EXP4-DFDC: A Non-Stochastic Multi-Armed Bandit for Cache Replacement

Anonymous Author(s)

Abstract

In this work we study a variant of the well-known multi-armed bandit (MAB) problem, which has the properties of a delay in feedback, and a loss that declines over time. We introduce an algorithm, EXP4-DFDC, to solve this MAB variant, and demonstrate that the regret vanishes as the time increases. We also show that LeCaR, a previously published machine learning-based cache replacement algorithm, is an instance of EXP4-DFDC. Our results can be used to provide insight on the choice of hyperparameters, and optimize future LeCaR instances.

1 Introduction

The *Multi-Armed Bandit* (MAB) problem [6] can be thought of as a sequence of "pull" requests for which each request must elicit a response. MAB can be defined as a game where a player plays a T-round prediction game: the player has access to a panel of N experts, each of whom is following a specific strategy. In any round t, the player consults the N experts. Expert i recommends an action $\xi_i(t) \in \{0, \ldots, K\}$ from K available options. Each action is associated with a loss value, which is revealed as feedback to the player either immediately or after a delay.

We consider a variant of the MAB problem, which takes into account cases for which as the delay in feedback gets larger, the loss associated with the chosen action decreases. This variant differs from any existing formulations of MAB with a delayed feedback [8, 1, 4, 5, 2, 3] where greater feedback delays incur higher loss.

The cache replacement problem in storage systems can be considered as an instance of the MAB problem: feedback is available on cache misses due to recent page evictions, and misses not found in the page eviction history correspond to evictions from a distant past and are ignored.

2 Problem Statement

For our version of MAB, each cache replacement algorithm is thought of as an "expert" that recommends which of the K items in the cache to evict on every miss. We assume that each algorithm is allowed a limited size "history" to keep track of recent evictions, and to help in its decision making.

The history also provides a natural aging mechanism for recent evictions. If the item is still in history, feedback on its eviction is considered valuable and is used; otherwise, it is assumed that a significant amount of time has passed, and the feedback is ignored. For an eviction that occurred at time (or step or round) t', the delayed feedback offered at time $t \geq t'$ is denoted by the indicator function, $f_{(t',t)} \in \{0, 1\}$. If feedback arrives at time t, then $f_{(t',t)} = 1$, and is 0 otherwise. The delay function is given by $d(t) = t - t' \in [1, m]$, where m is the delay threshold (as well as the history size). Feedback provided at time t, for an action $\xi_i(t')$ taken at time t', is based on the recommendation of expert i, and observes a *loss* that is denoted by $l_{\xi_i(t')}(t, t')$. Note that feedback provided after a delay threshold m incurs a loss of 0, and we consider the oblivious scenario where the loss function is fixed, not adaptive.

3 Technical Contributions

Expert i is associated with a weight w_i, and all experts start off with equal weights. For each request, each expert recommends an action; a final action is chosen to be expert i's recommendations with probability proportional to w_i. The weights of the experts get updated whenever $f_{(t',t)} = 1$, i.e., when feedback is provided. Shorter delays incur higher loss, and the loss decays with delay. Thus, regret is highest when feedback is immediate, i.e., when $t = t' + 1$ for the chosen action. If the maximum loss for a decision is $L = l_{\xi_i(t')}(t' + 1)$, then the estimated loss after delay $d(t)$ is given by $L/d(t)$, until the delay reaches the predefined threshold m, after which it becomes negligible. (See Figure 1.) As $t - t'$ increases, the loss for the chosen action decays. Thus, the Estimated loss calculation in EXP4-DFDC the major difference from EXP4.

