A Unified Analysis of Dynamic Interactive Learning

Xing Gao, Thomas Maranzatto, Lev Reyzin

Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
{xgao53,tmaran2,lreyzin}@uic.edu

***Abstract*—In this paper we investigate the problem of learning evolving concepts over a combinatorial structure. Previous work by Emamjomeh-Zadeh et al. [1] introduced dynamics into interactive learning as a way to model non-static user preferences in clustering problems or recommender systems. We provide many useful contributions to this problem. First, we give a framework that captures the two different models analyzed by Emamjomeh-Zadeh et al. [1], which allows us to study any type of concept evolution and matches the same mistake bounds and running time guarantees of the previous models. Using this general model we solve the open problem of closing the gap between the upper and lower bounds on the number of mistakes. Finally, we study an efficient algorithm where the learner simply follows the feedback at each round, and we provide mistake bounds for low diameter graphs such as cliques, stars, and general $o(\log n)$ diameter graphs by using a Markov Chain model.**

I. Introduction

The problem of recommending products or media is ubiquitous in many practical settings such as search engines, online marketplaces, or media streaming services (e.g. Google search, Amazon, Spotify, etc.). In such settings any algorithm that tries to optimize recommendations receives implicit feedback from users in the system. This feedback is then used to refine future queries.

Drawing inspiration from earlier work on query learning by Angluin [2], as well as more recent models for interactive clustering [3]–[5], Emamjomeh-Zadeh and Kempe [6] considered such product recommendation problems from the perspective of combinatorial learning, where specific orderings of recommendations are nodes in a (very large) digraph. In this graph there is a distinguished node (the target) that corresponds to the ordering that the learner wishes to discover. This can be thought of as the 'ideal' ordering of products in a marketplace, or the 'best' sequence of recommendations in a streaming service. A directed edge exists from node s to s' if the user might prefer s' when the learner proposes s. For example, a user might want to swap two items from an ordered list s to get the more preferred ordering s'. If the learner proposes a node and that is not the target, it receives noisy (random [6] or even adversarial [7]) feedback in the form of an edge on the shortest path from the proposed node and the target. This form of feedback is similar to the correction queries from query learning [8].

Emamjomeh-Zadeh et al. [1] subsequently considered cases when the combinatorial structure itself can evolve over time – as they noted, some of these settings resembled earlier work on shifting bandits [9]. These dynamic settings are also where our results lie, and among our results, we generalize the work of Emamjomeh-Zadeh et al. [1] and solve some of their open problems herein.

Supported in part by NSF awards ECCS-2217023 and CCF-2307106.

II. Preliminaries

Emamjomeh-Zadeh and Kempe [6] first introduced a static graph model for robust interactive learning, where there is one fixed concept (the target) in the concept class, and the learner is trying to learn under noisy feedback. Later Emamjomeh-Zadeh et al. [1] extended the model to dynamic interactive learning, where the target concept can change during learning. Our work is based on the same framework, so we will briefly describe previously defined models and results here.

A. Static model

For clarity we first state the static learning model from Emamjomeh-Zadeh and Kempe [6], as it is a foundation for later work on dynamic models.

Definition 1 (Feedback graph [6])**.** *Define a weighted (directed or undirected) graph $G = (V, E, w)$, where the vertices represent a set of $n = |V|$ candidate concepts. The edge set E captures all possible corrections a learner can receive: edge (s, s') exists if the user is allowed to propose s' in response to s. The edge weights w are given to the learning algorithm, satisfying a key property: if the learner proposes s and the ground truth (target) is $t \neq s$, then every correct user feedback s' lies on a shortest path from s to t with respect to edge weight w.*

Note that we assume that the weighted graph is given to the algorithm and faithfully represents the underlying problem.

For an undirected graph G, let $N_G(v)$ denote the neighborhood of v in G. For a directed graph G, let $N_D^{in}(v)$ be the in-neighborhood of v in digraph D (including self loops) and $N_D^{out}(v)$ be the out-neighborhood of v in digraph D (including self loops).

In the static model there exists a fixed target vertex $t \in V$ that the algorithm is attempting to learn over multiple rounds. In each round the learner proposes a query vertex $q \in V$ and receives a feedback vertex z which is noisy with probability p. Specifically, if $q = t$, with probability $1-p$ the learner receives feedback q indicating the query is correct, and with probability p it receives an incorrect feedback z which is adversarially chosen from $N_G(q)$; if $q \neq t$ the algorithm is given a feedback $z \in N_G(q)$ which is incorrect with probability p. Crucially both correct and incorrect feedbacks are adversarial. As discussed in Emamjomeh-Zadeh and Kempe [6] this implies learning is only feasible when $p < 1/2$.

Other important definitions used throughout include the collection of concepts that are consistent with a particular feedback, or the version space for a query-feedback pair, as well as the weighted median of the feedback graph, which can be interpreted as the 'center of mass' of the graph.

Definition 2 (Version space [6]). *If the learner proposes q and receives feedback z, let $S_G(q, z)$ be the collection of concepts (nodes) that are consistent with the feedback. Formally, $S_G(q, z) = \{v \mid z$ lies on a shortest weighted path from q to $v\}$.*

Definition 3 (Weighted median [10]). *Let $L : V \to \mathbb{R}^{\geq 0}$ be a function that assigns likelihood to every vertex in the feedback graph $G = (V, E, w)$. A weighted median u is a vertex that minimizes $\sum_{v \in V} L(v) \cdot w(u, v)$.*

Emamjomeh-Zadeh and Kempe [6] presented a multiplicative weight update algorithm, which assigns likelihoods for each vertex in the feedback graph, and repeatedly queries the weighted median, which has the property of halving the total likelihood of its version space each round.

