and examples with negative predictions, and gets one example on which majority voting makes a mistake (such a mistake can be found via binary search if the query is binary). In this way, hypotheses that predict incorrectly on this example cannot be the true hypothesis, and the size of the version space is shrunk by half. Since the majority voting could behave arbitrarily complicated over an arbitrary set of examples, the query family used by the algorithm could also be arbitrarily complicated. This suggests a new algorithmic framework should be come up with to break the bottleneck.

The intuition behind our algorithm is as follows. Assume the examples in S have been ordered as $x^{(1)}, \dots, x^{(n)}$. We consider $H^{(0)}$, the restriction of $\mathcal{H}$ over the dataset S. If we make a label query for $x^{(1)}$, then such a label query might not be very helpful because most of the hypotheses in $H^{(0)}$ could label this example in the same way, for example, $y^{(1)} \in \{\pm 1\}$. Let's assume we are in this case and define $H^{(1)} := \{h \in H^{(0)} \mid h(x^{(1)}) = y^{(1)}\}$. Similarly, a label query for $x^{(2)}$ is also not that useful, since many hypotheses in $H^{(1)}$ might label $x^{(2)}$ by some $y^{(2)}$. Assuming we are in this case, then we have a new class $H^{(2)}$ defined based on $H^{(1)}$. Although each single label query is not useful, if we repeat this process, at some point $t \in [n]$, the remaining hypothesis class $H^{(t)}$ should have a proper size. i.e $|H^{(t)}|/|H^{(0)}| \in (1/3, 2/3)$. This implies that after t label queries no matter what answer we get, the size of the version space is shrunk by a constant factor. Notice that these t label queries can be safely replaced by 2 region queries, $(\{x \mid x = x^{(i)}, i \in [t], h'(x) = 1\}, 1)$ and $(\{x \mid x = x^{(i)}, i \in [t], h'(x) = -1\}, -1)$, where h' is an arbitrary hypothesis whose restriction over S is in the class $H^{(t)}$. By Sauer's lemma, $|H^{(0)}| \leq O(n^d)$. So, if we repeat the above procedure $O(d \log n)$ times, we learn the labels of examples in S. Up to now, the problem has been almost solved, but the regions where we make queries still depend on the dataset S. However, the analysis above works for any order of S, if there is a natural order o for $\mathcal{X}$, then the constraint $x = x^{(i)}$ for some $i \in [t]$ can be simply replaced by $o(x) \in [o(x^{(1)}), o(x^{(t)})]$, because $L = S$. Thanks to the well-known well-ordering theorem, such a linear order exists for every non-empty space $\mathcal{X}$. Thus, we can construct the query family Q using $\mathcal{H}$, $\bar{\mathcal{H}}$(the set of negation hypothesises in $\mathcal{H}$), and the natural linear ordering defined in $\mathcal{X}$, which gives a simple query class.

2.2. Lower Bound on the VC Dimension of the Query Class

In this section, we give an overview of the proof of Theorem 3, showing a matching lower bound for Theorem 2. The full proof and more detailed discussions are presented in Appendix B.2.

We will assume $\mathcal{X}$ to be a space of n examples and $L = S = \mathcal{X}$. i.e. The labeling domain, the dataset to be labeled, and the example space are the same. Suppose there is some subset $C^* \subseteq S$ of size k and we want to distinguish hypothesis h_0, which labels every example in C^* positive and everything else negative, and the other k hypothesis $h_1, \dots, h_k$, each of which only differs from h_0 at a single example in C^*. Let's assume the learning algorithm is using a fixed region query class Q to learn. For any query $(T, z) \in Q$, if T has an intersection with both C^* and $S \setminus C^*$, then it will provide no useful information(even if some example $x \in T \cap S$ with label $-z$ is also returned), because we know that $q(T, z) = 0$ always holds. Furthermore, to distinguish the two cases, those regions $T \subseteq C^*$ should cover all examples in C^*, otherwise, an example $x \in C^*$ is not involved in any query. As we will show later, the optimal solution to this set cover instance roughly serves as a lower bound of the query complexity in this special instance. In particular, if every $T \subseteq C^*$ as size at most t, then the query complexity should be at least $\Omega(k/t)$.

So far, we have established a hard instance for a fixed query class. The most difficult part of our construction is to generalize the above instance so that it is hard for *every* query class Q with

VC dimension $O(d)$, where massive subsets of $\mathcal{X}$ would be possible to be involved. We use several key techniques to overcome this difficulty. The first one is the following observation. If we have $N > |\mathrm{dom}(Q)|$ subsets $C_1, \ldots, C_N \subseteq \mathcal{X}$ of size k such that the pairwise intersection of C_i, C_j is at most t, then there must be at least one C_i such that if $T \subseteq C_i$ and $T \in \mathrm{dom}(Q)$ then it must be the case $|T| \leq t$. Sauer's lemma tells us that each set family over $\mathcal{X}$ with VC dimension $O(d)$ contains at most $O(n^d)$ different sets. Thus, if we set up the above N to be $O(n^d)$, then we can embed the hard instance we mentioned above into each C_i so that any learning algorithm uses any query class with VC dimension $O(d)$ has query complexity at least $\Omega(k/t)$. In particular, we will see later, that the hypothesis class we use here has VC dimension $O(t)$. So the final step is to show we can construct these subsets C_i such that $k = \mathrm{poly}(n)$ while $t = O(d)$. To show this, we make use of the result in (Beideman and Blocki, 2014), which explicitly constructs set families with low pairwise intersections. This is why intuitively a query family with $\Omega(d)$ VC dimension is also necessary for a query complexity of $O(\log n)$. We want to remark that the construction of the example space $\mathcal{X}$ in theorem 3 is fully combinatorial. So, given any large enough space of examples, we can embed the hard instance we construct in Theorem 3 into that space to get a corresponding hard instance. Formally, we have the following corollary, which gives a stronger statement of Theorem 3. We refer the readers to Section B.3 for the proof of Corollary 5

Corollary 5 *There is a space of examples $\mathcal{X}$ such that for every $d \in N^+$ and $n \geq d$ large enough, there exists a hypothesis class $\mathcal{H}$ over $\mathcal{X}$ with VC dimension d such that there exists a set of n example S such that for every region query family Q over $\mathcal{X}$ with $VC\,\dim(Q) \leq (d-3)/3$ and every active learning algorithm $\mathcal{A}$, there exists a true hypothesis $h^* \in \mathcal{H}$, such that if $\mathcal{A}$ makes less than $\mathrm{poly}(n)$ region queries from Q, then with probability at least $1/3$, some example $x \in S$ is labeled incorrectly by $\mathcal{A}$. In particular, this even holds when $\mathcal{A}$ knows the labeling domain $L = S$.*

3. Efficient Active Learning with Simple Questions for Natural Hypothesis Classes

In Section 2.1, we have shown that given a hypothesis class $\mathcal{H}$ with dimension d, we can construct a query class Q with dimension $O(d)$, so that a learner can use Q to learn $\mathcal{H}$ with query complexity $O(d \log n)$. However, the learning algorithm we use Section 2.1 is not computationally efficient and works when the labeling domain is the same as the dataset. i.e. $L = S$. Such an assumption might be strong for some applications. For example, if a learner is interacting with a large language model, then the language model cannot know the learner's dataset S in advance and thus will answer the learner's query based on an unknown and potentially much larger superset L of S. In this section, we focus on designing learning algorithms with low query complexity for natural hypothesis classes including the union of k intervals, high dimensional boxes, and d-dimensional halfspaces, for which the query complexities are $\Omega(n)$ in the vanilla active learning setting. Our algorithms are not only efficient but also work even when the queries are not answered based on the learner's dataset S but on any unknown superset L of S. In particular, we will see that when the hypothesis class has a good structure, the query family Q used by our algorithm can have $O(\log d)$ or even a constant VC dimension. Due to space limitations, we leave the full proofs and detailed discussions to Appendix C.

3.1. Learning Union of k Intervals

The first hypothesis class we study is the class of the union of k intervals, perhaps one of the simplest classes studied in the active learning literature. In the following theorem, we design an efficient learning algorithm that uses $O(k \log n)$ "interval" queries to learn a target hypothesis over an arbitrary set of n examples.

Theorem 6 *Let $\mathcal{X} = \mathbb{R}$ be the space of examples and $\mathcal{H} = \{h \mid \exists [a_i, b_i], i \in [k], s.t. h(x) = 1 \iff x \in \cup_{i=1}^{k} [a_i, b_i]\}$ be the class of union of k intervals over $\mathbb{R}$. Let I be the class of intervals over $\mathbb{R}$ and query family $Q = \{(T, z) \mid T \in I, z \in \{\pm 1\}\}$. There is a learner $\mathcal{A}$ such that for every subset of n examples S, labeled by any $h^* \in \mathcal{H}$ and for every labeling domain $S \subseteq L$(possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T+n)k \log n)$ time, makes $O(k \log n)$ queries from Q and labels every example in S correctly, where T is the running time to implement a single region query.*

We give the proof overview of Theorem 6 here and leave the full proof for Appendix C.1. The main idea that we use is that, any $h^* \in \mathcal{H}$ partitions $\mathbb{R}$ into $2k+1$ intervals $I_1, \dots, I_{2k+1}$. Examples in the same interval have the same label, while examples in two adjacent intervals have different labels. So, instead of learning k intervals at the same time, it is sufficient to design a learning algorithm that learns examples in S in the left-most interval. This can be done easily using interval queries and binary search. We order S by $x^{(1)} < \cdots < x^{(n)}$. Suppose $x^{(1)} \in I_1$ and has label $y = -1$. Then no matter which L the labeler has, using $O(\log n)$ interval queries via binary search, we are able we find i^* such that $q([x^{(1)}, x^{(i^*)}], -1) = 1$ and $q([x^{(1)}, x^{(i^*+1)}], -1) = 0$. After this, we can safely label example $x^{(1)}, \dots, x^{(i^*)}$ by negative. In particular, examples in $I_1 \cap S$ are all labeled in this iteration because $I_1 \cap S \subseteq [x^{(1)}, x^{(i^*)}] \cap S$. By repeating the procedure $O(k)$ times, we perfectly label S.

3.2. Learning High-Dimensional Boxes

Our next result, Theorem 7, gives an efficient learning algorithm for learning a high dimensional box with low query complexity. The full proof of Theorem 7 is presented in Appendix C.2.

Theorem 7 *Let $\mathcal{X} = \mathbb{R}^d$ be the space of examples and $\mathcal{H} = \{\prod_{i=1}^{d} [a_i, b_i] \mid a_i, b_i \in [-\infty, \infty]\}$ be the class of axis-parallel boxes in $\mathbb{R}^d$ that labels $\mathcal{X}$. There is a query class Q over $\mathbb{R}^d$ with VC dimension $O(\log d)$ and an efficient algorithm $\mathcal{A}$ such that for every set of n examples $S \subseteq \mathbb{R}^d$, every target hypothesis $h^* \in \mathcal{H}$ and for every labeling domain $S \subseteq L$(possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T+n)d \log n)$ time, makes $O(d \log n)$ queries from Q and labels every example in S correctly, where T is the running time to implement a single region query.*

The idea behind Theorem 7 is similar to that of Theorem 6. Instead of learning the target box $h^* = \prod_{i=1}^{d} [a_i^*, b_i^*]$ directly, we learn each boundary a_i^*, b_i^* separately. Let b_i^* be a boundary of h^*. Suppose we can learn some $\hat{b}_i \leq b_i^*$ such that for every $x \in S$, if $x_i > \hat{b}_i$ then x is labeled by -1. Then the box $\hat{h} = \prod_{i=1}^{d} [\hat{a}_i, \hat{b}_i]$ perfectly label S. This is because if an example $x \in S$ is labeled negative by $\hat{h}$, then x must violate one of the constraints of $\hat{h}$ and have true label -1. On the other hand, since $\hat{h} \subseteq h^*$, every example labeled positive by $\hat{h}$ must also have true label $+1$. In fact, we can learn such a $\hat{b}_i$ via region queries of the form $(\{x \mid x_i \geq c\}, -1)$. If we order S such that $x_i^{(1)} \leq \cdots \leq x_i^{(n)}$, then we can use binary search with $O(\log n)$ queries to find the i^* such that $q(\{x \mid x_i \geq x_i^{(i^*)}\}, -1) = 0$ but $q(\{x \mid x_i \geq x_i^{(i^*+1)}\}, -1) = 1$. We will show in Appendix C.2 that $\hat{b}_i = x_i^{(i^*)}$ is a good approximation of b_i^* that we want no matter which L the labeler uses. This gives the idea of the query complexity in Theorem 7. In particular, the query class we use is defined by the set of axis-aligned halfspaces, which has a VC dimension $O(\log d)$.

3.3. Learning Arbitrary High-Dimensional Halfspaces

Our central results for efficient learning are on half-spaced learning problems. Before this work, even assuming the labeling domain $L = S$, there are no known efficient algorithms for the class of halfspaces that can achieve a query complexity of $\mathrm{poly}(d, \log n)$, even using arbitrarily complicated query classes. Previous work by (Bressan et al., 2022), assumes each example in S has a margin of γ with respect to the target w^* and some counter-example $x \in S \cap T$ with label $-y$ is returned if $q(S \cap T, y) = 0$. The query complexity of their algorithm is $O(d \log(d/\gamma))$ and could be potentially $\Omega(n)$ if γ is very small. However, we want to point out that if we are allowed to communicate arbitrary subsets of the dataset S, then by reducing the active learning problem to self-directed learning(Goldman and Sloan, 1994), it is easy to design an efficient halfspace learning algorithm with an expected query complexity $O(d \log^2 n)$ using the idea of Haussler et al. (1994) on binary prediction over random points. We summarize the discussion as the following theorem and leave the full proof and more detailed discussion for Appendix C.3.1.

