Paul Christiano Follow
Nov 20, 2015 · 3 min read

(*Warning: technical + uninteresting + unpolished.*)

In this proposal, the algorithms A and B were originally reinforcement learning algorithms rather than simple online learners. I made this choice because traditional regret guarantees weren't enough to analyze the overall system.

In this post I show that it was unnecessary to go all of the way to reinforcement learning; we can make simple changes to the exploration policy which give us good enough regret bounds for our purposes.

It's not totally clear what the practical upshot of this is—for practical algorithms, the situation with respect to local minima is considerably more complex than in these simple theoretical environments. But it does seem that there is at least in theory an intermediate possibility between conventional regret guarantees and reinforcement learning, and any practical algorithm that met this intermediate goal would be good enough.

Moreover, I think it's safe to say that my original motivation for using RL learners was not reasonable, and so I have modified the proposal to use online learners.

Bandit algorithms with episodic exploration

In the multi-armed bandit problem, there is a space of possible "arms" X and a sequence of rounds $t = 0, 1, \ldots$

In each round t, the player picks an arm $x^t \in X$ and receives a corresponding payoff $p^t(x^t)$. Their choice will generally be probabilistic; the payoffs are allowed to depend on this probability distribution but not on which arm is actually chosen. (Thus we can talk meaningfully about the payoff $p^t(y)$ that *would have been obtained* if the random choice had selected y instead of x^t.)The goal of the player is to maximize their total payoff.

There is a simple algorithm (EXP3) that guarantees that, for every arm y:

$\Sigma\, p^t(x^t) \geq \Sigma\, p^t(y)$ - O(sqrt(T|X|))

Where x^t is the arm actually selected by the algorithm in step t, and the sum is taken over the rounds t = 0, 1, ..., T.

The way you get this guarantee is by using multiplicative updates, and in each step exploring—essentially playing randomly—with some small probability.

Episodes

The payoffs p^t are the payoffs that would actually have been obtained "if you had played differently." In my proposal for approval-directed agents, we would like to avoid A and B getting stuck in a local minimum. This might arise if B declines to challenge because B is currently pursuing a strategy that would result in a low payoff if B had to argue against A. Because B never challenges, B never has an opportunity for exploring alternative strategies for arguing, while it turns out that one of those alternative strategies would have worked.

To fix this problem we would like a more aggressive form of exploration, which corresponds to a change both in the algorithm and in the regret bound.

We partition the rounds t into a sequence of episodes, each consisting of a sequence of contiguous rounds of length at most L. We allow the payoffs within each episode to depend in an arbitrary way on the choices in that episode—in particular, the payoff in one round can depend on payoffs later in the round.

We modify the regret bound as follows:

- On the RHS, we replace $p^t(y)$ with $p^{t*}(y)$, which is the payoff that would be obtained in round t if arm y was chosen in round t, and in every subsequent round of the episode.
- On the RHS, we replace the regret term O(sqrt(T |X|)) with O(L sqrt(T |X|)).

It's easy to see that the increase to the regret is necessary, once we allow the payoffs within an episode to depend on future choices in the episode. This increases the necessary amount of training data for convergence by a factor of L, which is quite a lot. In principle this increase applies to RL algorithms as well, and in practice it seems like our situation remains strictly easier than the RL situation.

In order to define p^{t*} formally we need to define the counterfactual, but the definition is pretty intuitive.

Changing the bandit algorithm to obtain this bound is basically trivial. When we play randomly, we use that random choice for the remainder of the episode.