B. Dynamic model

Our paper is concerned with dynamic interactive learning where the target t is allowed to move. Assume that over the R rounds of learning the target moves at most B times and at round r the target is located at some node t_r. Without further assumption on target evolution, Emamjomeh-Zadeh and Kempe [1] showed the following general mistake upper bound. For the remainder of the paper, denote the entropy as $H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$.

Theorem 4 (Emamjomeh-Zadeh et al. [1]). *Assume the total number of rounds R is known beforehand. Let $A = V^R$ be the set of all node sequences of length R and let $a^* = \langle t_1, \ldots, t_R \rangle$ be the sequence of true targets throughout the R rounds. Let $\lambda : A \to \mathbb{R}^{\geq 0}$ be a function that assigns non-negative weights (likelihoods) to these sequences, such that $\sum_{a \in A} \lambda(a) \leq 1$. There is an online learning algorithm that makes at most $\frac{1}{1-\mathrm{H}(p)} \cdot \log \frac{1}{\lambda(a^*)}$ mistakes in expectation.*

While this is a positive result for the mistake bound, it does not guarantee an efficient algorithm as it has to keep track of weights for all sequences V^R, and in the worst case the number of sequences is $O(n^R)$ where $n = |V|$. Relatively efficient implementations exist for the following two models without explicitly constructing the λ map for each sequence.

1) Shifting Target: In the Shifting Target model there exists an unknown subset of vertices $S \subseteq V$ where $|S| \leq k$, and the learner knows k. Target transition is restricted within S, viz. $t_r \in S$ for every round r. Previous work by Emamjomeh-Zadeh et al. [1] proved the following theorem.

Theorem 5 ([1]). *Under the Shifting Target model, there is a deterministic algorithm that runs in time $O(n^k poly(n))$ and makes at most $\frac{1}{1-\mathrm{H}(p)} \cdot (k \log n + (B+1) \log k + R \cdot \mathrm{H}(B/R))$ mistakes in expectation. Furthermore, there exists a graph such that every algorithm makes at least*

$$\min \Big\{ \frac{1}{1-\mathrm{H}(p)} \cdot [k \log n + (B - 2k + 1) \cdot (\log k)] - o(\log n) - B \cdot o(\log k)\,,\; R - o(R) \Big\}$$

mistakes in expectation.

Note in particular the exponential dependence on k in the runtime of the algorithm.

2) Drifting Target: In the Drifting Target model, it is assumed that the target slowly evolves over time, and the evolution can be modeled by following the edges of some transition graph defined below.

Definition 6 (Transition graph [1]). *There exists a known unweighted digraph $G' = (V, E')$, where in particular G' and G have the same vertex set but their edge sets can be different. The target is only allowed to move along edges of G'. Formally for every round r the model requires $t_{r+1} \in \{t_r\} \cup N^{out}_{G'}(t_r)$. Let Δ be the maximum degree in G'.*

In previous work, Emamjomeh-Zadeh et al. [1] proved the following theorem.

Theorem 7 (Emamjomeh-Zadeh et al. [1]). *Under the Drifting Target model, there is a deterministic algorithm that runs in time $poly(n)$ and makes at most $\frac{1}{1-\mathrm{H}(p)} \cdot (\log n + B \cdot \log \Delta + R \cdot \mathrm{H}(B/R))$ mistakes in expectation..*

Furthermore, there exists a graph such that every algorithm makes at least

$$\min \Big\{ R - o(R), \frac{1}{1-\mathrm{H}(p)} (\log n + B \log \Delta) - o(\log n) - B \cdot o(\log \Delta) \Big\}$$

mistakes in expectation.

Notice that for both Shifting Target and Drifting Target models, there is a gap of $R \cdot H(B/R)$ between the mistake upper bound and lower bound, which remains an open problem in Emamjomeh-Zadeh et al. [1]. We will show a new lower bound and close the gap in Section 4.

III. A UNIFIED MODEL

Our first contribution is to define a more generalized model inspired by the original Drifting Target model, and show that the results from Theorems 5 and 7 are both valid

Algorithm 1 Interactive learning likelihood update

1: Initialize $L'_1(u)\ \forall u \in V'$
2: **for** $1 \le r \le R$ **do**
3: $\forall i \in V : L_r(i) \leftarrow \sum_{u \in V'_i} L'_r(u)$ {Aggregate L_r from L'_r over the duplicates}
4: $q_r \leftarrow \arg\min_{i \in V} \sum_{j \in V} L_r(j) \cdot w(i,j)$ {Query the weighted median}
5: $z_r \leftarrow$ feedback from adversary
6: $\forall i \in V$ and $\forall u \in V'_i$:
7: $P(i) \leftarrow (1-p) \cdot \mathbb{1}[i \in S_G(q_r, z_r)] + p \cdot \mathbb{1}[i \notin S_G(q_r, z_r)]$ {Weight update multiplier for i}
8: $P(u) \leftarrow P(i)$ {Same weight update for each duplicate u of i}
9: $\forall u \in V' : L'_{r+1}(u) = \sum_{v \in N^{in}_{G'}(u)} P(v) \cdot L'_r(v) \cdot \pi_{vu}$ {Apply weight update and transition}
10: **end for**

under this generalization, thus unifying previous models. The technique used in Emamjomeh-Zadeh et al. [1] to efficiently implement the Shifting Target model was to keep track of likelihoods for each subset of nodes instead of each sequence, reducing computational complexity from $O(n^R)$ to $O(n^k)$. Similarly, their efficient implementation for the Drifting Target model keeps track of likelihoods for each node, reducing computational complexity to $O(poly(n))$. This method requires customization for each transition model, and can become tricky as the models become more complicated. A key motivation for a general model is that it unifies a wide class of transition models, and allows us to easily obtain mistake upper bounds and runtime guarantees based on a single algorithm.