Theorem 8 *Let $\mathcal{X} = \mathbb{R}^d$ be the space of examples and $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. Let $Q = \{(T, z) \mid z \in \{\pm 1\}, T \subseteq 2^{\mathbb{R}^d}\}$ over $\mathbb{R}^d$ be the query class that contains any subset of $\mathbb{R}^d$. There is an efficient algorithm $\mathcal{A}$ such that for every set of n examples S, labeled by any $w^* \in \mathcal{H}$ and for every labeling domain $S \subseteq L$ (possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T + B) d \log^2 n)$ time, makes $O(d \log^2 n)$ queries from Q in expectation and labels every example in S correctly, where T is the running time to implement a single query and B is the bit complexity of S.*

Although efficiently learning a halfspace with an arbitrarily complicated query class is easy, designing an efficient learning algorithm using a query class with low VC dimension is significantly more challenging, especially when a query (T, z) is answered based on an unknown superset L of S.

There are several difficulties with this problem. First, as (T, z) is checked over $L \supseteq S$, there is no way to find an example $x \in S$ with label $-z$, when $q(T, z) = 0$. It could be the case that every example in $T \cap S$ has label z but some hidden $x \in T \setminus S$ with label $-z$ makes $q(T, z) = 0$. Such difficulty makes it very hard to learn from mistakes without sending the whole dataset to the labeler, which results in a very complicated query family. The second difficulty is how to design the query class so that we can get enough information from a single query. As L is unknown to the learner if a region T is too large, it is very likely that T contains both positive examples and negative examples in L, and such queries (T, z) may always return 0 to the learner, sending no information. On the other hand, if a region T is very small, then each query can only send us very little information because if $L = S$, each query can only provide information about very few examples in S. We overcome the above difficulties and obtain the following theorem.

Theorem 9 *Let $\mathcal{X} = \mathbb{R}^d$ be a space of examples and let $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. There is an algorithm and a query class Q with VC dimension $\tilde{O}(d^3)$ such that for every subset of n examples $S \subseteq \mathcal{X}$, every labeling domain L with $S \subseteq L$ and every target hypothesis $w^* \in \mathcal{H}$ and every $\alpha \in (0, 1)$, it in expectation makes $O(d^3 \log^2 d \log(1/\alpha))$ queries from Q, runs in $\mathrm{poly}(d, n, T)$ time and labels $(1 - \alpha)$ fraction of examples in S correctly, where T is the running time of implement a single region query from Q. In particular, the algorithm makes $O(d^3 \log^2 d \log n)$ queries from Q and labels every example in S correctly in time $\mathrm{poly}(d, n, T)$.*

We want to remark that the query class Q we use has a VC dimension $\tilde{O}(d^3)$. Such a dependence could be improved to $\tilde{O}(d^2)$, if an example $x \in T \cap L$ with label $-z$ is also returned when $q(T, z) = 0$. For a more detailed discussion, we point the reader to Appendix C.3.2.

A particularly surprising part of our result is that if we only want to perfectly $(1 - \alpha)$ fraction of the examples in S, then the query complexity of our algorithm even does not depend on the size of S. We present the full proof of Theorem 9 in Appendix C.3.2 and give the intuition of why it is possible to get such a result. We start by assuming our dataset $S \subseteq S^{d-1}$ has the following nice property. For every $w \in S^{d-1}$, β-fraction of the examples in S have margin γ with respect to w. i.e. $|w \cdot x| \geq \gamma$. We create an $\gamma/2$-cover, $\mathcal{N} = \{u_1, \ldots, u_\ell\}$ for S^{d-1} and associate a ball $B(u_i)$ with radius $\gamma/2$ for each u_i. Then each $x \in S$ must belong to some $B(u_i)$. Furthermore, if example $x \in B(u_i)$ has margin γ with respect to w^*, then every point inside $B(u_i)$ has the same label as x. Since β fraction of the examples in S have γ margin with respect to w^*, if we make 2 queries for each $B(u_i)$, then we can safely label at least βn examples in S. So, if this margin assumption recursively holds after we remove examples we have labeled, we can repeat such a procedure $O(\log n)$ times and finally perfectly label every example in S. However, such an intuition does not directly lead to efficient learning algorithms. There are two issues we need to overcome. First, the above margin assumption in general can not be satisfied recursively and sometimes is even not satisfied by the original dataset S. Second, even if $\gamma = 1/\mathrm{poly}(d)$, $(1/\gamma)^{O(d)}$ queries have to be made each round, due to the large size of $\mathcal{N}$, which is not computationally efficient. We now give a sketch of how to address these two issues.

The first issue can be overcome with Forster's Transform (Forster, 2002). Roughly speaking, given any set of n examples $S \subseteq \mathbb{R}^d$, Forster's transform finds a subspace V of dimension k containing at least k/d fraction of examples in S and a matrix A such that $f_A(S \cap V) = \{f_A(x) := Ax/\|Ax\| \mid x \in S \cap V\}$ satisfies the above margin assumption with $\gamma = 1/(2\sqrt{k})$ and $\beta = 1/(4k)$. In particular, Diakonikolas et al. (2021, 2023b) shows that given any set of n examples S, we can compute in polynomial time a Forster's transform for S. This gives us a way to recursively find a large fraction of examples that satisfy the margin assumption and solve the first issue. So, for now, we assume S satisfies the margin assumption with $\gamma = 1/(2\sqrt{d})$ and $\beta = 1/(4d)$.

The technique we use to overcome the second issue is inspired by the modified perceptron algorithm used by Blum et al. (1998). Instead of creating a cover for S and doing a brute-force search, we will use queries to implement the modified perceptron algorithm to learn a halfspace $\hat{w}$ that can correctly label every example that has a large margin with respect to $\hat{w}$. The modified perceptron algorithm works as follows, it maintains a hypothesis w_t and makes an update $w_{t+1} = w_t - x_t(x_t \cdot w_t)$ if x_t is a point that is misclassified by w_t. Furthermore, if every x_t we use for an update has a margin $\Omega(1/\sqrt{d})$ with respect to w_t, then after $O(d \log d)$ updates, each example with a margin $\Omega(1/\sqrt{d})$ with respect to w_t is correctly classified by w_t. As we mentioned previously, finding such an example where we make a mistake is hard. However, we will show that using an x_t such that $(x_t \cdot w_t)(x_t \cdot w^*) \leq 1/\mathrm{poly}(d)$ to make an update is enough to achieve the same guarantee. In particular, such an x_t can be found using binary search together with $O(d \log d)$ region queries that are defined by $O(d)$ linear inequalities. To see why this is true, consider the region $T = \{x \mid v_t \cdot x \geq 1/(2\sqrt{d})\}$, where v_t is the unit vector parallel to w_t. According to the margin assumption, a large fraction of the examples are contained in T. So if $q(T, 1) = 1$, we safely label a lot of examples correctly. Otherwise, there is at least one point in T(not necessarily in S) that is misclassified by w_t and if we find such a point we can use it to make a perceptron update. In this case, we partition the region T into small strips $T_i := \{x \mid v_t \cdot x \in [a_i, b_i]\}$, where $|b_i - a_i| \leq 1/\mathrm{poly}(d)$.

With binary search, we can use $O(\log d)$ queries to find one such T_i that contains one point that is misclassified by w_t. Now, denote by $u_1, \ldots, u_{d-1}$ a standard basis of the subspace orthogonal to w_t. Using the same binary search approach over T_i for each direction u_i, we will finally find a small box $B \subseteq T_i$ with diameter $1/\text{poly}(d)$ that contains at least one point that is misclassified by w_t. Since B has a diameter only $1/\text{poly}(d)$, this implies that each point x_t in B is very close to the decision boundary and satisfies $(x_t \cdot w_t)(x_t \cdot w^*) \leq 1/\text{poly}(d)$. So, we can choose any point in B to make a perceptron update and after doing this $O(d \log d)$ rounds, we learn a w_t that safely classifies many examples correctly. We remark that there is still a small issue in the above analysis. Since L is unknown to our algorithm, it could be the case during the binary search a region Z we query has an empty intersection with L, and an undesirable answer is returned. This issue can be overcome with the following trick. We first query the label of an example $x \in T \cap S$. If x is misclassified by w_t, we immediately make an update. Otherwise, every time we make a query (Z, y), we can instead query $(Z \cup \{x\}, y)$, which prevents us from querying an empty region and does not make a query more complicated.

So far, we have given an overview of why $\tilde{O}(d^3 \log(1/\alpha))$ queries suffice to correctly label $1-\alpha$ fraction of examples in S. Finally, it remains to bound the VC dimension of the query class we use. Recall that the modified perceptron algorithm we used is implemented over the space under the transform $f_A(\cdot)$. As we will discuss in Appendix C.3.2 since the target hypothesis is a halfspace, the labels of points are preserved by Forster's transform. So, every time we make a query (Z, y) in the modified perceptron algorithm, the actual query we should make is $(\{x \in V \mid f_A(x) \in Z\}, y)$. As we discussed above, Z is a set of $O(d)$ linear inequalities. So, the query class we use is defined by $O(d)$ degree-2 polynomial inequalities, which has VC dimension $\tilde{O}(d^3)$.

4. Conclusion and Future Directions

The fast development of machine learning has not only resulted in many real applications but has also changed the learning paradigm itself. The success of foundation models makes it easier and faster for the learner to get feedback for more complicated questions, turning the learning paradigm from passively learning from labeled data to actively learning from interactions. In this work, we initiate the study of active learning with region queries, a specific type of such interaction. We summarize our contribution and list several interesting future directions as open questions.

An important novelty of this work is using the VC dimension as a measure of the complexity of queries. As we show in the paper, when the learner and the expert share the dataset S, the VC dimension gives a good tradeoff between the complexity of the query class and the query complexity of the learning algorithms. *Can VC dimension be used to measure the complexity of other learning problems that involve interaction and communication such as distributed learning (Balcan et al., 2012; Kane et al., 2019)?* We think this would be an interesting direction to investigate.

To actively learn a hypothesis class $\mathcal{H}$ with $O(\log n)$ queries, a query class with VC dimension $O(d)$ is enough. On the other hand, we have also seen that for some hypothesis classes with good structure, we can learn it with a query class with VC dimension $O(\log d)$ or even $O(1)$. It is natural to ask *which hypothesis class can be learned with a query class with $o(d)$ VC dimension?* Studying the query complexity of active learning algorithms using a fixed query class would be also an interesting direction.

For several natural hypotheses classes, we design simple query classes and efficient learning algorithms. Surprisingly, these learning algorithms even work when the dataset is not shared between the learner and the labeler. *Does such a phenomenon hold for general hypothesis classes?* It is important to understand such a question since the assumption that the learner and the labeler share the knowledge of S does not always hold for some real applications.

Another important direction is learning with noisy queries. In this paper, we only study the realizable cases, assuming each query is answered correctly. *Can we design learning algorithms robust to wrong answers in their queries?*

Acknowledgments

This work was supported by the NSF Award CCF-2144298 (CAREER).

References

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

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. *Advances in neural information processing systems*, 29, 2016.

Pranjal Awasthi, Maria Florina Balcan, and Philip M Long. The power of localization for efficiently learning linear separators with noise. *Journal of the ACM (JACM)*, 63(6):1–27, 2017.

Maria Florina Balcan and Steve Hanneke. Robust interactive learning. In *Conference on Learning Theory*, pages 20–1. JMLR Workshop and Conference Proceedings, 2012.

Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316. PMLR, 2013.

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*, pages 26–1. JMLR Workshop and Conference Proceedings, 2012.

Maria-Florina F Balcan and Hongyang Zhang. Sample and computationally efficient learning algorithms under s-concave distributions. *Advances in Neural Information Processing Systems*, 30, 2017.

José L Balcázar, Jorge Castro, and David Guijarro. A general dimension for exact learning. In *Computational Learning Theory: 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001 Amsterdam, The Netherlands, July 16–19, 2001 Proceedings 14*, pages 354–367. Springer, 2001.

José L Balcázar, Jorge Castro, and David Guijarro. A new abstract combinatorial dimension for exact learning via queries. *Journal of Computer and System Sciences*, 64(1):2–21, 2002.

Calvin Beideman and Jeremiah Blocki. Set families with low pairwise intersection. *arXiv preprint arXiv:1404.4622*, 2014.

Omri Ben-Eliezer, Max Hopkins, Chutong Yang, and Hantao Yu. Active learning polynomial threshold functions. *Advances in Neural Information Processing Systems*, 35:24199–24212, 2022.

Avrim Blum, Alan Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22:35–52, 1998.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of clusters in finite metric spaces using oracle queries. In *Conference on Learning Theory*, pages 775–803. PMLR, 2021.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, Andrea Paudice, and Maximilian Thiessen. Active learning of classifiers with label and seed queries. *Advances in Neural Information Processing Systems*, 35:30911–30922, 2022.

Hunter Chase and James Freitag. Bounds in query learning. In *Conference on Learning Theory*, pages 1142–1160. PMLR, 2020.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17, 2004.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. *Advances in neural information processing systems*, 18, 2005.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005. Proceedings 18*, pages 249–263. Springer, 2005.

Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

Alberto Del Pia, Mingchen Ma, and Christos Tzamos. Clustering with queries under semi-random noise. In *Conference on Learning Theory*, pages 5278–5313. PMLR, 2022.

Ilias Diakonikolas, Daniel Kane, and Christos Tzamos. Forster decomposition and learning halfspaces with noise. *Advances in Neural Information Processing Systems*, 34:7732–7744, 2021.

Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Self-directed linear classification. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 2919–2947. PMLR, 2023a.

Ilias Diakonikolas, Christos Tzamos, and Daniel M Kane. A strongly polynomial algorithm for approximate forster transforms and its application to halfspace learning. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1741–1754, 2023b.

Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002.

Servane Gey. Vapnik–chervonenkis dimension of axis-parallel cuts. *Communications in Statistics-Theory and Methods*, 47(9):2291–2296, 2018.

S. A Goldman and R. H Sloan. The power of self-directed learning. *Machine Learning*, 14:271–294, 1994.

Steve Hanneke and Liu Yang. Minimax analysis of active learning. *J. Mach. Learn. Res.*, 16(1): 3487–3602, 2015.

Sariel Har-Peled, Mitchell Jones, and Saladi Rahul. Active-learning a convex body in low dimensions. *Algorithmica*, 83:1885–1917, 2021.

David Haussler, Nick Littlestone, and Manfred K Warmuth. Predicting $\{0, 1\}$-functions on randomly drawn points. *Information and Computation*, 115(2):248–292, 1994.

Max Hopkins, Daniel Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. *Advances in Neural Information Processing Systems*, 33:6342–6353, 2020a.

Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Noise-tolerant, reliable active classification with comparison queries. In *Conference on Learning Theory*, pages 1957–2006. PMLR, 2020b.

Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1034–1044. IEEE, 2020c.

Max Hopkins, Daniel Kane, Shachar Lovett, and Michal Moshkovitz. Bounded memory active learning through enriched queries. In *Conference on Learning Theory*, pages 2358–2387. PMLR, 2021.

Daniel Kane, Roi Livni, Shay Moran, and Amir Yehudayoff. On communication complexity of classification problems. In *Conference on Learning Theory*, pages 1903–1943. PMLR, 2019.

Daniel M Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 355–366. IEEE, 2017.

Daniel M Kane, Shachar Lovett, and Shay Moran. Generalized comparison trees for point-location problems. *arXiv preprint arXiv:1804.08237*, 2018.

Sanjeev R Kulkarni, Sanjoy K Mitter, and John N Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11:23–35, 1993.

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

Arya Mazumdar and Barna Saha. Clustering with noisy queries. *Advances in Neural Information Processing Systems*, 30, 2017.

Nimrod Megiddo. Partitioning with two lines in the plane. *Journal of Algorithms*, 6(3):430–433, 1985.

David Pincus. The dense linear ordering principle. *The Journal of Symbolic Logic*, 62(2):438–456, 1997.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

David R Shanks and Mark F St John. Characteristics of dissociable human learning systems. *Behavioral and brain sciences*, 17(3):367–395, 1994.

R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. doi: 10.1017/9781108231596.

Jinghui Xia and Zengfeng Huang. Optimal clustering with noisy queries via multi-armed bandit. In *International Conference on Machine Learning*, pages 24315–24331. PMLR, 2022.

Yichong Xu, Hongyang Zhang, Kyle Miller, Aarti Singh, and Artur Dubrawski. Noise-tolerant interactive learning using pairwise comparisons. *Advances in neural information processing systems*, 30, 2017.

Gal Yona, Shay Moran, Gal Elidan, and Amir Globerson. Active learning with label comparisons. In *Uncertainty in Artificial Intelligence*, pages 2289–2298. PMLR, 2022.

Appendix A. Notation and Preliminaries

Let $\mathcal{X}$ be an example space. A hypothesis class $\mathcal{H}$ is a set of binary functions $h : \mathcal{X} \to \{\pm 1\}$. A hidden true hypothesis $h^* \in \mathcal{H}$ assigns a positive or negative label $y(x) = h^*(x)$ to each $x \in \mathcal{X}$.

A region query is a pair (T, z), where $T \subseteq \mathcal{X}$ is a region in $\mathcal{X}$ and $z \in \{\pm 1\}$ is a proposed label. A region query family Q is a set of region queries. We will define $\mathrm{dom}(Q) := \{T \mid (T, z) \in Q, z \in \{\pm 1\}\}$ the set of regions used in a query in Q. The complexity of a query family is defined by the VC dimension of the set family that Q uses.

Definition 10 (VC Dimension of A Query Class) *Let $\mathcal{X}$ be a space of example and $C \subseteq 2^{\mathcal{X}}$ be a set family over $\mathcal{X}$. The VC dimension $VC\dim(C)$ of C is defined as the largest number d such that there exists a set S of d examples such that $|\{c \cap S \mid c \in C\}| = 2^d$. Let Q be a family of region query family Q over S. The VC dimension of Q is defined as*

$$VC\dim(Q) := VC\dim(\{T \mid (T, z) \in Q, z \in \{\pm 1\}\}) = VC\dim(\mathrm{dom}(Q)).$$

A learning process is a sequence of interactions between a learning algorithm $\mathcal{A}$ and a labeler. The learning algorithm $\mathcal{A}$ is given the hypothesis class $\mathcal{H}$, a dataset $S \subseteq X$ of n examples, and a region query family Q. The labeler is given a labeling domain L such that $S \subseteq L$. In a single round of interaction, the learning algorithm $\mathcal{A}$ submits a query (T, z) to the labeler based on any information $\mathcal{A}$ received so far. The labeler returns an answer $q(T, z) \in \{0, 1\}$ of the query to $\mathcal{A}$. Here, $q(T, z) = 1$ if $\forall x \in T \cap L$, $y(x) = z$. In particular, if $T \cap L = \emptyset$, an arbitrary answer can be returned by the labeler. At the end of the learning process, the learning algorithm outputs a set of labeled examples $O = \{(x, \hat{y}(x)) \mid x \in S' \subseteq S\}$. For $\alpha \in [0, 1)$, we say $\mathcal{A}$ labels $1 - \alpha$ fraction of S if $|O| \geq (1 - \alpha)n$ and for each $(x, \hat{y}(x)) \in O$, $\hat{y}(x) = y(x)$. In particular, if $\alpha = 0$, we say $\mathcal{A}$ perfectly labels S.

Some Facts on VC Dimension We list some properties of VC dimension that will be frequently used during the proof.

(i) Let C_1, C_2 be two set families over a space of examples $\mathcal{X}$ such that $VC\dim(C_1) = d_1$ and $VC\dim(C_2) = d_2$. Then $VC\dim(C_1 \cup C_2) \leq d_1 + d_2 + 1$.

(ii) Let C be a set family over a space of examples $\mathcal{X}$ such that $VC\dim(C) = d$. The k-fold unions of C and k-fold intersections of C is defined as

$$C^{k\cup} := \{\cup_{i=1}^{k} c_i \mid c_i \in C\}, C^{k\cap} := \{\cap_{i=1}^{k} c_i \mid c_i \in C\}.$$

Then $VC\dim(C^{k\cup}) \leq O(dk \log k)$ and $VC\dim(C^{k\cap}) \leq O(dk \log k)$.

Appendix B. Missing Details in Section 2

B.1. Proof of Theorem 2

In this section, we prove Theorem 2, which shows every hypothesis class with VC dimension d can be learned with a query class with VC dimension $O(d)$ with an information-theoretic optimal query complexity. To remind the reader, we restate Theorem 2 here.

Theorem 11 *(Restatement of Theorem 2) Let $\mathcal{X}$ be a space of example and $\mathcal{H}$ be a hypothesis class over $\mathcal{X}$ with VC dimension d. There is a region query family Q over $\mathcal{X}$ with VC dimension $O(d)$ and* *a learning algorithm $\mathcal{A}$ such that for any set of n examples $S \subseteq \mathcal{X}$ labeled by any true hypothesis $h^* \in \mathcal{H}$, $\mathcal{A}$ makes $O(d \log n)$ region queries from Q and correctly label every example in S, if the labeling domain $L = S$.*

To start with, we will remind the reader of some basic background in set theory.

Definition 12 (Strcit Total Order) *Let $\mathcal{X}$ be a non-empty set. A binary relation " $<$ " over $\mathcal{X}$ is a strict total order if for every $a, b, c \in \mathcal{X}$, the following conditions are satisfied.*

- *Not $a < a$. (irreflexive)*
- *If $a < b$, then not $b < a$.(asymmetric)*
- *If $a < b$, $b < c$, then $a < c$. (transitive)*
- *If $a \neq b$, then $a < b$ or $b < a$. (connected)*

Consider a set $\mathcal{X}$ with a strict total order " $<$ ", we have the following lemma.

Lemma 13 *Let $\mathcal{X}$ be a set and " $<$ " be a strict total order over X. Let $I = \{[a, b] \mid a, b \in X\}$, where $x \in [a, b]$ if $a \leq x \leq b$. $VC(I) \leq 2$.*

Proof (Proof of Lemma 13) Let a, b, c be any 3 distinct points in $\mathcal{X}$ such that $a < b < c$. Since " $<$ " is a strict total order, we know that 3 distinct points can be ordered in the above way. Let $h = [l, r] \in H$ be any set such that $a \in h$ and $c \in h$. By transitive property, we know that $l \leq a < b < c \leq r$, which implies that $b \in h$. Thus, no hypothesis in I can label a, c positive but b negative, which implies $VC(I) \leq 2$. ■

Lemma 13 implies that if a space of examples $\mathcal{X}$ admits a strict total order, then we are able to define the class of intervals over $\mathcal{X}$, which has a very small VC dimension. If $\mathcal{X}$ is finite, such a strict total order can be easily defined by any permutation of $\mathcal{X}$. If $\mathcal{X}$ is infinite or continuous, we next explain that such a strict total order(linear order) can also be defined. This fact follows the following well-known well-ordering theorem (equivalent to Zorn's lemma and axiom of choice).

Theorem 14 (well-ordering theorem) *A set $\mathcal{X}$ is well-ordered by some strict total order if every non-empty subset of $\mathcal{X}$ has a least element under the order. Furthermore, every set $\mathcal{X}$ can be well ordered.*

According to (Pincus, 1997), well-ordering theorem implies that every example space admits a strict total order.

With the background of the basic set theory, we are able to prove the following structural result.

Lemma 15 *Let $\mathcal{X}$ be a space of examples and " $<$ " be a strict total ordering over $\mathcal{X}$. Let $\mathcal{H}$ be any hypothesis class over $\mathcal{X}$. Let $S \subseteq \mathcal{X}$ be any subset of n examples. Define $H_S = \{h_S : S \to \{\pm 1\} \mid \exists h \in \mathcal{H}, h_S(x) = h(x), \forall x \in S\}$ be the hypothesis class of $\mathcal{H}$ restricted at set S. If $|H_S| > 1$, then there exists an interval $[a, b]$ and a hypothesis h such that $|\{h_S \in H_S \mid h_S(x) = h(x), \forall x \in [a, b] \cap S\}| \in [|H_S|/3, 2|H_S|/3]$.*

Proof (Proof of Lemma 15) We order examples in S according to the strict total order " $<$ " and denote by $x^{(1)} < x^{(2)} < \cdots < x^{(n)}$ these ordered examples. Given this ordered dataset S, we recursively define ith majority prediction class $M^{(i)}$ in the following way, $M^0 = H_S$,

$$M^{(i+1)} = \{h_S \in M^{(i)} \mid |M^{(i)} \cap \{h' \in H_S \mid h'(x^{(i+1)}) = h_S(x^{(i+1)})\}| \geq |M^{(i)}|/2\}.$$

That is to say, $M^{(i+1)}$ is the class of hypothesis in $M^{(i)}$ that predicts the label of $x^{(i+1)}$ according to the majority of $M^{(i)}$. Let $i^* \in [n]$ be the smallest number such that $|M^{(i^*)}| \leq 2|H_S|/3$. We notice that $|M^{(i^*)}| \geq |H_S|/3$ because

$$|M^{(i^*)}| \geq |M^{(i^*-1)}|/2 > |H_S|/3,$$

by the definition of the majority prediction class and i^*. Next, we show that such an i^* exists. Notice that $M^{(n)}$ contains a single hypothesis in H_S, thus, we have $1 = |M^{(n)}| \leq |H_S|/|H_S| \leq |H_S|/2 < 2|H_S|/3$. Furthermore, since $|M^{(0)}| = |H_S|$, we know that $i^* \in [n]$ exists. Now we set $a = x^{(1)}, b = x^{(i^*)}$ and $h \in H$ be any hypothesis such that $\exists h_S \in M^{(i^*)}$ agrees with h for every example in S. Then we have

$$|\{h_S \in H_S \mid h_S(x) = h(x), \forall x \in [a,b] \cap S\}| = |M^{(i^*)}| \in [|H_S|/3, 2|H_S|/3],$$

since $M^{(i^*)} = \{h_S \in H_S \mid h_S(x) = h(x), \forall x \in [a,b] \cap S\}$.

■

Given the above structural result, we present Algorithm 1, the algorithm we use in the proof of Theorem 2.

