Active Learning with Label Comparisons

Abstract

Supervised learning typically relies on manual annotation of the true labels. When there are many potential classes, searching for the best one can be prohibitive for a human annotator. On the other hand, comparing two candidate labels is often much easier. We focus on this type of pairwise supervision and ask how it can be used effectively in learning, and in particular in active learning. We obtain several insightful results in this context. In principle, finding the best of k labels can be done with $k-1$ active queries. We show that there is a natural class where this approach is sub-optimal, and that there is a more comparison-efficient active learning scheme. A key element in our analysis is the "label neighborhood graph" of the true distribution, which has an edge between two classes if they share a decision boundary. We also show that in the PAC setting, pairwise comparisons cannot provide improved sample complexity in the worst case. We complement our theoretical results with experiments, clearly demonstrating the effect of the neighborhood graph on sample complexity.

1 INTRODUCTION

Supervised learning is a central paradigm in the empirical success of machine learning in general, and deep learning in particular. Despite the recent advances in unsupervised learning, and in particular self-training, large amounts of annotated data are still required in order to achieve high accuracy in many tasks. The main difficulty with supervised learning is, of course, the manual effort needed for annotating examples. Annotation becomes particularly challenging when there are many classes to consider. For example, in a text summarization task, we can ask an annotator to write a summary of the source text, but this will likely not result in the "best" summary. We could also present the annotator a summary and ask for feedback (e.g. is it good), but the quality could be difficult to judge in isolation. We could also ask the annotator to select the best summary of a candidates set (e.g. produced by a language model), but this could be taxing if not infeasible when there are many candidates.

Motivated by the above scenario, previous works [e.g., see Stiennon et al., 2020, Ouyang et al., 2022, for a recent application to large language models] have considered an alternative, and arguably natural, form of supervision: "Label Comparisons". Instead of presenting many potential labels to the annotator (e.g., candidate text summaries), we only present two candidates and ask the annotator to choose the better one. For example, when summarizing Snow White, we can ask to compare the summaries "A story about an evil step-mother" and "A story about a girl driven to the forest by an evil step-mother and lives with dwarves". Most annotators would easily choose the latter as a better summary.

Label comparisons clearly require a much lighter cognitive load than considering all alternatives, and thus have high potential as an annotation mechanism. However, our theoretical understanding of this mechanism is fairly limited. While there has been work on learning to rank, which also uses comparisons, the goal of label comparisons is typically not to learn a complete ranking, but rather to build a model that outputs optimal predictions. Here we set out to analyze label comparisons from this perspective, and obtain several surprising results and a new algorithm.

Our key question is what is the best way to learn with label comparisons. We assume that during learning we can only ask an annotator for label comparisons and not, for example, for the ground-truth label of the input, which we refer to as an argmax query. We then ask how one can design algorithms that make effective use of such queries, and what is the corresponding query complexity. Namely, how many queries are needed to achieve a given test error. Perhaps, the most natural way of using comparisons is simply for finding the argmax label, which can be done via $k-1$ active queries.

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

However, as we shall see, this is a suboptimal approach.

The first question we ask is whether access to comparisons is more informative than access to the argmax. If we know all $\binom{k}{2}$ comparisons for x, we can also infer the argmax and so it would seems like the answer to this should be in the affirmative. Our first result shows that in the PAC setting, this is in fact not the case, and that knowing all comparisons may result in the same sample complexity as knowing only the argmax. The intuition for this negative result is that for 1D classifiers, the informative points are those that lie close to the decision boundaries between classes, and the argmax label for these points can also be used to find the boundaries, so that comparisons do not provide further advantage.

The negative result above may seem to suggest that comparisons are only useful for inferring the argmax. However, we show that in the case of active learning, comparisons can be used more effectively. We consider the setting where the active learner can choose which label comparison queries to request for a given input x (including not requesting any queries at all). A natural approach here is to take a "standard" active learning algorithm based on argmax queries, and implement it using pairwise comparisons, by using $k-1$ active comparisons for each input x to obtain the argmax. This strategy results in an algorithm that asks $\gamma(k-1)$ comparisons, where γ is the number of argmax queries used.

Here we show that one can in fact do better than simulating argmax, by asking the "right" comparisons in an active fashion. These beneficial comparisons are closely related to the "Label Neighborhood Graph" (see Figure 2) where labels are neighbors if they share a decision boundary. We show that it is sufficient to ask queries only about neighbor pairs in this graph. Thus, if this graph is sparse, active learning can be implemented with fewer queries. In particular, for linear classifiers in $\mathbb{R}$, each class has at most two neighbors, and thus the neighborhood graph is very sparse, and our proposed active learning approach is highly effective. Taken together, our results demonstrate the richness of the label-comparison setting, and the ways in which its query complexity depends on the structure of the data.

2 RELATED WORK

Several lines of works have addressed alternative modes of supervision for multi-class learning.

Bandit Feedback: In this setting (e.g., Kakade et al. [2008], Crammer and Gentile [2013]) the learner only observes whether its predicted class is correct or not. On the one hand, this feedback is stronger than label comparisons, because positive bandit feedback implies knowledge of the argmax. On the other hand, label comparisons provide more information than bandit feedback, because comparisons provide knowledge about relative ordering of non argmax labels.

Maxing from pairwise comparisons. Maximum selection (maxing) from noisy comparisons is well-studied problem. Falahatgar et al. [2018] give an overview of known results under various noise models. Differently, we show that, for multiclass learning, using comparisons to first learn the global structure of the problem is more efficient than only using them for maxing. Daskalakis et al. [2011] consider maxing in partially ordered sets, where some pairs may be incomparable, which is interesting to explore in our setting.

Dueling Bandits: In online learning, learning from pairwise comparisons is studied under the dueling bandits setting [Saha et al., 2021, Dudík et al., 2015], in which the learner "pulls" a pair of arms and observes the result of a noisy comparison (duel) between them. The objective in these cases is to minimize the regret w.r.t a solution-concept from the social choice literature, e.g. the Condorcet winner [Yue et al., 2012], Borda winner, Copeland winner, or the Von Neuman winner [Dudík et al., 2015]. The focus on such regret minimization objectives is principally different from ours, since our primary goal is to minimize the number of queries made, rather than minimizing an online loss.

Active Learning with rich supervision: Several works have explored alternative forms of supervision. Balcan and Hanneke [2012] explore class-conditional queries, where the annotator is given a target label and a pool of examples, and must say whether one of the examples matches the target label. Several works [Kane et al., 2017, Hopkins et al., 2020, Xu et al., 2017] have studied comparison queries on instances, where the annotator receives two inputs x_1, x_2 and reports which one is more positive (for binary classification). Ben-Eliezer et al. [2022] study active learning of polynomial threshold function in $d = 1$ using derivative queries (e.g., is a patient getting sicker or healthier?). Our supervision is conceptually different from all of these, as it compares between several labels on the same sample x.