As above let $G = (V, E, w)$ be the graph representing the candidate concepts (the feedback graph), and $G' = (V', E', \pi)$ be a directed transition graph, representing all possible ways the target might change over time. The key difference is that for each vertex $i \in V$, V' contains possibly duplicated vertices corresponding to the same vertex i, denoted by $V'_i := \{u \in V' \mid u \text{ corresponds to } i \in V\}$. Define $n' = |V'|$, Δ' as the max degree of G', and π_{ij} as the transition probability in G', where $\pi_{ii} = (1-b)$ with $b = \frac{B}{R}$, and $\pi_{ij} = \pi_{\text{out}} := \frac{b}{\Delta'}$ for $i \neq j$ under a uniform transition assumption.

We present a modified version of the algorithm from Emamjomeh-Zadeh et al. [1]. In the r^{th} round, we keep track of the likelihoods for each vertex $u \in V'$ as $L'_r(u)$, and for each vertex $i \in V$ its likelihood $L_r(i)$ is aggregated from L'_r as the summation over all of i's duplicates in V'. The median of the feedback graph G is then calculated based on L_r. We update the likelihoods L'_{r+1} for all corresponding nodes in G' based on each node's consistency with the feedback using the same rules as in Emamjomeh-Zadeh et al. [1].

Theorem 8. *Assuming the first target is chosen uniformly at random from V', Algorithm 1 runs in time $O(\Delta' \cdot n' + poly(n))$, uses space $O(n')$, and makes at most*

$$\frac{1}{1 - H(1-p)} \cdot \Big(\log n' + B \cdot \log \Delta' + R \cdot H(B/R) \Big)$$

mistakes in expectation.

Alternatively writing the bound using transition probabilities instead of maximum degree, Algorithm 1 runs in time $O\left(\frac{1}{n'} \cdot \pi^B_{out} \cdot (1-b)^{R-B}\right)$, uses space $O(n')$, and makes at most

$$\frac{1}{1 - H(1-p)} \cdot \Big(\log n' + B \cdot \log(b/\pi_{out}) + R \cdot H(b) \Big)$$

mistakes in expectation.

Proof. We wish to show that the likelihood of the ground truth sequence a^* is at least

$$\lambda(a^*) = \frac{1}{n' \cdot \Delta'^B \cdot \binom{R}{B}} ,$$

or alternatively

$$\lambda(a^*) = \frac{1}{n'} \cdot \pi^B_{\text{out}} \cdot (1-b)^{R-B} .$$

Note that these two expressions correspond to two equivalent interpretations of the transition model: 1, the target changes at most B times during R rounds; 2, the target changes with probability at most $b = B/R$ at each round.

For the first interpretation, we provide the expression for $\lambda(a^*)$ with a combinatorial argument: there are n' choices for the first node, and the next node differs from the previous node at most B times, each time with Δ' choices, and these changes can occur at $\binom{R}{B}$ locations in the sequence. Thus the total number of valid sequences is $n' \cdot \Delta'^B \cdot \binom{R}{B}$. The initial likelihoods are assigned uniformly among all sequences, so dividing 1 by the total number of sequences gives us $\lambda(a^*)$.

For the second interpretation, we have a probabilistic argument: the sequence starts with any particular node in the transition graph with probability $\frac{1}{n'}$, and the next node changes to one of the neighbors with probability $\pi_{\text{out}} = \frac{b}{\Delta'}$ for B times, and stays the same with probability $1-b$ for $R-B$ times. Taking the product of these probabilities gives the result.

The mistake upper bound follows by substituting $\lambda(a^*)$ into Theorem 5 of Emamjomeh-Zadeh et al. [1]. Note that for large R we approximate $\binom{R}{B}$ by $\left(\frac{R}{B}\right)^B \cdot \left(\frac{R}{R-B}\right)^{(R-B)}$, which contributes to the term $R \cdot H(B/R) = R \log R - B \log B - (R-B)\log(R-B)$ after taking logarithm.

For algorithmic complexity, steps 3 (aggregating likelihoods) and 6 (computing weight multiplier) both take time $O(n')$, step 4 (computing median) takes time $O(n^3)$, and step 9 (updating weight and transition) takes time $O(\Delta' \cdot n')$, and L_r, L'_r takes space n and n' respectively. □

Under our generalized model, any dynamic interactive learning problem can be reduced to defining the feedback graph G to represent the concept class, and defining the transition graph G' to represent the concept evolution. Specifically, the Shifting Target model and Drifting Target model studied in the original paper can be shown as special cases under this general model, and we will show that the general bounds agree with the original results.

Corollary 9. *In the Drifting Target model, Algorithm 1 runs in time $O(\Delta \cdot poly(n))$, uses space $O(n)$, and makes at most*

$$\frac{1}{1-H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta + R \cdot H(B/R) \Big)$$

mistakes in expectation.