Algorithm 1 GENERALQUERY$(S, \mathcal{H}, Q)$ (Label S with query set Q given hypothesis class $\mathcal{H}$)

```
Let $H_S = \{h_S \mid \exists h \in \mathcal{H}, h_S(x) = h(x), \forall x \in S\}$.
while $|H_S| > 1$ do
    Find interval $[a,b] \in 2^{\mathcal{X}}$ and $\hat{h} \in \mathcal{H}$ that satisfies the property in the statement of Lemma 15.
    Let $S^+ = \{x \in [a,b] \mid \hat{h}(x) = 1\}$ and $S^- = \{x \in [a,b] \mid \hat{h}(x) = -1\}$
    Make query $(S^+, 1)$ and $(S^-, -1)$.
    if $q(S^+, 1) = q(S^-, -1) = 1$ then
        $\mathcal{H} \leftarrow \{h \in \mathcal{H} \mid h(x) = \hat{h}(x), \forall x \in S \cap [a,b]\}$.
    else
        $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h \in \mathcal{H} \mid h(x) = \hat{h}(x), \forall x \in S \cap [a,b]\}$.
    $H_S = \{h_S \mid \exists h \in \mathcal{H}, h_S(x) = h(x), \forall x \in S\}$
Label $S$ according to the single partial hypothesis in $H_S$.
```

Proof (Proof of Theorem 2) We show that Algorithm 1 uses a query class Q with VC dimension $O(d)$ that labels S correctly with $O(d \log n)$ queries.

We first show the correctness of the algorithm. Let h_S^* be the target hypothesis restricted at S. Every time we make queries $(S^+, 1), (S^-, -1)$ during the execution of Algorithm 1, h_S^* agrees with $\hat{h}$ at every example in $S \cap [a,b]$ if and only if $q(S^+, 1) = q(S^-, -1) = 1$, which implies that h_S^* is always contained in H_S. So, at the end of Algorithm 1, every example in S is labeled according to h_S^* and thus is labeled correctly.

Next, we bound the number of queries used by the algorithm. According to Lemma 15, we know that every time we find an interval $[a, b]$ and $\hat{h}$, we have

$$|\{h_S \in H_S \mid h_S(x) = \hat{h}(x), \forall x \in [a, b] \cap S\}| \in [|H_S|/3, 2|H_S|/3].$$

This implies

$$|H_S \setminus \{h_S \in H_S \mid h_S(x) = \hat{h}(x), \forall x \in [a, b] \cap S\}| \leq 2|H_S|/3.$$

So, whether h^*_S agrees with $\hat{h}$ over $S \cap [a, b]$ or not, after each update the size of H_S will always shrink by a factor of $2/3$. Since H has a VC dimension of d, we know from Sauer's lemma that $|H_S| \leq O(n^d)$ at the beginning of the execution of Algorithm 1. Thus, after $O(d \log n)$ updates $|H_S| = 1$ and Algorithm 1 will terminate. The total number of queries is $O(d \log n)$ since we make 2 queries for a single update.

Finally, we upper bound the VC dimension of the query class Q that Algorithm 1 uses. Notice that $Q = \{[a, b] \cap \{x \mid g(x) = 1\} \mid a, b \in X, g \in H \cup \bar{H}\}$, where $\bar{H} = \{-h(x) \mid h \in H\}$ is the set of complement of hypothesis in H. By the property of VC dimension, we have

$$VC(Q) \leq VC(I)VC(H \cup \bar{H}) \leq VC(I)(2VC(H) + 1) \leq 2(2VC(H) + 1) \leq 6d.$$

■

B.2. Proof of Theorem 3

In this section, we present the proof of Theorem 3, showing a matching lower bound for Theorem 2. Here, we restate Theorem 3 as a reminder.

Theorem 16 *(Restatement of Theorem 3) For every $d \in N^+$ and $n \geq d$ large enough, there exists a space of examples $\mathcal{X}$ and a hypothesis class $\mathcal{H}$ over $\mathcal{X}$ with VC dimension d such that there exists a set of n example S such that for every region query family Q over $\mathcal{X}$ with $VC\dim(Q) \leq (d-2)/3$ and every active learning algorithm $\mathcal{A}$, there exists a true hypothesis $h^* \in \mathcal{H}$, such that if $\mathcal{A}$ makes less than $\mathrm{poly}(n)$ region queries from Q, then with probability at least $1/3$, some example $x \in S$ is labeled incorrectly by $\mathcal{A}$. In particular, this even holds when $\mathcal{A}$ knows the labeling domain $L = S$.*

We start with Lemma 17, showing how to construct a hard instance for a fixed query family.

Lemma 17 *Let $\mathcal{X}$ be a space of examples and let Q be a region query class over $\mathcal{X}$. Let $C^* \subseteq \mathcal{X}$ be a set of k examples. Let $H_{C^*} = \{h_S \mid S \subseteq C^*, |S| \leq 1\}$ be a hypothesis class over $\mathcal{X}$, where $h_S(x) = 1$ if and only if $x \in C^* \setminus S$. Assuming for every $T \in \mathrm{dom}(Q)$, if $T \subseteq C^*$, then $|T| \leq t$. Then for every learner $\mathcal{A}$ that makes $k/2t$ queries from Q, there is some hypothesis $h^* \in H_{C^*}$ such that with probability at least $1/3$, there exists some $x \in C^*$ that is mislabeled by $\mathcal{A}$ assuming the labeling domain is the same as the example space i.e $L = \mathcal{X}$.*

Proof (Proof of Lemma 17) Let $x \in C^*$ be an example. We say x is covered by some query $T \in \mathrm{dom}(Q)$ if either $x \in T \subseteq C^*$ or $T \cap C^* = \{x\}$. Assume that the target hypothesis h^* is drawn uniformly from H_{C^*}. Denote by $Q_S \subseteq Q$ the random subset of queries that $\mathcal{A}$ makes and $\hat{h}_S$ the output hypothesis by $\mathcal{A}$ conditioned on the target hypothesis is $h^* = h_S$. Notice that if x is not covered by $\mathrm{dom}(Q_\emptyset)$, then we must have $\hat{h}_{\{x\}} = \hat{h}_\emptyset$. This is because no matter whether the target hypothesis is $h_\emptyset$ or $\hat{h}_{\{x\}}$, each query $\mathcal{A}$ made so far will have exactly the same answer. Specifically, let $(T, z) \in Q_\emptyset$ be any query used by $\mathcal{A}$ so far. We know that $x \notin T$. If $T \subseteq C^*$, then T only contains positive examples. If $T \subseteq \mathcal{X} \setminus C^*$, then T contains only negative examples. Otherwise, T contains at least one positive example and one negative example.

Now, assuming $\mathbf{Pr}(\hat{h}_\emptyset \neq h_\emptyset) \leq 1/3$, we will show there must be some $x \in C^*$ such that $\mathbf{Pr}(\hat{h}_{\{x\}} \neq h_{\{x\}}) > 1/3$. This will follow the standard way of bounding the probability of making an error used in the active learning literature such as (Hanneke and Yang, 2015).

$$\begin{aligned}
\max_{x \in C^*} \mathbf{Pr}(\hat{h}_{\{x\}} \neq h_{\{x\}}) &\geq \frac{1}{k} \sum_{x \in C^*} \mathbf{Pr}(\hat{h}_{\{x\}} \neq h_{\{x\}}) \geq \frac{1}{k} \sum_{x \in C^*} \mathbf{Pr}(\hat{h}_{\{x\}}(x) = h_\emptyset(x)) = \frac{1}{k} \mathbf{E} \sum_{x \in C^*} 1_{\{\hat{h}_{\{x\}}(x) = h_\emptyset(x)\}} \\
&\geq \frac{1}{k} \mathbf{E} \sum_{x \in C^*} 1_{\{x \text{ not covered by } Q_\emptyset\}} 1_{\{\hat{h}_{\{x\}}(x) = h_\emptyset(x)\}} = \frac{1}{k} \mathbf{E} \sum_{x \in C^*} 1_{\{x \text{ not covered by } Q_\emptyset\}} 1_{\{\hat{h}_\emptyset(x) = h_\emptyset(x)\}} \\
&\geq \frac{1}{k} \mathbf{E} \sum_{x \in C^*} 1_{\{x \text{ not covered by } Q_\emptyset\}} 1_{\{\hat{h}_\emptyset(x) = h_\emptyset(x)\}} \geq \frac{1}{k} \mathbf{E}\, 1_{\{\hat{h}_\emptyset = h_\emptyset\}} \sum_{x \in C^*} 1_{\{x \text{ not covered by } Q_\emptyset\}} \\
&\geq \frac{1}{k} \mathbf{Pr}(\hat{h}_\emptyset = h_\emptyset)(k - k/2) > 1/3.
\end{aligned}$$

So, we conclude that ant learner $\mathcal{A}$ that makes $k/2t$ queries from Q will with probability at least $1/3$ label at least one example in C^* incorrectly. ■

Next, we present Lemma 18, which gives a way to extend the hard instance we constructed in Lemma 17 for a single query class to multiple query classes.

Lemma 18 *Let $\mathcal{X}$ be any space of n examples and Q be a query class over $\mathcal{X}$. Let $\{C_1, \ldots, C_N\} \subseteq 2^{\mathcal{X}}$ be a collection of $N > |\mathrm{dom}(Q)|$ subsets over $\mathcal{X}$ such that for every $i, j \in [N]$, $i \neq j$, $|C_i \cap C_j| \leq t$. There is some $C^* \in \{C_1, \ldots, C_N\}$ such that for every $T \in \mathrm{dom}(Q)$ if $T \subseteq C^*$, $|T| \leq t$.*

Proof (Proof of Lemma 18) We say a query $T_i \in \mathrm{dom}(Q)$ witnesses a set $C_i \in \{C_1, \ldots, C_N\}$ if $T_i \subseteq C_i$ and $|T_i| > t$. Let $T \in \mathrm{dom}(Q)$ be any region such that $|T| > t$, we claim that T can witness at most one set C from $\{C_1, \ldots, C_N\}$. This is because if there exists $C_i, C_j \in \{C_1, \ldots, C_N\}$, $i \neq j$, that are witnessed by T, then $T \subseteq C_i \cap C_j$, which implies that $|T| \leq |C_i \cap C_j| \leq t$ and gives a contradiction. Since $N > |\mathrm{dom}(Q)|$, we know that there must be at least one $C^* \in \{C_1, \ldots, C_N\}$ that is not witnessed by any $T \in \mathrm{dom}(Q)$. Thus for every $T \in \mathrm{dom}(Q)$, if $T \subseteq C^*$ then we must have $|T| \leq t$. ■

Besides the above two technical lemmas, we will make use of the following results that construct set families with small pairwise intersections.

Theorem 19 (Theorem 3 in (Beideman and Blocki, 2014)) *For every positive integer $k \geq \gamma$, there exist $N \geq (2k \ln 2k)^{\gamma+1}$ subsets $S_1, \ldots, S_N \subseteq [4k^2 \ln 4k]$ such that for every $i \neq j \in [N]$, $|S_i| = |S_j| = k$ and $|S_i \cap S_j| \leq \gamma$.*

With the above technical lemmas, we are ready to present the proof of Theorem 3.

Proof (Proof of Theorem 3) Let $\mathcal{X}$ be a space of $n = 4k^2 \ln 4k$ examples. By Sauer's lemma, we know that any query family Q with VC dimension d must have

$$|\mathrm{dom}(Q)| \le \sum_{i=0}^{d} \binom{n}{i} \le cn^d < n^{d+1} = \left(4k^2 \ln 4k\right)^{d+1},$$

when n is larger than some suitably large constant c. By Theorem 19, there exists some integer $N \ge (2k \ln 2k)^{\gamma+1} > (4k^2 \ln 4k)^{d+1} > |\mathrm{dom}(Q)|$, such that we are able to find subsets $C_1, \dots, C_N \subseteq X$, where each subset has size k and any pair of these N sets has at most γ common examples. Notice that when k is larger than some suitably large constant, $\gamma = 3d$ is sufficient to make $(2k \ln 2k)^{\gamma+1} > (4k^2 \ln 4k)^{d+1}$.

Our next step is to use the set family $\{C_1, \dots, C_N\}$ to construct our hypothesis class $\mathcal{H}$. For $i \in [N]$, define $\mathcal{H}_{C_i} = \{h_S \mid S \subseteq C_i, |S| \le 1\}$, where $h_S(x) = 1$ if and only if $x \in C^* \setminus S$. The hypothesis class we use is $\mathcal{H} = \bigcup_{i \in [N]} \mathcal{H}_{C_i}$.

We start by showing $\mathcal{H}$ has VC dimension at most $\gamma + 2 \le 3d + 2$. Let $I = \{x_1, \dots, x_{\gamma+3}\}$ be $\gamma + 3$ different examples in $\mathcal{X}$. Assuming that $\mathcal{H}$ can shatter I, then we obtain that there exists some $h \in \mathcal{H}$ that labels every example in I positive. By construction, there must be some $i \in [N]$ such that $h \in \mathcal{H}_{C_i}$, which implies that $I \subseteq C_i$. However, we next show that there is no $h' \in \mathcal{H}$ can label $x_1, \dots, x_{\gamma+1}$ positive but $x_{\gamma+2}, x_{\gamma+3}$ negative. Assuming such an h' exists, then there must be some $j \in [N]$ such that $\{x_1, \dots, x_{\gamma+1}\} \subseteq C_j$. However, if $j \ne i$, then $|C_i \cap C_j| \le \gamma$. Thus, we must have $i = j$, which means $h' \in \mathcal{H}_{C_i}$. However, by construction each hypothesis in $\mathcal{H}_{C_i}$ can only label at most one example contained in C_i negative, which gives us a contradiction. So, we conclude the hypothesis class $\mathcal{H}$ we use has VC dimension at most $\gamma + 2 = 3d + 2$.