Learning Ranking as a Reward Signal: A recent line of work demonstrated that pairwise label-comparisons elicited from humans can be used to improve the performance of LLMs. Stiennon et al. [2020] collect a dataset of human comparisons between summaries of a given text, and use it to obtain better summarization policies, and Ouyang et al. [2022] extend this idea to aligning LLMs with user intent. Our focus here is to understand the theoretical properties of such label comparisons, which we expect will result in more effective ways of collecting and using such comparisons.

3 PRELIMINARIES

Multi-class learning. Let $\mathcal{X} \subset \mathbb{R}^d$ denote the feature space and Y denote the label space, consisting of k classes. We use $\mathcal{D}$ to denote an (unknown) distribution on $\mathcal{X}$ and $\mathcal{H}$ to denote a class of target functions, $f : \mathcal{X} \to \mathbb{R}^k$. In this work, our focus is on a realizable setting in which the target

function is some (unknown) $f^\star \in \mathcal{H}$. For $\boldsymbol{x} \in \mathbb{R}^d$ and a class $i \in [k]$, $f_i(\boldsymbol{x})$ is the score assigned to class i on instance $\boldsymbol{x}$. Given a target function $f^\star$, the loss of a candidate classifier f is the standard (multiclass) 0-1 loss: $L(f) = \mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}}[\arg\max_{i\in[k]} f_i(\boldsymbol{x}) \neq \arg\max_{i\in[k]} f_i^\star(\boldsymbol{x})]$.

Since we are interested in how the difficulty of learning scales with the number of classes k, we will explicitly parameterize hypothesis classes in terms of k, $\{\mathcal{H}^k\}_{k\in\mathbb{N}}$. For example, the class of homogeneous linear classifiers[1] over $k \in \mathbb{N}$ classes in dimension d is $\mathcal{H}^{k,d}_{\text{lin}} = \{h(\cdot;\boldsymbol{W}) : \boldsymbol{W} \in \mathbb{R}^{k\times d}\}$, where $h(\boldsymbol{x};\boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^k$.

Supervision Oracles. Pertinent to this work is a distinction between two types of access to the target multiclass function: *argmax (i.e. label) queries* and *label-comparison queries*.

Definition 3.1 (Supervision Oracles)**.** *Given a target function $f^\star : \mathcal{X} \to \mathbb{R}^k$, we define the following oracles:*

$$A^{f^\star}(\boldsymbol{x}) = \arg\max_{i\in[k]} f_i^\star(\boldsymbol{x})$$

$$A^{f^\star}(\boldsymbol{x}, j_1, j_2) = \mathbf{1}[f_{j_1}^\star(\boldsymbol{x}) > f_{j_2}^\star(\boldsymbol{x})]$$

In the rest of the manuscript we simply use $A^{f^\star}$ to denote the supervision oracle, where it's understood that if it receives an input $\boldsymbol{x}$ it invokes the argmax oracle and if it receives a triplet $\boldsymbol{x}, j_1, j_2$ it invokes the comparisons oracle.

4 PASSIVE LEARNING

We define the sample and query complexities of PAC learnability using both argmax and label-comparisons supervision.[2] We begin with the usual *passive learning* setup, and differentiate between the situation in which every example arrives with its argmax (i.e., the standard PAC setup), and where every example arrives with all the $\binom{k}{2}$ pairwise label comparisons (essentially, the total order on the classes).

Definition 4.1 (Sample complexity of passive learning with argmax supervision)**.** *Fix a distribution $\mathcal{D}$ over $\mathcal{X}$ and a target function $f^\star : \mathcal{X} \to \mathbb{R}^k$. Let $\mathcal{D}^{f^\star}$ denote the distribution on $\mathcal{X} \times Y$ in which a sample $(\boldsymbol{x}, y) \sim \mathcal{D}^{f^\star}$ is generated by drawing $\boldsymbol{x} \sim \mathcal{D}$ and taking $y = A^{f^\star}(\boldsymbol{x})$.*

We say that the sample complexity of passively learning a class $\{\mathcal{H}^k\}_{k\in\mathbb{N}}$ is $m_{\mathcal{H}} : (0,1) \times \mathbb{N} \to N$ if there exists a learning algorithm with the following property: for every distribution $\mathcal{D}$ on $\mathcal{X}$, for every $k \in \mathbb{N}$, for every $f^\star \in H^k$, and for every $\varepsilon \in (0,1)$, given $m \geq m_{\mathcal{H}}(\varepsilon, k)$ i.i.d samples from $\mathcal{D}^{f^\star}$, the algorithm returns an hypothesis h s.t w.p at least $7/8$, $L_{\mathcal{D}}(h) \leq \varepsilon$.

[1]Our convention will be to use homogeneous linear classifiers. Thus when we refer to our results for 1d, we mean the class $\mathcal{H}^{k,2}_{\text{lin}}$.

[2]For simplicity, we consider a PAC notion where the goal is to return ϵ-accurate solutions with constant probability (e.g. 7/8).

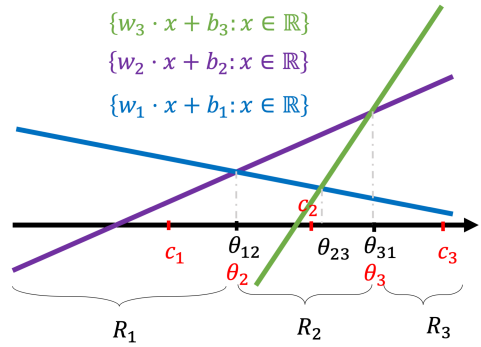

Figure 1: Equivalent view of non-homogeneous linear classifiers in 1d in terms of 1NN classification.

Definition 4.2 (Sample complexity of passive learning with label-comparisons.)**.** *Fix a distribution $\mathcal{D}$ over $\mathcal{X}$ and a target function $f^\star : \mathcal{X} \to \mathbb{R}^k$. Let $\mathcal{D}^{f^\star}$ denote a distribution on $\mathcal{X} \times \{\pm 1\}^{k^2}$ where a sample $(\boldsymbol{x}, \{b_{ij}\}_{i,j=1}^k)$ is generated by drawing $\boldsymbol{x} \sim \mathcal{D}$ and for $i, j \in [k]$, taking $b_{ij} = A^f(\boldsymbol{x}; i, j)$.*

We say that the sample complexity of passively learning a class $\{\mathcal{H}^k\}_{k\in\mathbb{N}}$ is $m_{\mathcal{H}} : (0,1) \times \mathbb{N} \to N$ if there exists a learning algorithm with the following property: for every distribution $\mathcal{D}$ on $\mathcal{X}$, for every $k \in \mathbb{N}$, for every $f^\star \in H^k$, and for every $\varepsilon \in (0,1)$, given $m \geq m_{\mathcal{H}}(\varepsilon, k)$ i.i.d samples from $\mathcal{D}^{f^\star}$, the algorithm returns an hypothesis h s.t w.p at least $7/8$, $L_{\mathcal{D}}(h) \leq \varepsilon$.