Proof. The transition graph G' is the same as the feedback graph G, and transition probability is assumed to be uniform. Thus $n' = n$, and $\Delta' = \Delta$. Plugging into Theorem 8 gives the result. □

Corollary 10. *In the Shifting Target model, Algorithm 1 runs in time $O(k^2 \cdot n^k)$, uses space $O(k \cdot n^k)$, and makes at most*

$$\frac{1}{1-H(1-p)} \cdot \Big(k \cdot \log n + (B+1) \cdot \log k + R \cdot H(B/R) \Big)$$

mistakes in expectation.

Proof. The transition graph G' consists of $\binom{n}{k}$ disconnected sub-graphs, where each sub-graph is a clique of size k, corresponding to a subset of k vertices in V. Each round the target might shift within a k-clique, and each clique represents a possible choice of the k-subset of targets. Thus $n' = \binom{n}{k} \cdot k$ and $\Delta' = k$. Plugging this into Theorem 8 gives the result. □

Since the mistake and computational upper bounds mostly depend on the size of transition graph, namely n' and Δ', minimality of the transition graph is crucial for query and computational efficiency. We want to find the worst case mistake upper bound, which can be used as a benchmark when modeling various types of transitions. A trivial upper bound on mistakes occurs in the case that the learner does not have any information about how target might change over time, thus the transition graph G' is a complete graph on $n' = n$ vertices, and $\Delta' = n$. Plugging into Theorem 8 gives the following result.

Corollary 11. *The worst case mistake upper bound using Algorithm 1 is*

$$\frac{1}{1-H(1-p)} \cdot \Big((B+1) \cdot \log n + R \cdot H(B/R) \Big) \ ,$$

and runs in time $O(poly(n))$ and space $O(n)$.

In the following sections, we will discuss a few examples of other transition models, showing a hierarchy of mistake bounds.

A. Shortest path

Given two vertices $s, t \in G$, define $S_G^B(s,t)$ to be the collection of all subsets of $S_G(s,t)$ that contain at most B vertices. Formally, $S_G^B(s,t) = \{H \subseteq S_G(s,t) : |H| \leq B\}$. In the Shortest Path model we insist the target can only move along a shortest path in G.

We can describe this model in the language of our generalized framework. The transition graph G' consists of many disconnected directed paths, each corresponding to some element of $S_G^B(s,t)$ for some $s, t \in G$. This procedure over counts, so we also restrict G' to only include one copy of any subset of vertices in a path in G. Finally the vertices in any subgraph of G' are connected with $B-1$ arcs that correspond to the ordering imposed by traversing $S_G(s,t)$ from s to t.

The number of vertices in G' is bounded as $n' \leq B \cdot \binom{n}{B}$. We can't hope to do better than this, as there are classes of graphs with exponentially many shortest paths between two distinguished vertices. The maximum degree of G' is 2, as all disconnected components are paths.

This model is a variation of the Shifting Target model. If the target can move B times, then the target can only move in one direction in each valid path. We can still apply Thm. 8 and get a naive mistake upper bound that runs in time $n^B \cdot poly(n)$. We can achieve the $\binom{n}{B}$ bound on the number of subsets when G is a path with n vertices. However, the target can only move in one direction along the path, so it's natural to think a better algorithm can be developed at least for this case.

Corollary 12. *In the Shortest-path model, Algorithm 1 runs in time $O(n^B)$, uses space $O(B \cdot n^B)$, and makes at most*

$$\frac{1}{1-H(p)} \cdot (B \cdot \log n + (B+1) \cdot \log B + R \cdot H(B/R))$$

mistakes in expectation.

B. m-Neighborhood

Let $N_G^m(v)$ denote the set of vertices in G that have a shortest path of length m to v. In the m-Neighborhood model, the target can move within N_G^m, and m is known to the learner. This model is a variation of the Drifting Target model, and note that $m = 1$ is exactly the case when $G = G'$ in the original Drifting Target model. The transition graph G' is constructed by including an arc from every $v \in V$ to every node in its m-Neighborhood. Note that $n' = n$ and $\Delta' \leq \Delta^m$. Applying Theorem 8 gives the following mistake bound for the m-Neighborhood model:

Corollary 13. *In the m-Neighborhood model, Algorithm 1 runs in time $O(\Delta^m \cdot n)$, uses space $O(n)$, and makes at most $\frac{1}{1-H(p)} \cdot (\log n + B \cdot m \cdot \log(\Delta) + R \cdot H(B/R))$ mistakes in expectation.*

To complete our hierarchy, in descending mistake upper bounds we have: Shortest Path model, the original Shifting Target model, the m-Neighborhood model, and the original Drifting Target model.

IV. Query complexity lower bound

In this section we close the gap between upper and lower bounds on the nubmer of mistakes, which remained an open problem in Emamjomeh-Zadeh et al. [1]. We show a mistake lower bound that matches the upper bound asymptotically.

Our result requires some background on the noisy binary search problem. Here there is a distinguished integer t from the set $\{1, ..., m\}$. In each round r, the learner queries some integer x. If $x = t$ then the item has been found and the procedure stops. Otherwise with probability $1-p$ the learner receives correct feedback of the form $x > t$ or $x < t$. We make use of the following lower bound.

Theorem 14 (Emamjomeh-Zadeh et al. [1]). *Every algorithm for the noisy binary search problem requires at least* $\frac{\log m}{1-H(p)} - o(\log m)$ *queries in expectation.*

The idea of our proof is to establish a reduction from noisy binary search: given a noisy binary search problem where the target is uniformly random among m items, we will reduce it to a specific Drifting Target problem under our dynamic interactive learning model. Thus the lower bound on noisy binary search is also a lower bound on interactive learning.