Next, we show that assuming the labeling domain, the dataset and the space of examples are the same. i.e. $L = S = \mathcal{X}$, for every query class Q with VC dimension at most d there exists a subset of k examples C^* such that every learner $\mathcal{A}$ that makes less than $k/(2\gamma)$ queries will with probability at least $1/3$ mislabel some example $x \in C^*$. Since $N > |\mathrm{dom}(Q)|$, by Lemma 18, we know that there exists some $C^* \in \{C_1, \dots, C_N\}$ such that for every query $(T, z) \in Q$, if $T \subseteq C^*$, then $|T| \le \gamma$. By Lemma 17, we know that if $\mathcal{A}$ only makes less than $k/(2\gamma)$ queries from Q then with probability at least $1/3$ some example $x \in C^*$ will be mislabeled by $\mathcal{A}$.

Thus, for every d and every k that is larger than some constant, we constructed a hypothesis class $\mathcal{H}$ with VC dimension at most $3d + 2$ over an example space $\mathcal{X}$ with size $\tilde{O}(k^2)$, which is also the dataset S to be labeled, such that for every learner $\mathcal{A}$ and query class Q with VC dimension at most d, if $\mathcal{A}$ makes less than $k/2d$ queries than there is a true hypothesis $h^* \in \mathcal{H}$ such that with probability at least $1/3$, $\mathcal{A}$ will misclassify at least one of the examples, even assuming the labeling domain $L = S$. ■

We remark that the construction of the hard instance in Theorem 3 is fully combinatorial. So, given any large enough space of examples $\mathcal{X}$, we can embed the hard instance we constructed into $\mathcal{X}$ to get a hard instance in that example space.

B.3. Proof of Corollary 5

Corollary 20 (Restatement of Corollary 5) *There is a space of examples $\mathcal{X}$ such that for every $d \in N^+$ and $n \ge d$ large enough, there exists a hypothesis class $\mathcal{H}$ over $\mathcal{X}$ with VC dimension d such that there exists a set of n example S such that for every region query family Q over $\mathcal{X}$ with*

$VC\dim(Q) \leq (d-3)/3$ and every active learning algorithm $\mathcal{A}$, there exists a true hypothesis $h^* \in \mathcal{H}$, such that if $\mathcal{A}$ makes less than $\text{poly}(n)$ region queries from Q, then with probability at least $1/3$, some example $x \in S$ is labeled incorrectly by $\mathcal{A}$. In particular, this even holds when $\mathcal{A}$ knows the labeling domain $L = S$.

Proof (Proof of Corollary 5) For each m, let $\mathcal{X}_m = \{x_i^{(m)}\}_{i=1}^m$ be the space of examples constructed in Theorem 3 with parameter m. Let $\mathcal{X} = \cup_m \mathcal{X}_m$ be a space of examples. Since the constructions of $\mathcal{X}_m$ are fully combinatorial, we can assume for each $m_1, m_2 \in N^+$, $\mathcal{X}_{m_1} \cap \mathcal{X}_{m_2} = \emptyset$.

Let $d \in N^+$ and let H_m be the hypothesis class over $\mathcal{X}_m$ with VC dimension d constructed in Theorem 3. For each m and for each $f \in H_m$, we extend f to $\mathcal{X}$ in the following way. For every $x \in \mathcal{X} \setminus \mathcal{X}_m$, $f(x) = -1$. H_m still has VC dimension d over $\mathcal{X}$ under the extension. Furthermore, since each $\mathcal{X}_m$ is disjoint, $H = \cup_m H_m$ has VC dimension at most $d+1$. For each $n > d$ larger enough, let $S_n = \mathcal{X}_n \subseteq \mathcal{X}$ be a subset of n examples. By Theorem 3, we know that for every learning algorithm $\mathcal{A}$ and for every query class Q with VC dimension at most $(d-2)/3$ there exists a hypothesis $h^* \in H_n \subseteq \mathcal{H}$ such that $\mathcal{A}$ must make $\text{poly}(n)$ queries from Q to perfectly label S_n with probability more than $2/3$, even if $\mathcal{A}$ knows that a query will be checked based on S_n. ■

Appendix C. Missing Details in Section 3

In this section, we design efficient learning algorithms for several concrete hypothesis classes including the class of union of k intervals, the class of high dimensional boxes, and the class of high dimensional halfspaces, giving missing details in Section 3.

C.1. Proof of Theorem 6

In this section, we prove Theorem 6 by designing an efficient learning algorithm for the class of the union of k intervals. We restate Theorem 6 as follows.

Theorem 21 *(Restatement of Theorem 6) Let $\mathcal{X} = \mathbb{R}$ be the space of examples and $\mathcal{H} = \{h \mid \exists [a_i, b_i], i \in [k], s.t. h(x) = 1 \iff x \in \cup_{i=1}^k [a_i, b_i]\}$ be the class of union of k intervals over $\mathbb{R}$. Let I be the class of intervals over $\mathbb{R}$ and query family $Q = \{(T, z) \mid T \in I, z \in \{\pm 1\}\}$. There is a learner $\mathcal{A}$ such that for every subset of n examples S, labeled by any $h^* \in \mathcal{H}$ and for every labeling domain $S \subseteq L$(possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T+n)k \log n)$ time, makes $O(k \log n)$ queries from Q and labels every example in S correctly, where T is the running time to implement a single region query.*

We start with Algorithm 2, a sub-routine used to label examples in the left-most interval of the target hypothesis. The guarantee of Algorithm 2 is presented in Lemma 22.

Lemma 22 *Let $S = (x^{(1)}, \dots, x^{(m)}) \subseteq \mathbb{R}$ be a subset of n examples labeled by a union of k intervals $h^* = \cup_{i=1}^k [a_i, b_i]$. Let $L \subseteq \mathbb{R}$ be any arbitrary labeling domain such that $S \subseteq Y$. $\textsc{FindLeft}(S)$ makes $O(\log m)$ interval queries and returns the smallest index i^* such that there is some $y \in \{\pm 1\}$ such that $q([x^{(1)}, x^{(i^*)}], y) = 1$ and for every $y \in \{\pm 1\}$, $q([x^{(1)}, x^{(i^*+1)}], y) = 0$.*

Proof (Proof of Lemma 22) We first notice that if $q([x^{(1)}, x^{(i)}], y) = 0$, $\forall y \in \{\pm 1\}$, then $\forall j > i$, we also have $q([x^{(1)}, x^{(j)}], y) = 0$, $\forall y \in \{\pm 1\}$. This is because for any labeling domain L,

Algorithm 2 FindLeft(S) (Find the smallest i^* such that $q([x^{(1)}, x^{(i^*)}], y) = 1$) for some $y \in \{\pm 1\}$

```
Order $S$ as $x^{(1)} < \cdots < x^{(m)}$, where $m = |S|$.
if $q([x^{(1)}, x^{(m)}], y) = 1$ for some $y \in \{\pm 1\}$ then
    return $m$
Let $C = \{x^{(1)}, \ldots, x^{(m)}\}$                ▷ Candidates of the boundary points
while $|C| > 1$ do                                     ▷ Find the boundary via binary search
    Let $x'$ be the median of $C$.                     ▷ If $|C|$ is even, select $x'$ as the larger one
    if $q([x^{(1)}, x'], y) = 0, \forall y \in \{\pm 1\}$ then
        Remove $x'$ and all points greater than $x'$ from $C$
    else
        Remove all points less than $x'$ from $C$
return the index of the single element in $C$
```

$[x^{(1)}, x^{(i)}] \cap L \subseteq [x^{(1)}, x^{(j)}] \cap L$. Thus, if $[x^{(1)}, x^{(i)}] \cap L$ contains examples with both positive examples and negative examples then so does $[x^{(1)}, x^{(j)}] \cap L$. This implies that if some x' such that $q([x^{(1)}, x'], y) = 1$, for some $y \in \{\pm 1\}$, is removed from C, then we must have found some $x'' > x'$ such that $q([x^{(1)}, x'], y) = 1$, for some $y \in \{\pm 1\}$. In particular, this implies that $x^{(i^*)}$, where i^* is the index that satisfies the statement, will never be removed from C. This proves the correctness of Algorithm 2. It remains to prove the query complexity of Algorithm 2. In each iteration of Algorithm 2, we only make at most 2 region queries and remove half of the remaining points in C. This implies that Algorithm 2 will run at most $O(\log(|C|))$ iterations and the query complexity is $O(\log m)$. ■

Given Algorithm 2 and Lemma 22, we are now ready to present Algorithm 3, the learning algorithm and the proof of Theorem 6.

Algorithm 3 Label k-Interval$(S, \mathcal{H})$ (Label S with interval queries given hypothesis class $\mathcal{H}$)

```
while $|S| > 0$ do
    Order $S$ as $x^{(1)} < \cdots < x^{(m)}$, where $m = |S|$.
    $i^* \leftarrow$ FindLeft$(S)$
    if $q([x^{(1)}, x^{(i^*)}], 1) = 1$ then
        Label $x^{(1)}, \ldots, x^{(i^*)}$ by 1
    else
        Label $x^{(1)}, \ldots, x^{(i^*)}$ by -1
    $S \leftarrow S \setminus \{x^{(1)}, \ldots, x^{(i^*)}\}$.
```

Proof (Proof of Theorem 6) We first show the correctness of Algorithm 3. By Lemma 22, we know that every time Algorithm 3 calls Algorithm 2, we will find some i^* such that for some $y \in \{\pm 1\}$, $q([x^{(1)}, x^{(i^*)}], y) = 1$, which implies that the true labels of $x^{(1)}, \ldots, x^{(i^*)}$ are y. Thus, Algorithm 3 labels every example correctly.

Next, we bound the query complexity of Algorithm 3. By Lemma 22, we know that each time we call Algorithm 2, the example $x^{(i^*)}$ satisfies the following property. There is some $y \in \{\pm 1\}$ such that $q([x^{(1)}, x^{(i^*)}], y) = 1$ but $q([x^{(1)}, x^{(i^*+1)}], y) = 0$. Since the target hypothesis h^* is a union of k intervals, over any dataset S, there are at most $2k$ such pair of $x^{(i^*)}$ and $x^{(i^*+1)}$. Each call of Algorithm 2 finds one of such pair. Thus Algorithm 3 calls Algorithm 2 at most $2k$ times. By Lemma 22, each call of Algorithm 2 will make $O(\log n)$ queries. Thus, the query complexity of Algorithm 3 is $O(k \log n)$.

Furthermore, we notice that the running time of Algorithm 2 is $O((T+n)\log n)$, since each time we do a binary search, make 2 region queries and remove examples from the candidate set C, which takes $O(T+n)$ time. Thus the running time of Algorithm 3 is $O((T+n)k\log n)$. ■

C.2. Proof of Theorem 7

We present the proof of Theorem 7, restated as follows.

Theorem 23 *(Restatement of Theorem 7) Let $\mathcal{X} = \mathbb{R}^d$ be the space of examples and $\mathcal{H} = \{\prod_{i=1}^d [a_i, b_i] \mid a_i, b_i \in [-\infty, \infty]\}$ be the class of axis-parallel boxes in $\mathbb{R}^d$ that labels $\mathcal{X}$. There is a query class Q over $\mathbb{R}^d$ with VC dimension $O(\log d)$ and an efficient algorithm $\mathcal{A}$ such that for every set of n examples $S \subseteq \mathbb{R}^d$, every target hypothesis $h^* \in \mathcal{H}$, and for every labeling domain $S \subseteq L$(possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T+n)d\log n)$ time, makes $O(d \log n)$ queries from Q and labels every example in S correctly, where T is the running time to implement a single region query.*

Similar to what we did in Appendix C.1, we start with Algorithm 4, a subroutine we use to approximately learn a boundary of the target hypothesis. The guarantee of Algorithm 4 is presented in Lemma 24.

Algorithm 4 FINDBOUNDARY(S, w) (Find the boundary point in S along direction w)

```
Order $S$ as $x^{(1)}, \dots, x^{(m)}$, where $m = |S|$, such that $w \cdot x^{(1)} \le \cdots \le w \cdot x^{(m)}$.
if $q(\{x \mid w \cdot x \ge w \cdot x^{(1)}\}, 0) = 1$ then
    return $-\infty$
Let $C = \{w \cdot x^{(1)}, \dots, w \cdot x^{(m)}\}$                ▷ Candidates of boundary points
while |C|>1 do                                  ▷ Find the boundary point via binary search
    Let $b$ be the median of $C$.               ▷ If $|C|$ is even, select $b$ as the larger one
    if $q(\{x \mid w \cdot x \ge b\}, -1) = 1$ then
        Remove $b$ and all elements greater than $b$ from $C$
    else
        Remove all elements less than $b$ from $C$
return the single element in $C$
```

To prove Theorem 7, we first prove the following technical lemma.