Note that in the latter setting, the learner receives strictly more information about every example: the argmax can always be inferred from the total order on the classes. We will therefore consider label-comparisons as helpful in this setup if knowing all comparisons results in improvement to the sample complexity. Our first result is negative: in general, label-comparisons may not be helpful in the passive regime.

Theorem 4.3. *Any algorithm that PAC learns $\mathcal{H}^{k,2}_{\text{lin}}$ must use $m_{\mathcal{H}}(\epsilon, k) \in \Omega(k/\epsilon)$ samples, irrespective of whether it has access to argmax or label-comparison supervision.*

Proof. For regular PAC learning (with argmax supervision), the standard approach for lower bounding the sample complexity is to lower bound the Natarajan dimension [Natarajan, 1989]. To extend this result to the setting of Definition 4.2, we employ the dimension introduced in Daniely and Shalev-Shwartz [2014], which we will refer to as the Daniely-Shwartz dimension. It provides a tighter lower bound on the sample complexity, and it is also easier to adapt to our label comparison setting.

To emphasize the difference between functions mapping $\boldsymbol{x} \in \mathcal{X}$ to a single class $y \in [k]$ and functions mapping $\boldsymbol{x} \in \mathcal{X}$ to a total order over the k classes, we will denote the former with f and the latter with $f_\triangledown$ (and likewise for hypotheses classes). We write $\arg\max f_\triangledown(\boldsymbol{x}) \in [k]$ for the class ranked first in the total order $f_\triangledown(\boldsymbol{x})$.

Definition 4.4. *Given a set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subset \mathcal{X}$, we say that f_∇ and g_∇ are $\boldsymbol{x}_i$-close if:*

$$\begin{cases} f_\nabla(x_j) = g_\nabla(x_j) & j \neq i \\ \arg\max f_\nabla(x_j) \neq \arg\max g_\nabla(x_j) & j = i \end{cases}$$

With this we can define a variant of Definition 12 in Daniely and Shalev-Shwartz [2014] for the case of extra supervision.

Definition 4.5 (The Daniely-Shwartz dimension for label comparisons.)**.** *A set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ is shattered by $\mathcal{H}_\nabla$ if there exists a subset of functions $\mathcal{H}'_\nabla \subset \mathcal{H}_\nabla$ with the following property: for every $f_\nabla \in \mathcal{H}'_\nabla$ and for every $i \in [n]$, there exists $g_\nabla \in \mathcal{H}'_\nabla$ such that f_∇, g_∇ are $\boldsymbol{x}_i$-close. The Daniely-Shwartz dimension of $\mathcal{H}_\nabla$, dim$(\mathcal{H}_\nabla)$, is the maximal cardinality of a shattered set.*

Daniely and Shalev-Shwartz [2014] prove that the sample complexity of learning a class $\mathcal{H}$ is $\Omega(\dim(\mathcal{H})/\varepsilon)$. Thus, our objective is to prove that $\dim(\mathcal{H}^{2,k}_{\text{lin}}) \in \Omega(k)$.

To show this, we will construct a shattered set of size k for $\mathcal{H}^{2,2k}_{\text{lin}}$. Consider $2k$ labels of the form (b, i), where $b \in \{0, 1\}$ and $i \in \{1, \dots, k\}$. Partition the numbers $1, \dots, 3k$ to k triples: $\{1, 2, 3\}$, $\{4, 5, 6\}$, $\dots$ $\{3k-2, 3k-1, 3k\}$. We claim that k middle points, $S = \{2, 5, \dots 3k-1\}$ are shattered by $\mathcal{H}^{2,2k}_{\text{lin}}$. Showing this requires defining a subset $\mathcal{F}$ of $\mathcal{H}^{2,2k}_{\text{lin}}$ with the property of Definition 4.5. To define each function $f_\nabla \in \mathcal{F}$ we will use an equivalent parametrization of linear classifiers in 1d as 1NN classification. i.e., each total order in $\mathcal{H}^{2,2k}_{\text{lin}}$ is parameterized by $\boldsymbol{c} \in \mathbb{R}^{2k}$, where the total order $h(x; \boldsymbol{c})$ is the one implied by sorting the classes according to the distance of their centers $\boldsymbol{c}$ to x. See Figure 1 for an illustration. With this parameterization in mind, $\mathcal{F}$ consists of all functions which satisfy the following: for each $i \leq k$, the centers corresponding to labels $(0, i)$ and $(1, i)$ are located in the i'th triplet, and *exactly one of them* is located in the middle of the triplet, on the point $3i - 1$. By construction, $|\mathcal{F}| = 4^k$ (for each of the k triplets we need to specify which of the two centers is located in the middle of the triplet, and whether to locate the other center on the left or on the right of it).

To see that S is shattered, consider $f_\nabla \in \mathcal{F}$ and a point $3i - 1 \in S$. W.l.o.g, assume that the center located on $3i - 1$ is $(0, i)$. We define $g_\nabla \in \mathcal{F}$ based on the location of the center of $(1, i)$, which by definition of $\mathcal{F}$, could be either to the right (on $3i$) or to the left (on $3i - 2$). In the first case, g_∇ is obtained by shifting both centers one unit to the left: in g_∇ the center $(0, i)$ is located on $3i - 2$ and the center $(1, i)$ is located on $3i - 1$. In the second case, g_∇ is obtained by shifting both centers one unit to the right: in g_∇ the center $(0, i)$ is located on $3i$ and the center $(1, i)$ is located on $3i - 1$. By the definition of $\mathcal{F}$, $g_\nabla \in \mathcal{F}$. Crucially, f_∇ and g_∇ are $\{3i - 1\}$-close (Definition 4.4): moving from f_∇ to g_∇ the center located on $3i - 1$ (and therefore the argmax) has changed, but the total order for every other point in S is remained unchanged, per the requirement of Definition 4.4. This proves S is shattered, and so $\dim(\mathcal{H}^{2,k}_{\text{lin}}) \in \Omega(k)$, as required. □

5 ACTIVE LEARNING

Next, we consider the *active* learning setting. Specifically, we focus on pool-based active learning, where the learner has access to unlabeled samples and can decide which queries to ask the oracle for (including not asking any queries). The performance of the algorithm is now measured in terms of the query complexity, namely the number of queries it makes to the labeling oracle in question:

Definition 5.1 (Query complexity of active learning.)**.** *The query complexity of actively learning a class $\left\{\mathcal{H}^k\right\}_{k \in \mathbb{N}}$ is $q_{\mathcal{H}} : (0, 1) \times \mathbb{N} \to N$ if there exists a function $m_{\mathcal{H}} : (0, 1) \times \mathbb{N} \to N$ and a learning algorithm with the following property: for every distribution $\mathcal{D}$ on $\mathcal{X}$, for every $k \in \mathbb{N}$, for every $f^\star \in \mathcal{H}^k$, and for every $\varepsilon \in (0, 1)$, given $m \geq m_{\mathcal{H}}(\varepsilon, k)$ i.i.d samples from $\mathcal{D}$ and at most $q_{\mathcal{H}}(\varepsilon, k)$ queries to $A^{f^\star}$, the algorithm returns an hypothesis h s.t w.p at least $7/8$, $L_{\mathcal{D}}(h) \leq \varepsilon$. We refer to $q_{\mathcal{H}}$ as the query complexity of learning $\mathcal{H}$ with argmax supervision or with label-comparison supervision, depending the oracle $A^{f^\star}$.*

We note that every active learning algorithm that uses argmax queries can always be simulated using comparison queries: in the adaptive setting (where the choice of query to ask at time t can depend on the previous answers), $k - 1$ label-comparison queries suffice to implement a "tournament" that reveals the argmax. This provides a generic way to use the label-comparison oracle: simply request the label-comparison queries necessary for a "regular" active learner. We therefore say that *comparisons are useful for active learning* if the number of label-comparison queries required to learn a class $\mathcal{H}$ is *strictly lower* than the number of label-comparison queries required to simulate the best active learner that uses argmax queries to learn $\mathcal{H}$.

Interestingly, the distinction between passive and active learning is important. Our main result is that when the learner is allowed to decide which queries to request, label-comparisons are helpful: we provide a learning algorithm that uses label comparisons more efficiently than simply using them to implement the best "regular" active learner.

Theorem 5.2. *The label-comparison query complexity for active learning $\mathcal{H}^{k,2}_{\text{lin}}$ is $\tilde{O}(k \cdot \log \frac{1}{\varepsilon})$, whereas the query complexity of simulating the best argmax active learner is $\tilde{\Omega}(k^2 \cdot \log \frac{1}{\varepsilon})$.*

The proof of Theorem 5.2 will employ a specific multiclass to binary reduction that uses the concept of the *label neighborhood graph* of the target classifier. Intuitively, two classes

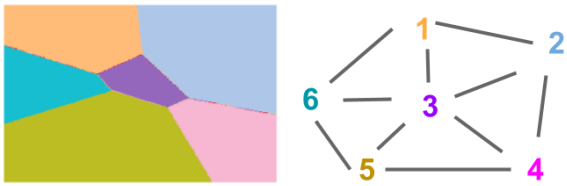

Figure 2: Decision regions of a linear classifier in 2d (left) and its corresponding label neighborhood graph (right).

Algorithm 1 `NbrGraphM2B`**: active learning of** $\mathcal{H}_{\text{lin}}^{k,d}$ **using** G**.**

Input: $\varepsilon > 0$, a binary active learning algorithm `B` with query complexity $q_b(\gamma)$, a neighborhood graph G.
Output: $f : \mathcal{X} \to \mathbb{R}^k$.
for $(i, j) \in G$ **do**
  Use `B` to learn a binary classifier that distinguishes class i from class j with error at most $\varepsilon/\mathrm{e}(G)$.
end for
Let C denote the set of all the learned binary classifiers.
Return $f^{(G,C)}$ (Definition 5.4).

i and j are considered neighboring if they share a decision boundary; i.e., there are two arbitrarily close points in $\mathbb{R}^d$, where for one the argmax is i and for the other the argmax is j. See Figure 2 for an example of the label neighborhood graph of a linear classifier in $d = 2$.

Definition 5.3 (Label Neighborhood graph)**.** *For a function $f : \mathbb{R}^d \to \mathbb{R}^k$, the neighborhood graph $G = G(f)$ is an undirected graph on k vertices, with an edge between vertices $i \in [k]$ and $j \in [k]$ if and only if there exists $\boldsymbol{x} \in \mathbb{R}^d$ for which for every $r \in [k]$, $f_i(\boldsymbol{x}) = f_j(\boldsymbol{x}) \geq f_r(\boldsymbol{x})$.*

To simplify notation, we use $(i, j) \in G$ to refer to an edge in G, and $\mathrm{e}(G)$ for the total number of edges. The degree of $i \in [k]$ is the number of neighbors i has in G.

We next define `NbrGraphM2B` (Neighborhood Graph Multiclass-to-Binary), a procedur for actively learning a multiclass classifier f using a neighborhood graph G (see Algorithm 1). Given as input a binary active learning algorithm and a neighborhood graph, it uses comparison queries to learn a binary classifier for distinguishing every pair of neighboring classes i, j in G. It then aggregates these into a multi-class classifier using the following scheme:

Definition 5.4 (Binary to multiclass aggregation.)**.** *Fix (G, C), where G is a neighborhood graph and $C = \{h_{ij}\}_{(i,j)\in G, i<j}$ is a collection of binary classifiers, one for every edge in G. The graph-based aggregation of (G, C) is a function $f^{(G,C)} : \mathcal{X} \to \mathbb{R}^k$ defined as follows:*

$$f_i^{(G,C)}(\boldsymbol{x}) = \frac{\sum_{(i,j)\in G} \mathbf{1}[h_{ij}(\boldsymbol{x}) \geq 0]}{\sum_{(i,j)\in G} \mathbf{1}}$$

Namely, the label of $\boldsymbol{x}$ is the class in $[k]$ that won the largest fraction of "duels" against its neighbors in the graph G.

An important component in analyzing `NbrGraphM2B` is the following lemma. It establishes that when invoked w.r.t the true neighborhood graph $G^\star$, if the binary classifiers are sufficiently accurate, then so is the resulting multiclass classifier. See Supplementary C for proof.

Lemma 5.5. *Fix a distribution $\mathcal{D}$ on $\mathcal{X}$ and a classifier $\boldsymbol{W}^\star$. Fix (G, C). If $G = G(\boldsymbol{W}^\star)$ and every $h_{ij} \in C$ has error at most $\varepsilon/e(G)$, then $f^{(G,C)}$ has error at most ε.*

From this, we obtain the following upper bound on the query complexity of learning $\mathcal{H}_{\text{lin}}^{k,d}$ using label comparisons.