Theorem 15. *For every n and Δ', there exists a Drifting Target problem such that every algorithm makes at least*

$$\frac{1}{1-H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta' + R \cdot H(B/R) \Big) - o\left(\log n + B \cdot \log \Delta' + R \cdot H(B/R) \right)$$

mistakes in expectation.

Proof. We reduce a noisy binary search problem over m items to an interactive learning problem defined in the following way: choose n and R such that $m \leq n^R$ so that each of the m items can be represented as a base n encoding/enumeration of a sequence with R digits. The feedback graph G for the learning problem is a simple path on n vertices, ordered in the same way as in the encoding: the left end is the smallest digit while the right end is the largest. The transition graph G' is defined on the same set of vertices as in G so $n' = n$. G' includes all the edges in G as bi-directional edges, potentially with additional edges up to some degree Δ'.

For example, to search among $m \leq 1000$ items, we can choose $n = 10$ and $R = 3$, so that each item can be encoded by a length-3 sequence between $\langle 0, 0, 0 \rangle$ and $\langle 9, 9, 9 \rangle$, which are our familiar base-10 natural numbers up to 999. The graph G consists of vertices $0, 1, 2, \ldots, 9$ on a path, and interactive learning continues for 3 rounds. Suppose the target item is encoded as the sequence $\langle 1, 1, 2 \rangle$, then the ground truth target locations during the 3 rounds are vertices $1, 1, 2$ respectively (target shifted once from vertex 1 to 2).

In the r^{th} round of the interaction, the learner queries vertex $i \in [n]$, which can be interpreted as guessing the r^{th} digit of the target's encoding sequence. Without loss of generality, suppose the adversary's feedback is some vertex j to the left of vertex i, which is interpreted as the r^{th} digit of the target sequence is less than the value i. The learner updates likelihoods for sequences whose r^{th} digit is less than i by a factor of $1-p > \frac{1}{2}$, and the other sequences by a factor of $p < \frac{1}{2}$. According to Lemma 6 from [1], the likelihood of the target item's encoding sequence (ground truth) decreases exponentially more slowly than the rest of the sequences and will eventually prevail.

To establish the lower bound for a Drifting Target problem with arbitrary n and Δ', there exists a noisy binary search problem on $m = n \cdot \Delta'^B \cdot \binom{R}{B}$ items that reduces to the Drifting Target problem. The encoding is restricted such that after the first digit is chosen, the remaining digits can change at most B times among Δ' choices (all other sequences are initialized with 0 likelihoods). Plugging in our value of m into Theorem 14 gives a mistake lower bound of $\frac{1}{1-H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta' + R \cdot H(B/R) \Big) - o(\log n + B \cdot \log \Delta' + R \cdot H(B/R))$, as desired. □

V. Efficient algorithm for low diameter graphs

While Algorithm 1 emphasizes on bounding the number of mistakes for general interactive learning problems, its computation can be inefficient in each round and deteriorates as the transition model becomes more complex. We realize that the computational complexity mainly comes from keeping track of the likelihoods under all possible transitions, so we consider an alternative approach where the learner ignores the transition model completely and simply follows the adversary's feedback each round. After the initial query, the algorithm requires no computation. In this section, we study this simple algorithm's performance on low diameter graphs. We formally present this algorithm below.

Algorithm 2 'Follow the Feedback' Procedure for Interactive Learning

$q_1 \leftarrow \text{argmin}_{i \in V} \sum_{j \in V} w(i, j)$ {Start with a 'center' vertex}
for $1 \leq r \leq R$ **do**
  $z_r \leftarrow$ feedback from adversary after querying q_r
  $q_{r+1} \leftarrow z_r$ {Follow the feedback for next round}
end for

A. Cliques: graphs with diameter 1

A clique is the most symmetric graph, where each vertex can be considered the center, and the graph has diameter 1. This means no matter which node the learner queries, a correct feedback from the adversary will reveal the true target at each round. Therefore after each mistake, the learner will keep querying the correct node until the target's next move. The mistake upper bound is stated in the theorem below.

Theorem 16. *If the feedback graph G' is fully-connected (a clique on the concept class), Algorithm 2 makes at most $B + p(R - B)$ mistakes in expectation.*

Proof. By assumption the learner queries a node then receives a feedback, and the target may move at any point during this process. To help with the analysis in this case, we break down the chain of events in the following way: in each round we assume that at first the target moves (or stays put), then the learner makes a query and receives a new feedback. Notice that in every round where the target moves the learner will make a mistake regardless of the correctness of the previous feedback. If we assume target can move at most B times, this leads to B mistakes. For the $R - B$ rounds where the target doesn't move, the learner makes a mistake if and only if the previous feedback is incorrect. As feedback is noisy with probability p, this leads to $p(R - B)$ mistakes. So the expected number of mistakes over the course of R rounds is:

$$E[M] = B + p(R - B)$$

In anticipation of discussing other classes of graphs we present a second analysis of Algorithm 2 on cliques. Assume that each round the target moves with probability $b = \frac{B}{R}$. We can model the process as a Markov Chain where the states $\{0, 1\}$ represent the learner's distance from the target at each round. Note that these states do not represent the learner's position in the graph. Now we break down the chain of events in a slightly different way: first, the learner queries the node received from previous feedback and receives a new feedback, then target either moves or stays put for the next round.