Lemma 24 *Let $S \subseteq \mathbb{R}^d$ be a subset of n examples labeled by an axis-parallel box $h^* = \prod_{i=1}^d [a_i, b_i]$. Let $L \subseteq \mathbb{R}^d$ be any arbitrary labeling domain such that $S \subseteq L$. For every $i \in [d]$,* FINDBOUNDARY(S, e_i) *returns $\hat{b}_i \le b_i$ by making $O(\log n)$ queris such that for every $x \in S$ with $x_i > \hat{b}_i$, x is labeled by -1. Symmetrically,* FINDBOUNDARY$(S, -e_i)$ *returns $\hat{a}_i \ge a_i$ by making $O(\log n)$ queris such that for every $x \in S$ with $x_i < \hat{a}_i$, x is labeled by -1.*

Proof (Proof of Lemma 24) We prove the case for $\textsc{FindBoundary}(S, e_i)$ and the case for $\textsc{FindBoundary}(S, -e_i)$ can be proved symmetrically. We first prove the correctness of the algorithm. If Algorithm 4 terminates in the first round ($q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(1)}\}, -1) = 1$), then clearly $-\infty = \hat{b}_i \leq b_i$. Furthermore, since $S \subseteq \{x \mid e_i \cdot x \geq e_i \cdot x^{(1)}\} \cap L$, we know that every example in S is labeled by -1. In this case, the statement of Lemma 24 is true. In the rest of the proof, we assume Algorithm 4 does not terminate in the first round. We observe that for every $1 \leq i < j \leq m$, we have $\{x \mid e_i \cdot x \geq e_i \cdot x^{(j)}\} \subseteq \{x \mid e_i \cdot x \geq e_i \cdot x^{(i)}\}$. This implies that there exists a largest index $j^* \in [m]$ such that $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(j^*)}\}, -1) = 0$. In particular, $x_i^{(j^*)} \leq b_i$, because otherwise, any example $x \in \mathbb{R}^d$ with $x_i \geq x_i^{(j^*)}$ will be labeled by -1, which gives a contradiction to the answer to $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(j^*)}\}, -1)$. So, it is sufficient to show that the output $\hat{b}_i$ of Algorithm 4 is $x_i^{(j^*)}$.

Assuming we receive a feedback $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(k)}\}, -1) = 1$ for some $x^{(k)} \in S$, then no $x_i^{(j)}$ with $j < k$ is removed from C. In particular, no $x_i^{(j)}$ with $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(j)}\}, -1) = 0$ is removed from C. This implies that the final element remained in C must be some $x_i^{(j)}$ with $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(j)}\}, -1) = 0$. On the other hand, suppose we are removing some $x_i^{(k)}$ with $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(k)}\}, -1) = 0$. This implies we received a feedback of the form $q(\{x \mid e_i \cdot x \geq e_i \cdot x^{(k')}\}, -1) = 0$ for some $k' > k$. Thus, any $x_i^{(j)}$ with $j < k$ is either removed together with $x_i^{(k)}$ or has already been removed from C. This implies that the single element remaining in C is $x_i^{(j^*)}$, which is the output.

Finally, it remains to show the query complexity of Algorithm 4 is $O(\log n)$. Since b is selected as as the median of C, after every query, we remove half elements from C. So after making at most $O(\log n)$ queries, there is a single element remained in C and is output by Algorithm 4. ■

Given Algorithm 4 and Lemma 24, we are ready to present the learning algorithm, Algorithm 5 and the proof Theorem 7.

Algorithm 5 $\textsc{LabelBox}(S, \mathcal{H})$ (Label S with halfspace query given hypothesis class $\mathcal{H}$)

```
for $i \in [d]$ do
    $x_r^i \leftarrow \textsc{FindBoundary}(S, e_i)$
    $x_l^i \leftarrow \textsc{FindBoundary}(S, -e_i)$
Label all examples in $S \cap \prod_{i=1}^d [x_l^i, x_r^i]$ to be 1 and the others to be $-1$.
```

Proof (Proof of Theorem 7) We first prove the correctness of Algorithm 5. Let $B = \prod_{i=1}^d [a_i, b_i]$ be the target box that labels S. By the first part of Lemma 24, we know that the estimator $\hat{B} = \prod_{i=1}^d [x_l^i, x_r^i]$ of Algorithm 5 is a subset of B. Thus, every negative example in S is also labeled negative by Algorithm 5. Furthermore, by the second part of Lemma 24, we know that any example $x \in S \setminus \hat{B}$ has a true label -1. Thus, no positive example in S is labeled incorrectly.

Next, we upper bound the query complexity of Algorithm 5. By Lemma 24, we know that every time we call Algorithm 4, we make $O(\log n)$ queries. So the query complexity of Algorithm 5 is $O(d \log n)$. Furthermore, since every time we call Algorithm 4 in Algorithm 5, we just do a binary search. The running time of Algorithm 5 is $O((T + n)d \log n)$.

Finally, we show the query class Q has a small VC dimension. Notice that each query in Q corresponds to some linear classifier $\{x \mid w \cdot x \geq b\}$, where w is parallel to some e_i for $i \in [d]$. According to (Gey, 2018), we know that Q has VC dimension $O(\log d)$. ■

C.3. Learning Arbitrary High-Dimensional Halfspaces

In this section, we move to our main algorithmic result for learning halfspace. Since in this work, we want to label an arbitrary dataset $S \subseteq \mathbb{R}^d$, we can without loss of generality to assume that the target halfspace is homogeneous w^*.

C.3.1. EFFICIENT HALFSPACE LEARNING WITH ARBITRARILY COMPLICATED QUERY FAMILY

As we discussed in Section 3.3, we will first give an efficient halfspace learning algorithm using an arbitrarily complicated query class using the connection between active learning with region queries and self-directed learning. To start with, we remind the readers of the model of self-directed learning.

Definition 25 (Self-Directed Learning(Goldman and Sloan, 1994)) *Let $\mathcal{X}$ be a space of examples and let $\mathcal{H}$ be a class of hypothesis over $\mathcal{X}$. Let $h^* \in \mathcal{H}$ be an unknown target hypothesis let $S = \{x^{(1)}, \dots, x^{(n)}\} \subseteq \mathcal{X}$ be a subset of $n \in \mathbb{N}$ examples. The learner has access to the full set of (unlabeled) points $\mathcal{X}$.*
Until the labels of all examples of S have been predicted:

- *The learner $\mathcal{A}$ picks a point $x \in S$ and makes a prediction $\hat{y} \in \{0, 1\}$ about its label.*
- *The true label $h^*(x)$ of x is revealed and the learner makes a mistake if $\hat{y} \neq h^*(x)$*

The mistake bound $M(\mathcal{A}, S, h^*)$ is the total number of mistakes that $\mathcal{A}$ makes during the learning process.

Theorem 26 *Let $\mathcal{X}$ be a space of example and $\mathcal{H}$ be a class of hypotheses over $\mathcal{X}$. Let $S \subseteq X$ be a subset of n examples and let $h^* \in \mathcal{H}$ be the target hypothesis. Let $Q = \{(T, z) \mid z \in \{\pm 1\}, T \subseteq 2^{\mathcal{X}}\}$ over $\mathcal{X}$ be the query class that contains any subset of $\mathcal{X}$. If there is a self-directed learner $\mathcal{A}$ with mistake bound $M = M(\mathcal{A}, S, h^*)$ that labels S, then there is an active learner $\mathcal{A}'$ that makes $O(M \log n)$ queries from Q and labels S correctly in time $M(nT_{\mathcal{A}} + T_Q \log n)$, where T_A is the running time of $\mathcal{A}$ to predict a single example and T_Q is the running time to implement a single query.*

Proof (Proof of Theorem 26) We construct $\mathcal{A}'$ as follows. In a single round, if there is still an example $x \in S$, for which we don't know the true label, we run the self-directed learner $\mathcal{A}$ over S from the beginning to predict every example in S. If $\mathcal{A}$ makes a prediction for some x, whose label is already known, we provide the true label for $\mathcal{A}$ as feedback, otherwise, we provide the prediction of $\mathcal{A}$ as feedback assuming the prediction of $\mathcal{A}$ is correct. Denote by $\{(x, \tilde{y}(x))\}_{x \in S}$ the feedback that $\mathcal{A}$ receives in this execution. Denote by $S^+ := \{x \in S \mid \tilde{y}(x) = 1\}$ and $S^- := \{x \in S \mid \tilde{y}(x) = -1\}$. We make two queries $(S^+, 1)$ and $(S^-, -1)$. If $q(S^+, 1) = q(S^-, -1) = 1$, we label every $x \in S$ according to $\tilde{y}(x)$. Otherwise, we do a binary search over S^+ and S^- to find the first example x' where $\mathcal{A}$ makes a mistake in this execution. Then we know the true label of every example up to x', and we enter the next round. Clearly, when $\mathcal{A}'$ terminates, we label every example in S correctly.

Next, we upper bound the query complexity of $\mathcal{A}'$. We notice that in every round of execution of $\mathcal{A}'$, before the self-directed learner $\mathcal{A}$ predicts some example x whose label we don't know and $\mathcal{A}$ actually makes a mistake at x, the performance of $\mathcal{A}$ in this setting is the same as the performance of $\mathcal{A}$ who receives the true feedback. When $\mathcal{A}$ actually makes a mistake at x, we use $O(\log n)$ region queries to do a binary search and find the first misclassified examples x whose label we don't actually know. This implies in the next round $\mathcal{A}$ will be fed with the true feedback at example x and $\mathcal{A}$ will keep performing well until it makes the next mistake at some example we don't know the true label. Since the mistake bound of $\mathcal{A}$ is M, we know that $\mathcal{A}'$ will have at most M rounds and thus the total query complexity is $O(M \log n)$.

Finally, we analyze the running time of $\mathcal{A}'$. As we analyzed in the last paragraph, we know $\mathcal{A}'$ in total have at most M rounds, in each round, we make n predictions and make $O(\log n)$ queries. Thus, the running time of $\mathcal{A}'$ is $O(M(nT_{\mathcal{A}} + T_Q \log n))$. ■

Given Theorem 26, to prove Theorem 8, it is sufficient to design an efficient self-directed halfspace learning algorithm that makes $O(d \log n)$ mistakes for every S any every target halfspace w^*. Such an algorithm is easy to design using the idea from Haussler et al. (1994).

Theorem 27 *Let $\mathcal{X} = \mathbb{R}^d$ be the space of the examples and let $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. There is an efficient self-directed learning algorithm $\mathcal{A}$ such that for every subset $S \subseteq \mathcal{X}$ of n examples and for every target hypothesis $w^* \in \mathcal{H}$, $\mathcal{A}$ predicts each example in time $\mathrm{poly}(B)$, where B is the bit complexity of S and makes $O(d \log n)$ mistakes in expectation*

Proof (Proof of Theorem 27) We first describe the self-directed learning algorithm. The algorithm randomly order S and obtain a sequence of example $x^{(1)}, \dots, x^{(n)}$. To predict the label of example $x^{(i+1)}$, it computes $w^{(i)}$, a solution of the support vector machine (SVM) of $(x^{(1)}, y^{(1)}), \dots, (x^{(i)}, y^{(i)})$ and predicts $\hat{y}^{(i+1)} = \mathrm{sign}(w^{(i)} \cdot x^{(i+1)})$.

Now denote by $w^{(i+1)}$ the solution of SVM of $(x^{(1)}, y^{(1)}), \dots, (x^{(i+1)}, y^{(i+1)})$. Notice that $w^{(i+1)}$ is uniquely determined by the d support vectors. Since we make a random permutation of S, the probability that $x^{(i+1)}$ is one of the support vector is at most $d/(i+1)$, which implies that with probability at most $d/(i+1)$, $w^{(i)} \neq w^{(i+1)}$. Thus, the probability that we make a mistake at $x^{(i+1)}$ is at most $d/(i+1)$. Denote by M, the total number of mistakes made by $\mathcal{A}$. We have

$$\mathbf{E}\, M = \sum_{i=1}^{n} \mathbf{E}\, \mathbb{1}(x_i \text{ is misclassified}) = \sum_{i=1}^{n} \mathbf{Pr}\, \mathbb{1}(x_i \text{ is misclassified}) \leq \sum_{i=1}^{n} \frac{d}{i} \leq O(d \log n).$$

This shows in expectation the mistake bound of $\mathcal{A}$ is $O(d \log n)$. Furthermore, every time $\mathcal{A}$ makes a prediction, it solves a convex program based on S, and thus the running time is $\mathrm{poly}(B)$. ■

With Theorem 26 and Theorem 27, we give the following active learning algorithm and the proof of Theorem 8.

