A Unified Analysis of Dynamic Interactive Learning

Allerton 2023

Xing Gao, Thomas Maranzatto, Lev Reyzin

University of Illinois Chicago

September 27, 2023

Some Motivation: Recommender Systems

- Recommender systems are integral to many online services today (Amazon, Google, Spotify,...)
- These systems are not static, users preferences change due to external factors
- The system itself may also induce changes in the user
- What is a good mathematical model for this scenario, and can we obtain convergence results?

Previous Work on Static Models

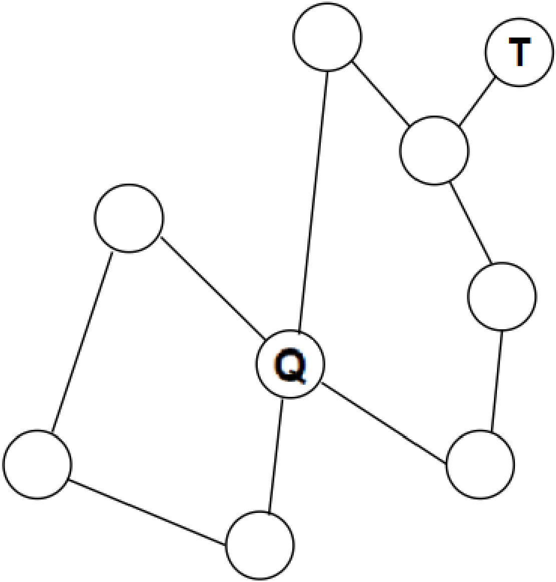
Emamjomeh-Zadeh and Kempe introduced a (static) model for learning with user feedback [1]. It is assumed the user has a finite set of possible preferences modelled as a graph, and a single target preference ranked as 'the best'.
The learner queries vertices in $v \in V(G)$, and receives feedback from the user as an edge $(v, u) \in E(G)$ on the shortest path from v to t

Static Models (cont.)

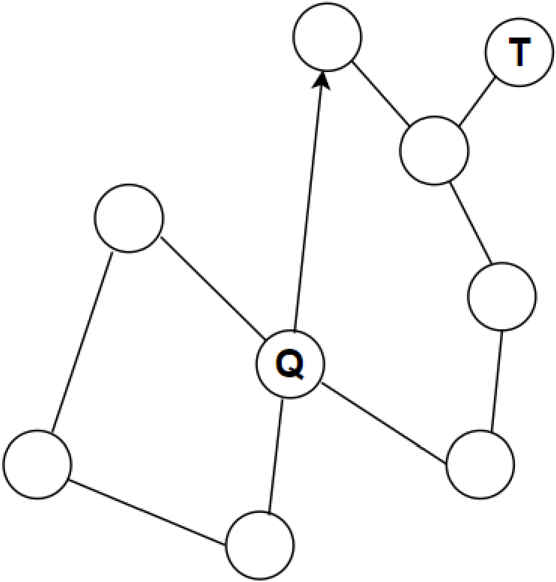
A specification can be given as:

- The learner has access to a weighted graph $G = (V, E, w)$, nodes represent concepts and edges are possible transitions between them
- A target concept $t \in V(G)$ is a fixed node that the learner wishes to discover
- The learner proposes a vertex v as the target, and receives feedback as an edge out of v
- Edge weights satisfy a key property: if the learner proposes $q \neq t$, correct user feedback z lies on a shortest path from q to t
- user feedback may be noisy (and adversarial) with probability $p < \frac{1}{2}$

A Simple Example



A Simple Example



Quick Recap on Weighted Majority

parameter: $\epsilon \in [0, 1]$

Initialize the weights $w_i = 1$ for all experts.
For each round t:
  Make predictions using weighted majority vote based on w.
  For each expert i:
    If the i-th expert's prediction is correct, w_i stays the same.
    Otherwise, $w_i \leftarrow w_i(1 - \epsilon)$.

- Setting: N experts, each makes predictions that are correct some of the time
- Updates bias towards experts which have a history of being correct
- Theorem [2] : MW makes at most $\frac{\log N + m_T^* \log \frac{1}{\epsilon}}{\log \frac{2}{1+\epsilon}}$ mistakes after T rounds, where m_T^* is the number of mistakes made by the best expert

A Multiplicative Weights algorithm for Graph Feedback

Emamjomeh-Zadeh and Kempe [1] gave a multiplicative weight update algorithm for the digraph problem:

Noiseless case

- Assign initial likelihoods $\lambda(v) = 1$ for each node $v \in G$.
- Query the 'median' node with shortest weighted distance to all other nodes: $q = argmin_v \sum_{v'} d(v, v') \cdot \lambda(v')$.
- Feedback z is a neighbor of the median. $\forall v \in G$:
  if v is *consistent* with z, $\lambda(v) \times 1$
  if v is *inconsistent* with z, $\lambda(v) \times 0$
  v is *consistent* with z means z is on a shortest $q - v$ path, ie., v could be the target
- $O(\log n)$ queries suffice [1]