The new feedback is correct with probability $1 - p$, and the target stays at the same vertex with probability $1 - b$, so in the next round, the learner queries the correct vertex (transitions to state 0) with probability $(1 - p)(1 - b)$. If either the new feedback is incorrect or the target moves at the end of this round, the learner will make a mistake next round (transitions to state 1) with probability $p+b-pb$, assuming noise of feedback and target evolution are independent. The state transition matrix is:

$$P = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} (1-p)(1-b) & p+b-pb \\ (1-p)(1-b) & p+b-pb \end{pmatrix} \end{array}$$

Each row in the transition matrix is already in its stationary distribution $\pi = (\pi_0, \pi_1)$. The expected number of mistakes over the course of R rounds is: $E[M] = R(1 - \pi_0) = B + p(R - B)$. □

B. Stars: graphs with diameter 2

A simple star graph is a graph of diameter 2, with one center vertex connecting to all the other vertices (leaf nodes). After querying the center vertex, a correct feedback will reveal the true target, so an efficient strategy is to query the center first then follow the feedback. We assume the target only moves among the leaf nodes, because the learner will make no more mistakes in the case that the target can move to the center: if the learner queries a wrong leaf, it takes at least 2 queries if target is on another leaf, and takes 1 query if the target is at the center.

Theorem 17. *If the feedback graph G' is a star then Algorithm 2 makes at most $2B + p(R - B) + p^2(R - B)$ mistakes in expectation.*

Proof. We again break down the chain of events in this order: first, the target either moves or stays put, then the learner queries the previous feedback received and the adversary provides a new feedback. For the B rounds that the target moves, the learner will make 1 mistake each time. For the $R - B$ rounds when the target doesn't move, if the previous feedback was incorrect, the learner will make 1 mistake; if the previous feedback was correct, the learner will make a mistake if the feedback pointed to the center, which means the previous query was a wrong leaf. Another case that the feedback points to the center is when the learner queried the correct leaf, but received an incorrect feedback pointing to the center.

Let x, y, z represent the number of times the learner queries the correct leaf, the wrong leaf, and the center respectively. Based on the analysis above, we can set up a system of linear equations:

$$x + y + z = R \tag{1}$$
$$px + (1 - p)y = z \tag{2}$$
$$B + p(R - B) + (1 - p)(R - B)(y/R) = R - x \tag{3}$$

Equation 1 is trivial; equation 2 represents the number of times the learner queries the center as a function of queries to correct/incorrect leaf nodes; equation 3 is the expected number of mistakes, which is the number of times the learner does not query the correct leaf. After elimination, equation 3 becomes:

$$\begin{aligned} E[M] &= B + p(R - B) \\ &\quad + [B + p^2(R - B)] \cdot \frac{(1 - p)(R - B)}{B + p^2(R - B) + (1 - p)R} \\ &\leq 2B + p(R - B) + p^2(R - B) \end{aligned}$$

Alternatively, if we assume each round the target moves with probability $b = \frac{B}{R}$, we can model the process using a Markov Chain with states $\{0, 1, 2\}$ representing the learner's distance from the target at each round. Similar to the analysis of the clique, we have state transition matrix:

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} (1-p)(1-b) & p & (1-p)b \\ (1-p)(1-b) & 0 & p+b-pb \\ 0 & 1-p & p \end{pmatrix} \end{array}$$

This is a fully-connected Markov Chain, and the stationary distribution $\pi = (\pi_0, \pi_1, \pi_2)$ can be calculated numerically. The expected number of mistakes over the course of R rounds is $R(1 - \pi_0)$. It can be verified that the numerical solution agrees with the analytical solution above. □

C. Quasi-stars: graphs with diameter d

We can also extend this analysis for "quasi-stars" with a central vertex connecting otherwise disjoint paths of length $d/2$ (for even d). We can generalize the Markov Chain with states $\{0, 1, ..., d\}$, representing the distance from query node to the true target. Further assume that every time the target moves, it moves for a distance of at least 2, with uniform probability of landing at any distance (≥ 2) to the target. The $(d + 1)$ by $(d + 1)$ transition matrix P can be approximated as follows:

$$P_{ij} = \begin{cases} 0 & j = i - 2, \text{ moves 2 steps closer} \\ (1 - p)(1 - b) & j = i - 1, \text{ moves 1 step closer} \\ 0 & j = i, \text{ distance to target unchanged} \\ p(1 - b) & j = i + 1, \text{ moves 1 step further} \\ p' & \text{all remaining probabilities sum to 1} \end{cases}$$

With the exception that $P_{00} = (1 - p)(1 - b)$.

With stationary distribution $\pi = (\pi_0, ..., \pi_d)$, we get expected total mistakes as $E[M] = R(1 - \pi_0)$, which can be computed numerically.

D. Graphs with diameter $o(\log n)$

From our previous analysis, we notice that the mistake bound does not depend on the number of nodes in the feedback graph, but rather the diameter, which is the largest distance from any node to the target. Therefore we consider general graphs bounded by diameter d. The mistake bound is stated as the theorem below.