Theorem 28 *(Restatement of Theorem 8) Let $\mathcal{X} = \mathbb{R}^d$ be the space of examples and $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. Let $Q = \{(T, z) \mid z \in \{\pm 1\}, T \subseteq 2^{\mathbb{R}^d}\}$ over $\mathbb{R}^d$ be the query class that contains any subset of $\mathbb{R}^d$. There is an efficient algorithm $\mathcal{A}$ such that for every set of n examples S, labeled by any $w^* \in \mathcal{H}$ and for every labeling domain $S \subseteq L$ (possibly unknown to $\mathcal{A}$), $\mathcal{A}$ runs in $O((T+B)d \log^2 n)$ time, makes $O(d \log^2 n)$ queries from Q in expectation and labels every example in S correctly, where T is the running time to implement a single query and B is the bit complexity of S.*

Proof (Proof of Theorem 8) The proof of Theorem 8 follows directly by Theorem 26 and Theorem 27. The algorithm we use is Algorithm 6, which converts the self-directed learning algorithm used in the proof of Theorem 27 to an active learner using the proof of Theorem 26. ■

Algorithm 6 RANDOMIZEDSVM(S) (Label S with arbitrary region query)

```
Randomly order dataset $S$ and obtain sequence of examples $x^{(1)}, \dots, x^{(n)}$
$i^* \leftarrow 0$
while $i^* < n$ do
    Let $\hat{w}$ be a solution of the SVM over labeled data $(x^{(1)}, y^{(1)}), \dots (x^{(i^*)}, y^{(i^*)})$
    for $i \in [n]$ do
        $\hat{y}^{(i)} \leftarrow \mathrm{sign}(\hat{w} \cdot x^{(i)})$
        if $i > i^*$ then
            Update $\hat{w}$ to be a solution of the SVM over labeled data $(x^{(1)}, \hat{y}^{(1)}), \dots (x^{(i)}, \hat{y}^{(i)})$
    Let $S^+ : \{x^{(i)} \mid \hat{y}^{(i)} = 1\}$ and $S^- : \{x^{(i)} \mid \hat{y}^{(i)} = -1\}$
    Make queries $(S^+, 1)$ and $(S^-, -1)$.
    if $q(S^+, 1) = q(S^-, -1) = 1$ then
        Label every $x^{(i)}$ by $\hat{y}^{(i)}$ and return
    else
        Binary search over $S^+$ and $S^-$ to find the smallest $j$ such that $\hat{y}^{(j)} \neq y^{(j)}$ via region queries.
    $i^* \leftarrow j$, $\hat{y}^{(i^*)} \leftarrow 1 - \hat{y}^{(i^*)}$
    Label every $x^{(i)}$ by $\hat{y}^{(i)}$
Label all examples in $S \cap \prod_{i=1}^{d} [x_l^i, x_r^i]$ to be $1$ and the others to be $-1$.
```

C.3.2. EFFICIENT HALFSPACE LEARNING WITH SIMPLE QUERY FAMILY

In this section, we design an efficient halfspace learning algorithm with low query complexity using a query class with $\mathrm{poly}(d)$-VC dimension and prove Theorem 9.

Theorem 29 *(Restatement of Theorem 9) Let $\mathcal{X} = \mathbb{R}^d$ be a space of examples and let $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. There is an algorithm and a query class Q with VC dimension $\tilde{O}(d^3)$ such that for every subset of n examples $S \subseteq \mathcal{X}$, every labeling domain L with $S \subseteq L$ and every target hypothesis $w^* \in \mathcal{H}$ and every $\alpha \in (0, 1)$, it in expectation makes $O(d^3 \log^2 d \log(1/\alpha))$ queries from Q, runs in $\mathrm{poly}(d, n, T)$ time and labels $(1 - \alpha)$ fraction of examples in S correctly, where T is the running time of implement a single region query from Q. In particular, the algorithm makes $O(d^3 \log^2 d \log n)$ queries from Q and labels every example in S correctly in time $\mathrm{poly}(d, n, T)$.*

As mentioned in Section 3.3, we will make use of Forster's transform to make our dataset well-behaved. So, we will start with some background on Forster's transform before diving into the proof. We first introduce the notion of Approximate Radially Isotropic Position.

Definition 30 (Approximate Radially Isotropic Position) *Let S be a multiset of non-zero points in $\mathbb{R}^d$, we say S is in ϵ-approximate radially isotropic position, if for every $x \in S$, $\|x\| = 1$ and for every $u \in S^{d-1}$, $\sum_{x \in S} (u \cdot x)^2 / |S| \geq 1/d - \epsilon$.*

A simple calculation gives the following useful result, which has appeared in (Diakonikolas et al., 2023b,a), for a dataset that is in an approximate radially isotropic position.

Lemma 31 *Let S be a multiset of non-zero points in $\mathbb{R}^d$ that is in $1/2d$-approximate radially isotropic position. Then for every $u \in S^{d-1}$, we have $\mathbf{Pr}_{x\sim S}\left(|u \cdot x| \geq 1/2\sqrt{d}\right) \geq 1/4d$.*

In particular, several works have been done to show an approximate Forster's transform can be computed efficiently.

Theorem 32 (Approximate Forster's Transform (Diakonikolas et al., 2023b)) *There is an algorithm such that given any set of n points $S \subseteq \mathbb{R}^d \setminus \{0\}$ and $\epsilon > 0$, it runs in time $\mathrm{poly}(d, n, \log 1/\epsilon)$ and returns a subspace V of $\mathbb{R}^d$ containing at least $\dim(V)/d$ fraction of points in S and an invertible matrix $A \in \mathbb{R}^{d\times d}$ such that $f_A(S \cap V)$ is in ϵ-approximate radially isotropic position up to isomorphic to $\mathbb{R}^{\dim(V)}$, where $f_A(S \cap V) = \{f_A(x) := Ax/\left\|Ax\right\| \mid x \in S \cap V\}$.*

Combine Theorem 32 and Lemma 31, we know that given any set of n examples $S \subseteq \mathbb{R}^d$, we can find a subset of at least kn/d examples $S_V := S \cap V \subseteq S$ that lies in some k-dimensional subspace V and some invertible matrix A such that $f_A(S_V)$ is in $1/2k$-approximate radially isotropic position (up to isomorphic to $\mathbb{R}^k$). Now, for convenience, we assume our transformed data $f_A(S_V)$ is exactly our original dataset and we focus on the transformed data. Notice that for each $x \in S_V$, we have

$$\mathrm{sign}(w^* \cdot x) = \mathrm{sign}(A^{-T}w^* \cdot Ax) = \mathrm{sign}(A^{-T}w^* \cdot f_A(x)) = \mathrm{sign}(\mathrm{proj}_{A(V)}(A^{-T}w^*) \cdot f_A(x)),$$

which implies that each transformed example $f_A(x)$ is labeled by halfspace w_V^* and has the same label as x, where v^* is the unit vector parallel to $\mathrm{proj}_{A(V)}(A^{-T}w^*)$. (We can without loss of generality assume that $\mathrm{proj}_{A(V)}(A^{-T}w^*) \neq 0$, otherwise we only need to use a single query to check if examples in V are all labeled positive.) Given the above discussion, we design Algorithm 7, a learning algorithm that correctly labels a large fraction of the dataset S, if S is in approximate radially isotropic position. Formally, we prove Theorem 33.

Theorem 33 *Let $\mathcal{X} = \mathbb{R}^d$ be a space of examples and let $\mathcal{H} = \{w \mid w \in S^{d-1}\}$ be the class of homogeneous halfspaces in $\mathbb{R}^d$ that labels $\mathcal{X}$. Let $S \subseteq \mathbb{R}^d$ be a set of n examples that are classified by some unknown halfspace $w^* \in S^{d-1}$. Let $w \in S^{d-1}$ be a unit vector such that $w \cdot w^* \geq \Omega(1/\sqrt{d})$. Let L be any labeling domain such that $S \subseteq L$. Denote by L the output of Algorithm 7 with input (w, S). If $S \subseteq S^{d-1}$ is in $1/2d$-approximate radially isotropic position, then Algorithm 7 makes $O(d^2 \log^2 d)$ queries from a query family Q with VC dimension $O(d^2)$, runs in $\mathrm{poly}(d, n, T)$ time and returns L such that each $(x, y) \in L$, $y = w^*(x)$ and $|L| \geq n/4d$. Here, T is the running time to implement a single query from Q.*

Proof (Proof of Theorem 33) We first show the correctness of the algorithm. i.e. Each element in the labeled set L has the correct label. No matter what the labeling domain is if some query $q(Z, y) = 1$, then every example $x \in Z \cap S$ must has a label y. Thus, each labeled example in the output is correctly labeled.

In the second step, we show that when w and w^* have a good correlation, Algorithm 7 will terminate after $\Omega(d \log d)$ iterations. In particular, we show the following robust proposition of the modified perceptron algorithm. We have the following claim.

Claim 34 *Let $w^*, w_0 \in \mathbb{R}^d$ be two unit vectors such that $w^* \cdot w_0 \geq \Omega(1/\sqrt{d})$. Assume the following update, $w_{t+1} = w_t - x_t(x_t \cdot w_t)$ and for some $t_0 \in Z$, such that for every $t \leq t_0$, $|x_t \cdot w_t| \geq \left\|w_t\right\|/2\sqrt{d}$ and $(x_t \cdot w_t)(x_t \cdot w^*) \leq 1/\mathrm{poly}(d)$. Then we have $t_0 \leq O(d \log d)$.*

We prove the claim here. By the update, we have

$$w_{t+1} \cdot w^* = (w_t - x_t(x_t \cdot w_t)) \cdot w^* = w_t \cdot w^* - (x_t \cdot w_t)(x_t \cdot w^*) \geq w_t \cdot w^* - \frac{1}{d^2} \geq w_0 \cdot w^* - \frac{t_0}{d^2}.$$

On the other hand, we have

$$\|w_{t+1}\|^2 \leq \|w_t\|^2 - (w_t \cdot x_t)^2 \leq (1 - 1/2d) \|w_t\|^2 .$$

If $t_0 \geq \Omega(d \log d)$, then before reach t_0, at some point we will have $\frac{w_t \cdot w^*}{\|w_t\|} > 1$, which gives a contradiction.

Now, we will show Algorithm 7 terminates before $t \geq \Omega(d \log d)$, by showing that in each iteration, the example x_t we use to update w_t satisfies the condition in the statement of Claim 34. There are two cases to consider.

In the first case, we update w_t via some $x_t = x' \in S \cap \{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}$ because $(w_t \cdot x_t)(w^* \cdot x) \leq 0$. Clearly, x_t is an example that satisfies the update requirement.

In the second case, we know that the example x' is correctly labeled by our current hypothesis w_t. In this case, according to Algorithm 7, we partition the region $\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}$ into boxes with diameter $1/\text{poly}(d)$. Since $q(\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}, y) = 0$, we know that there must be a "point" $x'' \in \{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}$ such that x'' is labeled incorrectly by w_t. Although such a point x'' may or may not be in our dataset S, it must be in one of these small boxes. Thus, Algorithm 7 will finally find such a small box such that

$$q(\{x \in B \mid yv_k \cdot x \in [a^{(k)}, b^{(k)}], 0 \leq k \leq d-1\} \cup \{x'\}, y) = 0.$$

Since x' has a label y, we know that no matter what the labeling domain is, there must be some x'' labeled incorrectly by w_t that is in this small box. Let $\tilde{x}$ be any point in the box. If $\tilde{x}$ is actually labeled incorrectly by w_t, then we make a good enough update because $(\tilde{x} \cdot w_t)(\tilde{x} \cdot w^*) \leq 0$. Otherwise, we show such an update is not that bad. Since $\left|a^{(k)} - b^{(k)}\right| \leq 1/\text{poly}(d)$, we know that $\|\tilde{x} - x''\| \leq d/\text{poly}(d) = 1/\text{poly}(d)$. On the other hand, each update will increase the length of w_t by at most 1, which implies that $\|w_t\| \leq t + 1 \leq O(d \log d)$. This implies that

$$(\tilde{x} \cdot w_t)(\tilde{x} \cdot w^*) = (\tilde{x} \cdot w_t)(x'' \cdot w^* + (\tilde{x} - x'') \cdot w^*) \leq (\tilde{x} \cdot w_t)((\tilde{x} - x'') \cdot w^*) \leq \frac{O(d \log d)}{\text{poly}(d)} = \frac{1}{\text{poly}(d)}.$$

So in general, we have $(\tilde{x} \cdot w_t)(\tilde{x} \cdot w^*) \leq 1/\text{poly}(d)$. In the meantime, since

$$\tilde{x} \in \{x \in B \mid yv_k \cdot x \in [a^{(k)}, b^{(k)}], 0 \leq k \leq d-1\} \cup \{x'\} \subseteq \{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\},$$

we have $|\tilde{x} \cdot w_t| \geq \|w_t\| / 2\sqrt{d}$. So in each round the example $x_t(x'$ or $\tilde{x})$ we use to update w_t always satisfies the update requirement thus after at most $t = O(d \log d)$ update, w_t correctly label every example x in the unit ball B with $|\tilde{x} \cdot w_t| \geq \|w_t\| / 2\sqrt{d}$. When we reach this stage, no matter what the labeling domain is, as long as some example $x \in \{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\} \cap S$, we always have

$$q(\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}, y) = 1,$$

which will make Algorithm 7 terminate. Furthermore, if S is $1/2d$-approximate radially isotropic position, by Lemma 31, we know that

$$|\{x \in B \mid |v_0 \cdot x| \geq \frac{1}{2\sqrt{d}}\} \cap S| \geq n/4d,$$

which implies that Algorithm 7 correctly label $1/4d$-fraction of the examples.

In the third step, we upper bound the query complexity and the running time of Algorithm 7. In each perceptron update for w_t, we make d binary searches over $\mathrm{poly}(d)$ cells to find an example to update w_t. Each binary search makes $O(\log d)$ queries and in total we make $O(d \log d)$ queries to make one update. Since we make at most $O(d \log d)$ updates, we know the query complexity of Algorithm 7 is $O(d^2 \log^2 d)$.