A Multiplicative Weights algorithm for Graph Feedback, Noisy Setting

Noisy case

- Feedback is incorrect with probability $p < 1/2$ known to the learner
- Similar weight update idea: after querying median q and receiving feedback z, $\forall v \in G$:
  if v is *consistent* with z, $\lambda(v) \times (1 - p)$
  if v is *inconsistent* with z, $\lambda(v) \times p$
- with probability $1 - \delta$, target t can be found using

$$\frac{(1-\delta)}{1-H(p)} \cdot \left(\log n + o(\log n) + O(\log^2(1/\delta)) \right)$$

queries. [1]

Interactive Learning of a Dynamic Structure

In a follow-up paper, Emamjomeh-Zadeh and Kempe [3] proposed an extension to the previous paper, where the target is allowed to move B times in the digraph.

- **Shifting-target** model: the target moves in a pre-determined subset of k nodes. The subset is unknown, but the parameter k is known.
  Eg: multiple users
- **Drifting-target** model: there's a transition graph G' with maximum degree Δ, and the target moves following the transition graph (G' is known to the learner, and not necessarily the same as the feedback graph G)
  Eg: user preference changes over time

Generic Algorithm for Dynamic Interactive Learning

Sketch of the algorithm

- Consider all V^R possible sequences of targets throughout the R rounds, and treat these as Experts in the Multiplicative Weights Framework
- Query the 'median node' at each round, where this depends on the weight of sequences.
- On feedback, update the weight of *sequences* of nodes, as well as individual nodes.
- This gives polynomial mistake bounds, but at the cost of exponential computation. In the special cases of Drifting and Shifting, [3] give (quasi) polynomial-time algorithms.
- The key observation for speed-up is that we don't need to keep track of sequence weights, just node weights

Previous Bounds [3]

Shifting Target

$$\frac{1}{1-H(p)} \cdot \left(k \cdot \log n + B \cdot \log k\right)$$

$$\leq M(A) \leq$$

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

Drifting Target

$$\frac{1}{1-H(p)} \cdot \left(\log n + B \cdot \log \Delta\right)$$

$$\leq M(A) \leq$$

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

Our Contributions

1. We propose a generalized transition model that includes the Shifting and Drifting models as special cases
2. We improve the lower bound for the Drifting target model to match the upper bound given in [3]
3. We analyze algorithms for low diameter graphs that require $O(1)$ computation per round

A General Model

- Our model is inspired by the Drifting target model over a larger graph G'
- Feedback graph $G = (V, E, w)$
  Transition graph $G' = (V', E', \pi)$, in general G' is a bigger graph
- Allow multiple vertices in G' to represent each node in G

Likelihood Update

Sketch of the algorithm

- Apply a multiplicative weights style algorithm, with 'experts' given as sequences of vertices in the transition graph: $(V(G))^R$
- Transition graph G' bounds the number of possible sequences there are as $n' \cdot \Delta'^B \cdot \binom{R}{B}$
- Likelihoods in the r'th round are aggregated from duplicate vertices in the transition graph
- Compute median q_r using likelihoods in feedback graph
- update weights in G' based on feedback and transition information, then aggregate into G.

Interactive Learning Likelihood Upper Bound

Let n', Δ' be the number of vertices and maximum degree of the feedback graph G'.

Theorem

[GMR '23] The Likelihood Update Algorithm runs in time $O(\Delta' \cdot n' + poly(n))$, uses space $O(n')$, and makes no more than

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

mistakes in expectation.

Simplified Shifting Target Analysis

Corollary

([3], [GMR '23]) In the Shifting Target model, The Likelihood Update Algorithm runs in time $O(k^2 \cdot n^k)$*, uses space* $O(k \cdot n^k)$*, and makes at most*

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

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. Apply the Theorem from the previous slide. $\square$

Improved Drifting Target Lower Bound

Theorem

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.

Constant Computation Algorithms

Consider the following simple algorithm:

Algorithm 1 'Follow the Feedback' Procedure for Interactive Learning

$q_1 \leftarrow \mathrm{argmin}_{i \in V} \sum_{j \in V} w(i,j)$ {Start with a 'center' vertex}
for $1 \le r \le R$ **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 natural question is, for what classes of graphs and under what transition models does this give polynomial mistake bounds?

A Diameter Bound

Theorem

Suppose G' is as before, and let d be the diameter of G'. Then the 'Follow the feedback' algorithm makes at most

$$\frac{1}{1-p} \cdot \left(dB - \frac{pB}{1-2p} + pR \right)$$

mistakes in expectation.

Proof Idea

1. Reduce G' to P_d, a path on d vertices
2. Algorithm 1 makes a mistake anytime the feedback is not the target vertex
3. Apply Markov chain hitting time bounds on P_d