Theorem 18. *If the feedback graph has diameter d, then Algorithm 2 makes at most $\frac{1}{1-p} \cdot \left(dB - \frac{pB}{1-2p} + pR\right)$ mistakes in expectation.*

The proof of this theorem consists of two lemmas: Lemma 19 and 20. We model the learning process as a random walk on a Markov Chain with states $\{0, \dots, d\}$. However, now we reverse the meaning of the states: state 0 means the query node is distance d from the true target, and state d means the query node is the target. This change does not affect the result of analysis, but greatly simplifies the notation. Every time the target moves, the random walk restarts at state 0 and moves towards state d: the learner moves 1 step forward upon every correct feedback, and moves 1 step backward upon every noisy feedback. There are two types of mistakes during the random walk: before reaching the target for the first time, every query contributes a mistake; once the learner reaches the target, it will circle around it due to noise probability $p < 1/2$, and occasionally misses the target.

The first type of mistake is captured by the hitting time of random walk on the Markov Chain from state 0 to state d. We have the following lemma:

Lemma 19. *Let $p < 1/2$, for a Markov Chain on a path of length $d+1$, the random walk with forward probability $1-p$ and backward probability p has a hitting time*

$$h_{0,d} \le \frac{d}{1-2p} - \frac{p}{(1-2p)^2} \ .$$

Proof. The transition probabilities of the Markov Chain are:

$$P_{ij} = \begin{cases} 1-p & j = i+1 \text{ (move towards target)} \\ p & j = i-1 \text{ (move away from target)} \\ 1-p & j = i = d \text{ (self-loop on the target)} \\ p & j = i = 0 \text{ (self-loop on nodes furthest from the target)} \end{cases}$$

Let $r = \frac{p}{1-p}$. It follows from the assumption $p < \frac{1}{2}$ that $r < 1$. The hitting time analysis follows:

For $i = 1 \dots d$:

$$\begin{aligned} h_{i,i+1} &= (1-p) \cdot 1 + p \cdot (1 + h_{i-1,i+1}) \\ &= 1 + p \cdot (h_{i-1,i} + h_{i,i+1}) \end{aligned}$$

We solve the recurrence: $h_{i,i+1} = \frac{1+p \cdot h_{i-1,i}}{1-p}$ with base cases: $h_{0,1} = \frac{1}{1-p}$ and $h_{1,2} = \frac{1+\frac{p}{1-p}}{1-p} = \frac{1}{(1-p)^2}$.

For $i \ge 2$:

$$\begin{aligned} h_{i,i+1} &= \frac{\left(\sum_{j=0}^{i-2} (1-p)^{i-j} \cdot p^j\right) + p^{i-1}}{(1-p)^{i+1}} \\ &= \frac{(1-p)^i \cdot \sum_{j=0}^{i-2} \left(\frac{p}{1-p}\right)^j + p^{i-1}}{(1-p)^{i+1}} \\ &= \frac{1}{(1-p)^{i+1}} \cdot \left((1-p)^i \cdot \frac{1-r^{i-1}}{1-r} + p^{i-1}\right) \\ &= \frac{1}{(1-p)^{i+1}} \cdot \left((1-p)^{i+1} \cdot \frac{1-r^{i-1}}{1-2p} + p^{i-1}\right) \\ &= \frac{1}{1-2p} - \frac{r^{i-1}}{1-2p} + \frac{p^{i-1}}{(1-p)^{i+1}} \\ &= \frac{1}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right) \cdot r^{i-1} \end{aligned}$$

$$\begin{aligned} h_{2,d} &= \sum_{i=2}^{d-1} h_{i,i+1} \\ &= \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right) \cdot \sum_{i=2}^{d-1} r^{i-1} \\ &= \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right) \cdot \frac{r - r^{d-1}}{1-r} \\ &\le \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right) \cdot \frac{p}{1-2p} \end{aligned}$$

$$\begin{aligned} h_{0,d} &= h_{0,1} + h_{1,2} + h_{2,d} \\ &\le \frac{1}{1-p} + \frac{1}{(1-p)^2} + \frac{d-2}{1-2p} \\ &\qquad + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right) \cdot \frac{p}{1-2p} \\ &= \frac{d-2}{1-2p} - \frac{p}{(1-2p)^2} + \frac{1}{1-p} \\ &\qquad + \frac{1}{(1-p)^2} + \frac{p}{(1-p)^2(1-2p)} \\ &= \frac{d}{1-2p} - \frac{p}{(1-2p)^2} \end{aligned}$$

□

Next we consider the second type of mistake. Once the learner reaches the target, it will keep reporting the correct node unless it receives noisy feedback and is misguided to move away from the target, which will cause a mistake for the next query. We bound the fraction of time the learner misses the target with the following lemma.

Lemma 20. *Let $p < 1/2$, after reaching state d and before the next target transition, the expected fraction of time the learner wanders away from state d is bounded by $T_{off} \le \frac{p}{1-p}$.*

Proof. Once the random walk reaches state d (learner queried the correct target), let T_d denote the expected time spent at state d and $r = \frac{p}{1-p}$, we have the following recurrence relations:

$$\begin{aligned} p \cdot T_d = (1-p) \cdot T_{d-1} &\implies T_{d-1} = r \cdot T_d \\ T_{d-1} = (1-p) \cdot T_{d-2} + p \cdot T_d &\implies T_{d-2} = r \cdot T_{d-1} \\ &\dots \\ p \cdot T_1 = (1-p) \cdot T_0 &\implies T_0 = r \cdot T_1 \\ \text{For } i = 0 \dots d : T_i &= r^{d-i} \cdot T_d \end{aligned}$$

