Note on "Fast EXP3 Algorithms"

丛宇

December 15, 2025

A recent paper[1] shows that EXP3 algorithm for adversarial bandits can be implemented in $O(1)$ expected time per round (under RAM model).

1 Problem Settings

Then bandit has $K \geq 2$ arms. At round $t \in [T]$, the adversary decides the loss $(\ell_{t,1}, \ldots, \ell_{t,K}) \in [0,1]^K$ based on the history of the loss and player's choice for previous rounds. Note that the adversary knows the player's algorithm. Then the player selects one arm $a_t \in [K]$ and observes the loss ℓ_{t,a_t}. The player's goal is to minimize the expected pseudo-regret $\bar{R}_T$ defined as follows.

$$\bar{R}_T = \mathrm{E}\left[\sum_{t\in[T]} \ell_{t,a_t}\right] - \min_{i\in[K]} \sum_{t\in[T]} \ell_{t,i}$$

1.1 EXP3 Algorithm

EXP3 Algorithm works like multiplicative weight update method for linear programs.

EXP3:
learning rate $\eta = \sqrt{\frac{2\ln K}{KT}}$
initial weights $\{w_{1,i} = 1\}_{i\in[K]}$
in each round t:
  select arm i with probability $p_{t,i} = \frac{w_{t,i}}{\sum_{j\in[K]} w_{t,j}}$
  update weights:
    $w_{t+1,a_t} = w_{t,a_t} \exp(-\eta \frac{\ell_{t,a_t}}{p_{t,a_t}})$, $w_{t+1,i} = w_{t,i}$ for $i \neq a_t$.

EXP3 has regret $\bar{R}_T \leq \sqrt{2} \cdot \sqrt{TK \ln K}$. Note that unlike MWU, for EXP3 the weight update happens for only 1 entry in each round. So the hard part is sampling.

2 $O(\log K)$ Implementation

We can build a segment tree on the array $\{w_{t,i}\}_{i\in[K]}$. Single update in the segment tree takes $O(\log K)$ time. Hence, we forcus on the sampling part.

Let W_t be $\sum_{i\in[K]} w_{t,i}$ and let p be a random number sampled uniformly from $[0, W_t]$. Note that the prefix sums of array $\{w_{t,i}\}_{i\in[K]}$ divides the interval $[0, W_t]$ into K parts and there is a bijection from these parts to bandit arms. The probability of r falling in part i is exactly $\frac{w_{t,i}}{W_t}$. One can see that the sampling part also takes $O(\log K)$ time on the segment tree.

3 Alias Method

Alias method is a static data structure which after a $O(K)$ preprocessing samples element with probability $\frac{w_{t,i}}{W_t}$ in worst case $O(1)$ time.

We drop the subscripts t since alias method is static. Let $\bar{W}$ be the average weight W/K. The preprocess works as follows.

[1] https://arxiv.org/pdf/2512.11201v1

1. Divide elements in $\{w_i\}_{i\in[K]}$ into 2 groups. Large group contains elements with $w_i \geq \bar{W}$. The rest is Small group.

2. Repeat the following K times. Pick an element s from Small group and an element l from the Large group. Pack s with weight w_s and l with weight $\bar{W} - w_s$ into a new "bin". Remove s from Small group. Then update the weight of l to $w_l - (\bar{W} - w_s)$. If w_l becomes smaller than $\bar{W}$, move l to the Small group.

Remark The preprocess steps are well-defined. One can see in each step we remove one element from the Small group and the total weight decrease by $\bar{W}$. So the average weight of remaining elements is always $\bar{W}$ and the Large group is non-empty throughout the preprocessing. After K steps both the Large group and the Small group are empty.

When sampling elements, we first uniformly select one "bin" (which takes constant time) and then choose element s with $w_s/\bar{W}$ or l with $1 - w_s/\bar{W}$.

One can see that for small element s,

$$\Pr[s \text{ is selected}] = \frac{1}{K} \cdot \frac{w_s}{\bar{W}} = \frac{w_s}{W}$$

and for large element l,

$$\Pr[l \text{ is selected}] = \sum_{\text{bin containing } l} \frac{1}{K} \frac{\bar{W} - w_{s_i}}{\bar{W}} = \frac{w_l}{W}$$

which are the desired propabilities.

4 $O(1)$ Implementation

The fun part ends here. The idea is standard. They rebuild the alias method data structure every K rounds and always use the newly built one for sampling. Since the alias method structure is always outdated, they use reject sampling with accept chance $\frac{W_t}{W_\tau}$, where τ is when the last alias structure was built. This gives a $O(1)$ amortized expected time sampling per round.