Corollary 5.6. *If the target neighborhood graph $G^\star$ is known, the label-comparison query complexity of learning $\mathcal{H}_{\text{lin}}^{k,d}$ is $O(e(G^\star) \cdot q_b(\varepsilon/e(G^\star))$, where $q_b(\gamma)$ is the query complexity of active learning in the binary case (i.e. $k = 2$).*

Corollary 5.6 suggests that label-comparisons will be useful when (i) the target neighborhood graph is sparse (has low degree), and (ii) it can be learned with relatively few label-comparisons. We are now ready to prove Theorem 5.2: we will show that for learning $\mathcal{H}_{\text{lin}}^{k,2}$ (the class for which we demonstrated comparisons are not useful in the passive setting), both these conditions hold. Hence, comparisons indeed provide a provable gain over argmax supervision.

Proof of Theorem 5.2: We will begin by instantiating the bound from Corollary 5.6 for $d = 1$. Consider the degree of the neighborhood graph. Using the equivalent parameterization of linear classifiers in $d = 1$ (see Figure 1), it follows that every class $i \in [k]$ has at most 2 neighbors: exactly the preceding and succeeding classes in the sorted order of the classes. Thus, for every $f^\star \in \mathcal{H}_{\text{lin}}^{k,2}$, $\mathrm{e}(G(f^\star)) = O(k)$. Second, active learning in $d = 1$ is well-understood: unlike higher dimensions, the distribution-free query complexity of active learning for two classes is $q_b(\gamma) = \log(\frac{1}{\gamma})$ using binary search over $\mathbb{R}$ [Dasgupta, 2004]. Plugging both of these facts into the upper bound of Corollary 5.6, we conclude that the query complexity for learning $\mathcal{H}_{\text{lin}}^{k,2}$ when the target neighborhod graph is known is $O(k \cdot \log \frac{k}{\varepsilon})$.

It remains to reason about the number of label comparisons required to learn $G^\star$ (i.e., the sorted order of the classes). We claim that $O(k \log k)$ label-comparison queries suffice for this purpose. Consider Algorithm 2. It receives a sample of points n from $\mathcal{D}$ and uses exactly $2k + n \log n$ label comparisons to infer the sorted order of the classes they belong to. Thus, under the assumption that this procedure is correct, applying it to $O(k)$ samples (enough to guarantee that w.p at least 7/8, every label that has mass under $\mathcal{D}$ is also represented in S) indeed learns $G^\star$, and uses $O(k \log k)$ label comparisons to do so.

To prove correctness of the algorithm it suffices to prove that the learned total order is identical to the true total order.

Algorithm 2 **Learning $G(f^\star)$ for $f^\star \in \mathcal{H}^{k,2}_{\text{lin}}$.**

Input: n i.i.d samples from $\mathcal{D}$, $x_1, \ldots, x_n$.
Output: A neighborhood graph G.

Set $x_L = \min_i x_i$ and $x_R = \max_i x_i$.
for $i < j \in [k]$ **do**
  Determine the partial order between i and j: i appears before j if and only if (i) $A^f(x_L, i, j) = 1$ and $A^f(x_R, i, j) = 0$, or (ii) $A^f(x_L, i, j) = 1$ and $A^f(x_R, i, j) = 1$.
end for
Use a sorting procedure to obtain a total order over the k classes, $i_1 \succ \cdots \succ i_k$.
Define a neighborhood graph G with an edge between i and j iff classes are consecutive in the learned total order.

Return G.

Indeed, first note that if $A^f(x_L, i, j) = A^f(x_R, i, j)$ then the "losing" class is not an actual class, in that none of the n points belong to it. So we can safely place it last in the total order (effectively ignoring this class), and it will not effect the labels of $x_1, \ldots, x_n$. On the other hand, if $A^f(x_L, i, j) \neq A^f(x_R, i, j)$, then their values directly determine the order of the classes: class i appears before class j if and only if $A^f(x_L, i, j) = 1$ and $A^f(x_R, i, j) = 0$.

We can now combine both parts (using Algorithm 2 to learn $G^\star$, and then using Algorithm `NbrGraphM2B`) to conclude that the overall query complexity of learning $\mathcal{H}^{k,2}_{\text{lin}}$ is $O(k \log k + k \log \frac{k}{\varepsilon}) = \tilde{O}(k \cdot \log \frac{1}{\varepsilon})$, as required.

To conclude the proof of Theorem 5.2, it remains to lower bound the complexity of learning with argmax queries. We will prove that $\Omega(\frac{k}{\log k} \log \frac{k}{\varepsilon})$ argmax queries are needed. This will imply that simulating any argmax active learning requires at least $\tilde{\Omega}(k^2 \cdot \log \frac{1}{\varepsilon})$ label-comparisons. We will prove this via the label revealing task [e.g., see Kane et al., 2017], where the goal is to reveal the correct labels of a given (realizable) sample of n points, and show that $O(\frac{k}{\log k} \log n)$ argmax queries are needed to reveal all n labels.

Towards this, fix n points and consider a tree that denotes the run of an active learning algorithm (with nodes being the queries asked and the children the possible answers). Note that the number of unique labelings corresponds to the number of leaves in the tree and the query complexity corresponds to the depth of the tree, which we denote q. The number of ways to arrange n points into k classes in 1d is $k!\binom{n}{k-1}$ ($k!$ options for ordering the classes and then $\binom{n}{k-1}$ options for locating the thresholds). Since the degree of the tree is k for argmax queries, it must be that the $k^q \geq k!\binom{n}{k-1}$, which implies[3] a lower bound $q \geq O(\frac{k}{\log k} \cdot \log n)$.

Together, this concludes the proof of Theorem 5.2. □

[3]Using the fact that $\log\left(k!\binom{n}{k-1}\right) = \log k! + \log\binom{n}{k-1} = k \log k + \log\left(\left[\frac{n}{k}\right]^k\right) = k \log k + k(\log n - \log k) = k \log n$

Algorithm 3 `NbrGraphSGD`

Input: Label neighborhood graph G, buffer size R, steps T, confidence parameter τ, learning rate η, comparison oracle $A^{f^\star}$.
Output: classifier $h(\cdot; \boldsymbol{W})$, number of comparisons q.

Initialize $\boldsymbol{W}^{(0)}$, $L = 0$, $q = 0$, $b = 0$.
for $t = 1, 2, \ldots, T$ **do**
  Sample $\boldsymbol{x} \sim \mathcal{D}$.
  Sample (i, j) uniformly from the edges of G.
  if $\left|h_i(\boldsymbol{x}; \boldsymbol{W}^{(t-1)}) - h_j(\boldsymbol{x}; \boldsymbol{W}^{(t-1)})\right| < \tau$ **then**
    Obtain oracle comparison $c = 2(A^{f*}(\boldsymbol{x}, i, j) - 0.5)$

    $L \mathrel{+}= \log(1 + e^{-c(h_i(\boldsymbol{x};\boldsymbol{W}) - h_j(\boldsymbol{x};\boldsymbol{W}))})$.
    $q \mathrel{+}= 1$, $b \mathrel{+}= 1$.
  end if
  if $b \geq r$ **then**
    Update $\boldsymbol{W}^{(t)} \leftarrow \boldsymbol{W}^{(t-1)} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{W}}$
    Clear buffer: $L = 0$, $b = 0$.
  end if
end for

Our analysis suggests that when we can efficiently learn $G^\star$ and it is sparse, label-comparisons provide a gain over argmax queries. We showed this when $d = 1$, and it is natural to ask to what happens for $d > 1$. This requires addressing both the question of what is the binary active learning primitive that we use, as well as the questions of sparsity and learning the graph. See Supplementary A for discussion of these aspects.