The expected fraction of time not spent at state d:

$$T_{\text{off}} = 1 - \frac{T_d}{\sum_{i=0}^{d} T_i} = 1 - \frac{1-r}{1-r^{d+1}} \leq r = \frac{p}{1-p},$$

which finishes the proof. □

Note that the hitting time $h_{0,d}$ is linear in d, and T_{off} is positively related to entropy $H(p)$. Combining the results above, we can prove Theorem 18:

Proof of Theorem 18. Assume every time the random walk restarts at state 0, state d can be reached before the next restart. This means every time the target moves, the learner is able to reach the target before its next transition. Since the learner makes a mistake every round spent on the hitting time, this is the worst case assumption because the learner is forced to make all the mistakes possible for each target transition. Combining the two types of mistakes from previous lemmas, the total expected number of mistakes is:

$$\begin{aligned} E[M] &= B \cdot h_{0,d} + (R - B \cdot h_{0,d}) \cdot T_{\text{off}} \\ &\leq B \cdot \Big(\frac{d}{1-2p} - \frac{p}{(1-2p)^2} \Big) \cdot \frac{1-2p}{1-p} + \frac{pR}{1-p} \\ &= \frac{1}{1-p} \cdot \Big(dB - \frac{pB}{1-2p} + pR \Big), \end{aligned}$$

which completes the proof. □

In the case that $d = 2$, the bound in Theorem 18 for a general diameter-2 graph is slightly larger than the bound from Theorem 17 for the star graph. This makes sense because a star is the best case diameter-2 graph, with a center node that when queried provides information to the true target.

In the case that $d = o(\log n)$ and $p = o(H(B/R))$, we notice that the result from Theorem 18 is comparable to the trivial upper bound of Algorithm 1 as stated in Corollary 11. This means that if the learner has very limited information on target transition, or the transition model is complex, and the graph is bounded by low diameter, then Algorithm 2 makes a huge improvement on computational efficiency without too much sacrifice on the number of mistakes. Note that a complex transition model is often correlated with a low diameter feedback graph: highly connected graphs tend to have low diameters, and potentially complex transitions due to the close relationships between concepts.

E. Paths: graphs with diameter n

We also note that while path graphs seem like an easy case, they actually present difficulties due to their large diameter.

An upper bound on noisy binary search was given by Ben-Or and Hassidim [11]. Their algorithm returns the correct element with probability $(1-\delta)$ with an expected $\frac{(1-\delta)}{1-H(p)} \cdot \big(\log n + O(\log \log n) + O(\log(1/\delta)) \big)$ queries. This can be implemented in poly-time.

A naive algorithm for the shifting target case is to run their algorithm k times, setting δ appropriately small, for example, $\delta = 1/\log(kn)$. Then as both k and n go to infinity, the probability of failure goes to 0.

If $k \approx \log(n)$, $B \approx k$, and the number of rounds is much larger than the expected number of queries, this naive algorithm essentially matches the mistake bound from Emamjomeh-Zadeh and Kempe [6]. The difference is the $k \log k$ vs. $k^2 \log k$ and $R \cdot H(B/R)$ vs. $\log(\log(kd))$ terms.

References

[1] E. Emamjomeh-Zadeh, D. Kempe, M. Mahdian, and R. E. Schapire, "Interactive learning of a dynamic structure," in *Algorithmic Learning Theory*. PMLR, 2020, pp. 277–296.

[2] D. Angluin, "Queries and concept learning," *Mach. Learn.*, vol. 2, no. 4, pp. 319–342, 1987.

[3] P. Awasthi, M. Balcan, and K. Voevodski, "Local algorithms for interactive clustering," *J. Mach. Learn. Res.*, vol. 18, pp. 3:1–3:35, 2017.

[4] M.-F. Balcan and A. Blum, "Clustering with interactive feedback," in *International Conference on Algorithmic Learning Theory*. Springer, 2008, pp. 316–328.

[5] Á. D. Lelkes and L. Reyzin, "Interactive clustering of linear classes and cryptographic lower bounds," in *Algorithmic Learning Theory - 26th International Conference, ALT 2015, Banff, AB, Canada, October 4-6, 2015, Proceedings*, ser. Lecture Notes in Computer Science, K. Chaudhuri, C. Gentile, and S. Zilles, Eds., vol. 9355. Springer, 2015, pp. 165–176.

[6] E. Emamjomeh-Zadeh and D. Kempe, "A general framework for robust interactive learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 7085–7094.

[7] D. Dereniowski, S. Tiegel, P. Uznanski, and D. Wolleb-Graf, "A framework for searching in graphs in the presence of errors," in *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, ser. OASICS, J. T. Fineman and M. Mitzenmacher, Eds., vol. 69. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 4:1–4:17.

[8] L. Becerra-Bonache, A. Dediu, and C. Tirnăucă, "Learning DFA from correction and equivalence queries," in *Grammatical Inference: Algorithms and Applications, 8th International Colloquium, ICGI 2006, Tokyo, Japan, September 20-22, 2006, Proceedings*, ser. Lecture Notes in Computer Science, Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, Eds., vol. 4201. Springer, 2006, pp. 281–292.

[9] O. Bousquet and M. K. Warmuth, "Tracking a small set of experts by mixing past posteriors," *J. Mach. Learn. Res.*, vol. 3, pp. 363–396, 2002.

[10] E. Emamjomeh-Zadeh, D. Kempe, and V. Singhal, "Deterministic and probabilistic binary search in graphs," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, D. Wichs and Y. Mansour, Eds. ACM, 2016, pp. 519–532.

[11] M. Ben-Or and A. Hassidim, "The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well)," in *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 2008, pp. 221–230.