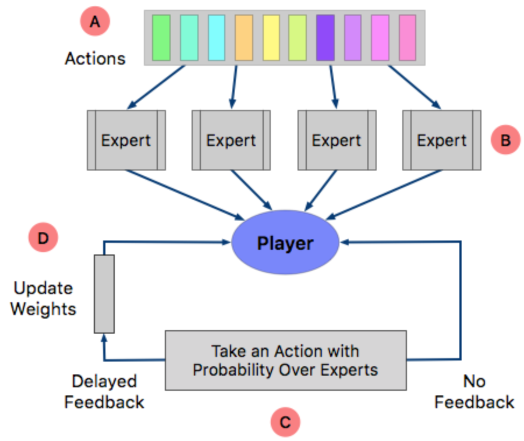

Figure 1: MAB problem variant. A) The set of K actions that are available to the panel of N experts. B) Each expert recommends an action to the player, which selects one. C) The feedback to the player can be immediate, or delayed, in which case the weights of the experts are updated based on the loss the player observed (D).

The regret, $R_A(T)$, of any adaptive learning algorithm A after round T is used to measure its performance. The main objective of an adaptive learning strategy is to minimize this regret, achieved by ensuring it vanishes over time. An explicit bound for the regret function can show that the algorithm EXP4-DFDC can be computed and that it has a vanishing regret. The regret of EXP4-DFDC can be shown to be of the form O($\sqrt{KT \ln N}$), and as a result, the average regret will vanish with increasing time as $\overline{R}_A(t) = R_A(t)/T = O(T^{-1/2})$. Thus, as $T \to \infty$, $\overline{R}_A(t) \to 0$. We focus on applications where the loss of an action decays with a delay in feedback, but only up to a limit, after which it becomes zero. Details of the proof are omitted from this short abstract.

4 Discussion

We consider the cache management problem, which is a classic resource allocation problem from storage systems. The LeCaR algorithm is a recent cache replacement algorithm that uses online reinforcement learning to help decide which entry to evict when a new item is to be stored in the cache following a "cache miss" [7]. LeCaR relies on only two very fundamental cache replacement policies, namely the *Least Recently Used* (LRU) policy and the *Least Frequently Used* (LFU) policy. As a result, $N = 2$ represents the number of experts in LeCaR. LeCaR assumes that the best strategy at any given time is a probabilistic mix of the two policies and attempts to "learn" the optimal mix using a regret minimization strategy. With a slight modification, the LeCaR [7] cache replacement algorithm is an instance of EXP-DFDC: the position of an item in history is a reflection of the delay in feedback for its eviction decision, and is bounded by the history size. The regret is greatest if an item is requested immediately after its eviction. Else, as time wears on, until a threshold is reached, the regret continues to decay. If it is not present in the history, then the feedback is ignored, thus making its regret equal to zero. Cache management can be thought of as a special case of the MAB problem, and LeCaR can be considered as an application-specific version of EXP4-DFDC with two experts, delayed feedback and decaying losses, and has been shown to have vanishing regret over time. We note that LeCaR outperforms other state-of-the-art cache replacement algorithms for small cache sizes [7] with a fixed learning rate of 0.45. From our analysis, we demonstrate that it is possible to infer the learning rate in this class of algorithm without it being fixed a priori. Details of the mathematical proof of vanishing regret can be found in a forthcoming manuscript.

References

[1] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[2] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[3] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*, 2011.

[4] John Langford, Alexander J Smola, and Martin Zinkevich. Slow learners are fast. *Advances in Neural Information Processing Systems*, 22:2331–2339, 2009.

[5] Gergely Neu, Andras Antos, András György, and Csaba Szepesvári. Online markov decision processes under bandit feedback. In *Advances in Neural Information Processing Systems*, pages 1804–1812, 2010.

[6] Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

[7] Giuseppe Vietri, Liana V. Rodriguez, Wendy A. Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. Driving cache replacement with ML-based LeCaR. In *10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, Boston, MA, 2018. USENIX Association.

[8] Marcelo J Weinberger and Erik Ordentlich. On delayed prediction of individual sequences. *IEEE Transactions on Information Theory*, 48(7):1959–1976, 2002.