Finally, we upper bound the VC dimension of the query family Q. Since each query Algorithm 7 is a set of $O(d)$ d-dimensional linear inequalities, we know that the VC dimension of Q is $O(d^2)$. ■

Algorithm 7 ACTIVEPERCEPTRON(w, S) (Label a large fraction of example in S)

```
$d \leftarrow \dim(S)$ $t \leftarrow 0$ $w_t \leftarrow w$
while $t \leq O(d \log d)$ do
    Let $v_0 = w_t / \|w_t\|$ and $B$ be the unit sphere
    Make query $q(\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}, y)$, if $\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\} \cap S \neq 0$.
    if Every query made above returns 1 then
        return $\{(x, y) \mid x \in S, y \in \{-1, 1\}, yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}$
    else
        Let $y \in \{-1, 1\}$ such that $q(\{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}, y) = 0$
        Query $q(\{x'\}, y)$, where $x' \in S \cap \{x \in B \mid yv_0 \cdot x \geq \frac{1}{2\sqrt{d}}\}$
        if $q(\{x'\}, y) = 0$ then
            $x_t = x'$, $w_t \leftarrow w_t - (w_t \cdot x_t)x_t$, $t \leftarrow t + 1$
        else
            Let $1/2\sqrt{d} = \theta_0 \leq \theta_1 \leq \cdots \leq \theta_\ell = 1$ such that $\theta_i - \theta_{i-1} = 1/\mathrm{poly}(d)$.
            Find some $[a^{(0)}, b^{(0)}] := [\theta_{i-1}, \theta_i]$ for some $i \in [\ell]$ such that $q(\{x \in B \mid yv_0 \cdot x \in [a^{(0)}, b^{(0)}]\} \cup \{x'\}, y) = 0$.
            ▷ This can be done with $O(\log d)$ queries via binary search by making query of the form $q(\{x \in B \mid yv_0 \cdot x \geq \theta_j\} \cup \{x'\}, y)$ .
            Let $v_1, \ldots, v_{d-1}$ be a standard basis of the subspace orthogonal to $w_t$.
            Let $-1 = \theta_0 \leq \theta_1 \leq \cdots \leq \theta_\ell = 1$ such that $\theta_i - \theta_{i-1} = 1/\mathrm{poly}(d)$.
            for $j \in [d-1]$ do
                Find some $[a^{(j)}, b^{(j)}] := [\theta_{i-1}, \theta_i]$ for some $i \in [\ell]$ such that $q(\{x \in B \mid yv_k \cdot x \in [a^{(k)}, b^{(k)}], 0 \leq k \leq j\} \cup \{x'\}, y) = 0$ via binary search.
            Let $\tilde{x}$ be any point in $\{x \in B \mid yv_k \cdot x \in [a^{(k)}, b^{(k)}], 0 \leq k \leq d-1\}$
            $x_t = \tilde{x}$, $w_t \leftarrow w_t - (w_t \cdot x_t)x_t$, $t \leftarrow t + 1$
return $\emptyset$                ▷ If $w$ is not a good initialization, no example will be labeled.
```

Finally, we present Algorithm 8, the halfspace learning algorithm and the proof of Theorem 9.

Proof (Proof of Theorem 9) In the first step, we show the correctness of Algorithm 8. In each round of Algorithm 8, we find a subspace V that contains k/d fraction of the unlabeled data in S and a

Algorithm 8 LEARNINGLTF(S, α) (Label $1-\alpha$ fraction of S with simple query)

```
$L \leftarrow \emptyset$, $n \leftarrow |S|$
while $|L| < (1-\alpha)n$ do
    Apply Theorem 32 to $S$ with $\epsilon = 1/2d$ to obtain a matrix $A$ and a $k$-dimensional subspace $V$
    if $q(V, 1) = 1$ then
        $L_V \leftarrow \{(f_A(x), 1) \mid x \in S \cap V\}$
    else
        $L_V \leftarrow \emptyset$
    while $|L_V| < |S \cap V|/4k$ do
        Draw $w_0$ uniformly from the unit sphere in $A(V)$
        $L_V \leftarrow$ ACTIVEPERCEPTRON$(w_0, f_A(S \cap V))$
                                                ▷ If $w_0$ is not a good initialization, $L_V = \emptyset$.
    ▷ To run ACTIVEPERCEPTRON$(w_0, f_A(S \cap V))$, we implement each query $(Z, y)$ by query
  $(\{x \in V \mid f_A(x) \in Z\}, y)$.
    Label every $x \in S \cap V$ by $y$ if $(f_A(x), y) \in L_V$
    $L \leftarrow L \cup \{x \in S \cap V \mid f_A(x) \in L_V\}$, $S \leftarrow S \setminus L$
```

matrix A that can make $f_A(S \cap V)$ in approximate radially isotropic position. Denote by B the unit sphere in $A(V)$, we notice that for every $x \in V$, we have

$$\mathrm{sign}(w^* \cdot x) = \mathrm{sign}(A^{-T} w^* \cdot Ax) = \mathrm{sign}(A^{-T} w^* \cdot f_A(x)) = \mathrm{sign}(\mathrm{proj}_{A(V)}(A^{-T} w^*) \cdot f_A(x)),$$

which implies that we can view $f_A(V)$ to be labeled by a halfspace $v^* = \mathrm{proj}_{A(V)}(A^{-T} w^*)$ furthermore, x and $f_A(x)$ have the same label. According to Theorem 33, we know that each labeled example in the output of Algorithm 7 has the correct label with respect to v^* and thus the corresponding original examples in $S \cap V$ are also labeled correctly.

However, up to now, we have not shown the correctness of the algorithm. This is because when we call Algorithm 7 as a subroutine in Algorithm 8, we are not able to make queries in the transformed subspace $A(V)$, since it could be the case that $A(V) \cap Y = \emptyset$. Instead, we have to simulate a query $q(Z, y)$ used by Algorithm 7 with a query $q(\{x \in V \mid f_A(x) \in Z\}, y)$ in the original space.

To show such a simulation is successful, it suffices to show the simulation has the following two properties. First, every subset $Z \subseteq B$ contains some transformed example $f_A(x) \in f_A(S \cap V)$ if and only if $\{x \in V \mid f_A(x) \in Z\}$ contains some example $x \in S$. This property ensures that the transformed labeling domain $f_A(Y \cap V)$ also contains the transformed dataset $f_A(S \cap V)$. Second, for every $f_A(x) \in Z \subseteq B$ if it is labeled y by v^* then x is labeled y by w^*. This implies that the $q(Z, v^*, y) = q(\{x \in V \mid f_A(x) \in Z\}, w^*, y)$ is always true. Thus, we can safely run Algorithm 7 with the simulated query. Since Algorithm 8 terminates when $(1-\alpha)$ fraction of the examples have been labeled and every labeled example has the correct label, we finish showing the correctness of the algorithm.

In the second step, we bound the query complexity and the running time of the algorithm. We first upper bound the number of calls for Algorithm 7. Since the transformed subspace $A(V)$ has dimension k, by (Vershynin, 2018), we know that with probability at least some constant c, a random selected w_0 satisfies $|w_0 \cdot v^*| \geq 1/2\sqrt{k}$. When this happens, according to Theorem 33, we know that

Algorithm 7 will correctly label $1/4k$ fraction of the transformed dataset $f_A(S \cap V)$ and Algorithm 8 will enter next round. Thus, in expectation Algorithm 7 will be called constant times and each call will make $O(k^2 \log^2 k) \leq O(d^2 \log^2 d)$ queries. According to Theorem 32, we know that in each round $|S \cap V|/|S| \geq k/d$ and $1/4k$ fraction of $S \cap V$ is labeled correctly. This implies that after $O(d \log(1/\alpha))$ rounds only αn examples are not labeled and Algorithm 8 will terminate. This implies the total query complexity is $O(d^3 \log^2 d \log(1/\alpha))$. In particular, by setting $\alpha = o(1/n)$, we know that by making $O(d^3 \log^2 d \log n)$ queries, Algorithm 8 perfectly label S. To upper bound the running time of the algorithm, we notice that in each round, we run Algorithm 7 and compute an approximate Forser's transform, each of time can be done in polynomial time. Since the total number of rounds is at most $O(d \log n)$, we know Algorithm 8 is also a polynomial time algorithm.

Finally, we upper bound the VC dimension of the query family Q. Notice that each query we make can be summarized as follows $\{x \in V \mid f_A(x) \in Z\}$, where $Z = \{x'\} \cup \{x \in A(V) \mid Cx \leq d\}$, where C has at most $O(d)$ constraints. Thus, each query is the interaction of $O(d)$ degree two polynomial inequalities and a subspace (unions with a single point), which has a VC dimension of $\tilde{O}(d^3)$. ∎

We notice that when we run Algorithm 7, the region T in a query (T, z) is a set of $O(d)$ linear inequalities. Since in our learning model, each query is binary, we have to use such region queries to do a binary search in order to find some point x such that $(x \cdot w_t)(x \cdot w^*) \leq 1/\text{poly}(d)$. Thus, when we run Algorithm 8, each query uses a region that is the interaction of $O(d)$ degree two polynomial inequalities, and a subspace. This is why the VC dimension of Q in Theorem 9 is $\tilde{O}(d^3)$. If we are in a stronger learning model, where a counter-example $x \in T \cap L$ with label $-z$ is also returned when $q(T, z) = 0$, then the binary search approach in Algorithm 7 is not necessary. In this setting, in Algorithm 7, each region is defined by a single halfspace, and thus the VC dimension of the query class Q we use for Algorithm 8 will be improved to $\tilde{O}(d^2)$.

Appendix D. Learning A Specific Hypothesis Class via A Specific Query Class

Although Theorem 2 shows that given a hypothesis class $\mathcal{H}$ with VC dimension d, we can construct a query class Q with VC dimension $O(d)$ so that using Q, we can design a learning algorithm with query complexity, we have also seen from Section 3 that if a hypothesis class has a good structure, a query class with VC dimension $O(\log d)$ or even constant is sufficient to achieve a query complexity of $O(\log n)$. So an interesting question is given a hypothesis class $\mathcal{H}$ and a query class Q, what is the query complexity of learning $\mathcal{H}$ with Q? Such a question has been extensively studied in many works in the literature of exact learning such as (Angluin, 1988; Balcázar et al., 2001, 2002; Chase and Freitag, 2020). Many different combinatorial characterizations have been developed. As a by-product of Theorem 2, we can also define a new combinatorial dimension to characterize the query complexity of using a specific query class Q to learn $\mathcal{H}$.

Definition 35 (Partial Labeling and Extension) *Let $\mathcal{X}$ be a space of examples and $\mathcal{H}$ be a hypothesis class over $\mathcal{X}$. Let $S \subseteq \mathcal{X}$ be a set of n examples. A partial labeling f over S is a labeling function $f : S' \subseteq S \to \{-1, 1\}$, where $S' \subseteq S$. We say hypothesis $h \in \mathcal{H}$ is an extension of f iff for every $x \in S'$, $f(x) = h(x)$. In particular, we denote by $H_f = \{h \in \mathcal{H} \mid h(x) = f(x), \forall x \in S'\}$ the set of extensions of f in $\mathcal{H}$.*

Definition 36 (Generalized Teaching Tree) *Let $\mathcal{X}$ be a space of examples, $\mathcal{H}$ be a hypothesis class over $\mathcal{X}$ and Q be a query family. Let $S \subseteq \mathcal{X}$ be a set of n examples. A generalized teaching tree T_f for f is a binary tree that satisfies the following properties.*

- *Each node v of T_f is associated with a subset H_v of hypothesis in $\mathcal{H}$. The root of T_f is associated with $\mathcal{H}$.*
- *Each internal node v of T_f is also associated with a query $q_v \in Q$.*
- *Denote by v_l and v_r the left child of an internal node v. $H_{v_l} := \{h \in H_v \mid q_v(h) = 0\}$, $H_r := \{h \in H_v \mid q_v(h) = 1\}$.*
- *The subset of hypothesis H_v associated with a leaf v is either a subset of H_f or a subset of $\mathcal{H} \setminus H_f$.*

Definition 37 (Query Dimension) *Let $\mathcal{X}$ be a space of examples. Let $\mathcal{H}$ be a class of hypotheses over $\mathcal{X}$ and Q be a family of queries. For any $n \in N^+$, we define $s(n)$ to be the query dimension of $(\mathcal{H}, Q)$ as follows.*

$$s(n) = \max_{S \subseteq \mathcal{X}, |S|=n} \max_{f\text{:partial labeling over } S} \min\{\textit{depth}(T_f) \mid T_f : \textit{a generalized teaching tree for } f\}.$$

Theorem 38 *Let $\mathcal{X}$ be a space of examples. Let $\mathcal{H}$ be a class of hypotheses with VC dimension d and Q be a family of queries. Let $s(n)$ be the query dimension of $(\mathcal{H}, Q)$.*

- *For any deterministic active learner $\mathcal{A}$, there is a subset $S \subseteq \mathcal{X}$ of n such that if $\mathcal{A}$ makes less than $s(n)$ queries then there is some h^* and some $x \in S$ such that $\mathcal{A}$ labels x incorrectly.*
- *There is an active learner $\mathcal{A}$ such that, for every subset of n example and every $h^* \in \mathcal{H}$, $\mathcal{A}$ makes $O(s(n)d \log n)$ queries from Q and labels every example in S correctly.*

Proof Clearly, given any set S of n examples, every active learning algorithm $\mathcal{A}$ constructs a generalized teaching tree for every partial labeling function f, because each leaf of the tree corresponds to the hypothesis in H that labels S in the same way. In particular, let S^* be the set of n examples that achieves the maximum in the definition of $s(n)$, then the number of queries needed for $\mathcal{A}$ to label S is at least the depth of the teaching tree it constructs which is larger than the number of queries to teach any partial labeling f.

On the other hand, by Lemma 15, we know that for every hypothesis class H' and every set of n examples S, there is some partial labeling f such that $|(H'_f)_S|/|H'_S| \in [1/3, 2/3]$. ($H'_S$ is H' restricted at S and $(H'_f)_S$ is H'_f restricted at S.) Thus, with at most $s(n)$ queries, we are able to check if the target hypothesis is in H'_f or not and shrink the hypothesis class by a factor of constant. Thus after making $O(s(n)d \log n)$ queries, we label S correctly. ■