6 A GENERAL PURPOSE ACTIVE LEARNING ALGORITHM

The approach of Algorithm 1 is to explicitly learn $\mathrm{e}(G)$ binary classifiers and aggregate them into a single classifier, that is not in $\mathcal{H}^{k,d}_{\text{lin}}$. For simplicity of optimization, we will prefer to work with models in $\mathcal{H}^{k,d}_{\text{lin}}$. To do so, in Algorithm 3 we present the `NbrGraphSGD` algorithm, a natural variation which can work directly with such models.

It works as follows: we first initialize a multiclass model $h(\cdot; \boldsymbol{W}) : \mathbb{R}^d \to \mathbb{R}^k$ (e.g. $\boldsymbol{W} \in \mathbb{R}^{k,d}$ for a linear model, but h can also be a neural network). For every data point $\boldsymbol{x}$, we sample an edge (i, j) in the graph G. This edge is a candidate label comparison. To decide whether to query it or not, we evaluate the difference in logits between labels i and j. If this difference is smaller than τ we query the pair (i, j) and add a binary cross entropy term that encourages the logit difference to have the correct sign. Once we accumulate sufficiently many comparisons, we perform an update step.

The remaining practical question is which graph G to use. Recall that $G^\star$ has an edge (i, j) iff $\exists \boldsymbol{x}$ where j was the 2nd best label and i was the argmax. For 1d, we showed this could be learned from data effectively. We leave the general case open, and consider here practical recipes for G. The simplest approach is to base G on prior knowledge regarding which classes are expected to be neighbors (e.g., via distances on their word embeddings, or other co-occurrence statistics). Another practical case is when first and second best labels are available without the $\boldsymbol{x}$ values (e.g., consider asking individuals what are their first and second most favorite products, without keeping user info). Note that in this case, we will receive evidence of edges only for $\boldsymbol{x}$ values sampled from $\mathcal{D}$. This corresponds to an empirical notion of the neighborhood graph, which we define below.

Definition 6.1 (Empirical Label Neighborhood graph)**.** *For a target function $f^\star : \mathbb{R}^d \to \mathbb{R}^k$, the neighborhood graph $G_{\mathcal{D}}(f)$ is an undirected graph on k vertices, where there is an edge between vertices i and j if and only if there exists $\boldsymbol{x} \in \mathbb{R}^d$ whose probability under $\mathcal{D}$ is non-zero, and for which for every $r \in [k]$, $f^\star(\boldsymbol{x})_i = f^\star(\boldsymbol{x})_j \geq f^\star(\boldsymbol{x})_r$.*

By definition, $G^\star_{\mathcal{D}} \subseteq G^\star$. One might hope that the discarded edges will not impact accuracy w.r.t $\mathcal{D}$. However, in the worst-case this is not true. Specifically, in proving Lemma 5.5 we used the fact that when B is given by the *true* binary classifiers (i.e. $h_{ij} = \boldsymbol{W}^\star{}_i - \boldsymbol{W}^\star{}_j$), the aggregated classifier $f^{(G^\star, C)}$ has perfect accuracy on $\mathcal{D}$. This may fail for $G^\star_{\mathcal{D}}$: $f^{(G^\star_{\mathcal{D}}, C)}$ may err on examples supported in $\mathcal{D}$; See Supplementary Figure 6 for an example. In Section 7 we observe that the performance of both graphs is comparable.

7 EXPERIMENTS

In this section we evaluate our label-comparisons algorithm `NbrGraphSGD` on synthetic as well as real data.

We consider the online active learning scenario. At each round $t \in [T] = \{1, ..., T\}$, the learner receives a batch of points drawn i.i.d. according to $\mathcal{D}$ and must decide which queries to request from the oracle $A^{f^\star}$ (including not requesting any queries). We compare the following methods:

- `NbrGraphSGD`(G): This is our algorithm which takes as input graph a G and, for a given $\boldsymbol{x}$, only considers labels pairs in G as possible pairs to query. For the given $\boldsymbol{x}$, we iterate over all $(i, j) \in G$. For each pair we check if $|W_i\boldsymbol{x} - W_j\boldsymbol{x}|$ is smaller than a fixed threshold. If it is, we query this pair. We consider different versions of `NbrGraphSGD`(G), that use different graphs.
- `PassiveTour`: This baseline uses label comparisons to simulate a standard argmax-based active learning algorithm [Joshi et al., 2009]. Namely, for each $\boldsymbol{x}$, we evaluate the logits $W_y\boldsymbol{x}$ and query $\boldsymbol{x}$ if the difference between the first and second best logits is below some threshold. In the standard argmax setting, we would have requested the label of $\boldsymbol{x}$. With label comparisons, we need to do this using $k - 1$ active comparisons. Namely, we perform a tournament between labels to reveal the maximizer.
- `ActiveTour`: It may seem wasteful to ask for $k - 1$ comparisons as above, since we may be sufficiently confident in some of these comparisons. We thus consider an "active tournament" algorithm: whenever the current model is sufficiently confident in a given pair in the tournament, we take the model's answer, and do not query for it.

Evaluation. In online active learning, the quality of an algorithm is measured by its accuracy after T rounds, and the total number of comparisons requested within these T rounds. We use a linear teacher model to simulate the comparison oracle (Definition 3.1), and measure accuracy as the categorical accuracy[4] on the test set, w.r.t the teacher's argmax. We use an *accumulating buffer* mechanism to control for the number of parameter updates across methods (each method accumulates the requested comparisons until the buffer is full, and only then performs a gradient update).

7.1 SYNTHETIC DATA

In this section we validate our theoretical findings from Sections 4 and 5 on synthetic data. Specifically, for $d \in \mathbb{N}$ and $\hat{k} \in \mathbb{N}$ we consider $\mathcal{D}$ to be the uniform distribution on a unit sphere in $\mathbb{R}^d$ and draw a random linear target model $\boldsymbol{W}^\star \in \mathbb{R}^{\hat{k}, d}$. This yields a multiclass classifier with $k \leq \hat{k}$ distinct decision regions ("effective classes"). We draw data from $\mathcal{D}$ and divide it into training and test sets.

Sparsity of the neighborhood graph. We begin by computing the sparsity level of both the true neighborhood graph $G^\star = G(\boldsymbol{W}^\star)$ and the empirical neighborhood graph $G^\star_{\mathcal{D}} = G_{\mathcal{D}}(\boldsymbol{W}^\star)$, where the latter is computed w.r.t the training set. We define the *sparsity level* as the number of edges in G, divided by $\binom{k}{2}$ (i.e,. the number of edges in a complete graph). In Figure 3 we plot the sparsity level as a function of k and d, as averaged over 25 random target models. We see that the empirical sparsity level tracks the true sparsity level, and that for a fixed dimension d, both decrease with the number of effective classes k. This confirms that we expect the sparsity to "kick in" when $k \gg d$.

Comparisons of Active Learning Methods. We next compare the different baselines described above. For `NbrGraphSGD`(G) we consider multiple variations, that use different versions of the graph G. In Figure 4 we report the performance of `NbrGraphSGD` relative to several natural baselines. First, it can be seen that the active tournament outperforms the passive one, suggesting that indeed some tournament queries can be avoided. Yet `NbrGraphSGD` outperforms the tournament baselines, indicating that tourna-

[4]Specifically, use Top-K accuracy, where $K = 0.1 \cdot k$.

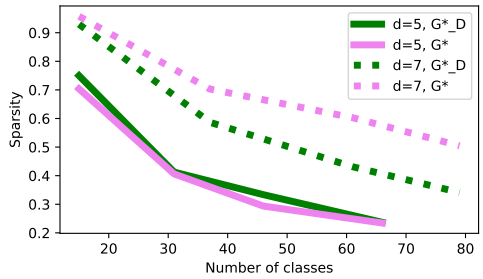

Figure 3: Sparsity level for a random linear model as a function of the number of effective classes k for $d = 5, 7$

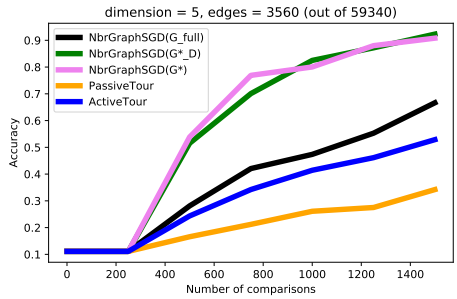

Figure 4: Comparing algorithm `NbrGraphSGD` w.r.t $G^\star_{\mathcal{D}}$ (green) and $G^\star$ (purple) against three baselines: passive tournament (yellow), active tournament (blue) and algorithm `NbrGraphSGD` with respect to a complete graph (black).

ment comparisons are generally not the optimal approach. Within the `NbrGraphSGD` methods, using the true graph (either G^* or G^*_D) provides the best performance, indicating that the graph plays an important role in active learning efficacy, and that `NbrGraphSGD` can use this structure.

7.2 REAL DATA

The QuickDraw dataset [Ha and Eck, 2017], is a collection of 50 million drawings across 345 categories, contributed by players of the game "Quick, Draw!". We use the bitmap version of the dataset, which contains these drawings converted from vector format (keystrokes) into 28x28 grayscale images. We randomly select $70,000$ examples from this large data and use $60,000$ as our training set and the rest as the test set. We train a linear teacher on the data after randomly projecting it into $\mathbb{R}^d$. We then use the resulting model $\boldsymbol{W}^\star$ to implement the label-comparison oracle (see Definition 3.1). We denote the true graph of $\boldsymbol{W}^\star$ (Definition 5.3) as $G^\star$ and the empirical graph of $\boldsymbol{W}^\star$ (Definition 6.1, as computed w.r.t the training set) as $G^\star_{\mathcal{D}}$.

We begin by comparing the performance of `NbrGraphSGD` w.r.t $G^\star_{\mathcal{D}}$ with the same baselines from Section 7.1. We explore the relationship between the sparsity of the *true graph* $G^\star$ and the performance of `NbrGraphSGD` w.r.t $G^\star_{\mathcal{D}}$

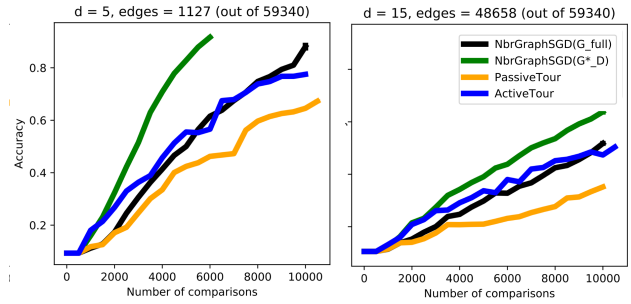

Figure 5: Comparing the performance of `NbrGraphSGD` w.r.t $G^\star_{\mathcal{D}}$ (green) against the baselines on the QuickDraw dataset. In the plot titles, d denotes the dimension of low-dimensional projection of the data and edges is $e(G^\star)$, the number of edges in the true neighborhood graph of $\boldsymbol{W}^\star$.

as a function of the dimension d and $k = 345$. In Figure 5 we report the query complexities w.r.t $d = 5$ (left) and $d = 15$ (right). Note that this is a realizable learning task since the models are measured in terms of their accuracy w.r.t the teacher's predictions, and the teacher is also a linear model. In line with our theoretical results from Section 5, we observe that the gain from using our method (over e.g. the passive or active tournament baselines) is smaller when the true neighborhood graph is less sparse.

8 CONCLUSIONS

We studied the setting where annotators are asked to provide only pairwise label comparisons. We believe this is a natural setting as it is both easy for humans to provide, and still results in sufficient information for learning. Our results provide several key characterizations of how this information should be gathered and used. We show that, perhaps counter-intuitively, there are cases for which having all the class comparisons per training point does not yield a sample complexity advantage over just receiving the one true class label. On the other hand, in the active setting, we show that comparisons can be used in an effective way that goes beyond obtaining the argmax training labels.

Many interesting open questions remain. First, our focus was on linear classification, and it would be interesting to generalize the result to other classes (such as neural networks). Second, one can consider a mixture of comparisons and true-labels, since the latter may be easy to obtain in some instances, and hence query-complexity should count these cases differently. Finally, here we assumed that annotators can provide answers to all queries. In practice, some queries may not be answerable (e.g., labels are too "close" or both are equally bad), and it would be interesting to extend the formalism and practical algorithm to these cases.

References

Franz Aurenhammer and Rolf Klein. Voronoi diagrams. *Handbook of computational geometry*, 5(10):201–290, 2000.

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.

Omri Ben-Eliezer, Max Hopkins, Chutong Yang, and Hantao Yu. Active learning polynomial threshold functions. *arXiv preprint arXiv:2201.09433*, 2022.

Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. *Machine learning*, 90(3):347–383, 2013.

Amit Daniely and Shai Shalev-Shwartz. Optimal learners for multiclass problems. In *Conference on Learning Theory*, pages 287–316. PMLR, 2014.

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

Constantinos Daskalakis, Richard M Karp, Elchanan Mossel, Samantha J Riesenfeld, and Elad Verbin. Sorting and selection in posets. *SIAM Journal on Computing*, 40(3): 597–622, 2011.

Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *Conference on Learning Theory*, pages 563–587. PMLR, 2015.

Rex A Dwyer. Higher-dimensional voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(3):343–367, 1991.

Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *International conference on machine learning*, pages 1427–1436. PMLR, 2018.

David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. URL `http://arxiv.org/abs/1704.03477`.

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, 2020.

Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, pages 2372–2379. IEEE, 2009.

Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pages 440–447, 2008.

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.

Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Preprint*, 2022.

Aadirupa Saha, Tomer Koren, and Yishay Mansour. Adversarial dueling bandits. In *International Conference on Machine Learning*, pages 9235–9244. PMLR, 2021.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

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.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

A ACTIVE LEARNING COMPLEXITY FOR $d > 1$

Our analysis in the main text suggested that when we can efficiently learn $G^\star$ and it is sparse, label-comparisons provide a gain over argmax queries. We showed this when $d = 1$, and it is natural to ask to what happens for $d > 1$. This requires addressing both the question of what is the binary active learning primitive that we use, as well as the questions of sparsity and learning the graph. We discuss these two points below.

Regarding binary active learning, for $d > 1$, active learning is known to no longer provide asymptotic benefits over passive learning in the distribution-free setting [Dasgupta, 2004]. Hence for $d > 1$ it's natural to focus on *distribution-specific* learning. Namely, we will understand $q_b(\cdot)$ to be the query complexity of active learning the binary class $\mathcal{H}_{\text{lin}}^{2,d}$ under structural assumptions on the target distribution. For example, Balcan and Long [2013] prove that for log-concave distributions, $q_b(\gamma) = \Theta(d \log \frac{1}{\gamma})$.

Regarding sparsity, we note that linear classifiers in $d = 1$ are maximally spase. Interestingly, linear classifiers in $d = 2$ also admit sparse graphs. By definition, the decision regions of $h(\cdot; \boldsymbol{W}) \in \mathcal{H}_{\text{lin}}^{k,3}$ are convex polyhedrons. By Steinitz's theorem, these are 3-connected planar graphs, and so a corollary of Euler's Formula implies that $\mathrm{e}(G) \leq 3k - 6$. In Section 7.1 we experimentally evaluate the sparsity of random linear teachers as a function of k, d, observing that the graphs are becoming sparse as $k \gg d$. This is related to a known result in computational geometry Dwyer [1991], that the expected number of edges in a Voronoi regions is linear, when points are sampled uniformly from the sphere (see also Aurenhammer and Klein [2000]). We leave open the question of whether the graph learning procedure of Algorithm 2 has an efficient analogue in $d > 1$.

B DEMONSTRATION OF G_D^* MISSES

In the main text, we discuss the empirical neighborhood graph G_D^*, which is a subset of the true graph G^*. In Figure 6 below we show an example where G_D^* will result in erroneous classification, because it will miss important boundaries points.

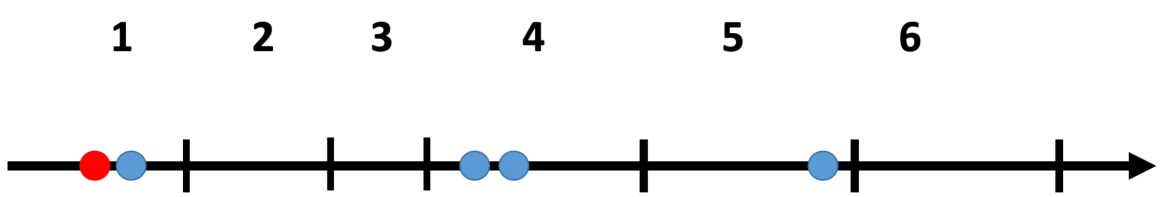

Figure 6: For a linear classifier in $d = 1$ and distribution on $\mathbb{R}$ given by the blue circles, the empirical graph $G_{\mathcal{D}}^\star$ only has the edges $(1, 2), (3, 4)$ and $(5, 6)$. Thus the aggregated classifier $f^{(G_{\mathcal{D}}^\star, C^\star)}$ may incorrectly classify the red circle as belonging to class 3 since it also "beats" all its' neighbor classes. Here, the culprit is that $G_{\mathcal{D}}^\star$ misses the edge $(2, 3) \in G^\star$, as it was not witnessed by x in the support of $\mathcal{D}$.

C PROOF OF LEMMA 5.5

Fix $\mathcal{D}$ and $\boldsymbol{W}^\star$. We will use an important observation: if $f^\star$ is $f^{(G,C)}$ w.r.t $G = G(\boldsymbol{W}^\star)$ and C given by the *true* binary classifiers (i.e. $h_{ij} = \boldsymbol{W}^\star{}_i - \boldsymbol{W}^\star{}_j = h_{ij}^\star$), then $L_{\mathcal{D}}(f^\star) = 0$. I.e, for every $\boldsymbol{x} \in \mathbb{R}^d$ with argmax $y = \arg\max_{i \in [k]} \boldsymbol{W}^\star{}_i \boldsymbol{x}$, the following holds: $\forall j : \quad 1 = f_y^\star(\boldsymbol{x}) > f_j^\star(\boldsymbol{x})$.

Now, fix (G, C) satisfying the conditions in the theorem statement. We can upper bound $L(f^{(G,C)})$ as follows:

$$\begin{aligned}
L(f^{(G,C)}) &= \mathop{\mathbf{E}}_{x\sim\mathcal{D}}[f^{(G,C)}(\boldsymbol{x}) \neq \arg\max_{i\in[k]} \boldsymbol{W}^{\star}{}_{i}\boldsymbol{x}] \\
&= \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[\exists j \in [k] : \ f^{(G,C)}(\boldsymbol{x})_y < f^{(G,C)}(\boldsymbol{x})_j] \\
&\leq \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[\exists (i,j) \in G : \ \mathrm{sign}(h_{ij}(\boldsymbol{x})) \neq \mathrm{sign}(h^{\star}_{ij}(\boldsymbol{x}))] \\
&\leq \sum_{(i,j)\in G} \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[\mathrm{sign}(h_{ij}(\boldsymbol{x})) \neq \mathrm{sign}(h^{\star}_{ij}(\boldsymbol{x})] \\
&\leq \sum_{(i,j)\in G} \epsilon/\mathrm{e}(G) \leq \varepsilon
\end{aligned}$$

As required.