Attaining coarse correlated equilibrium with bounded memory

Lokesh Vijaykumar

Department of Electrical Engineering, Delhi University.

Assignment CS321 - 20 December 2018

Here I am going to implement a distributed playing rule of coarse correlated equilibrium in finite games, with a proven convergence. I use the multiplicative weight algorithm to minimize players' regrets, and a circular memory buffer to maintain the game at the equilibrium. The simulations results validate the solution.

Game Definition

Throughout this paper, we consider finite n-player games, where each player i has a finite pure action set A_i; let $A = \prod_i A_i$, and let $A_{-i} = \prod_{j \neq i} A_j$. We let a_i denote a pure action for player i, and let $s_i \in \Delta(A_i)$ denote a mixed action for player i. We will typically view s_i as a vector in $\mathbb{R}^{A_i}$, with $s_i(a_i)$ equal to the probability that player i places on a_i. We let $\Pi_i(\boldsymbol{a})$ denote the payoff to player i when the composite pure action vector is $\boldsymbol{a}$, and by an abuse of notation also let $\Pi_i(\boldsymbol{s})$ denote the expected payoff to player i when the composite mixed action vector is s.

The game is played repeatedly by the players. We let $h^T = \left(\boldsymbol{a}^0, \ldots, \boldsymbol{a}^{T-1}\right)$ denote the

history up to time T. The external regret of player i against action s_i after history h^T is:

$$ER_i\left(h^T; s_i\right) = \sum_{t=0}^{T-1} \Pi_i\left(s_i, \boldsymbol{a}_{-i}^t\right) - \Pi_i\left(a_i^t, \boldsymbol{a}_{-i}^t\right).$$

We let $p_i^T \in \Delta\left(A_i\right)$ denote the marginal empirical distribution of player i 's play up to time T :

$$p_i^T\left(a_i\right) = \frac{1}{T}\sum_{t=0}^{T-1} \mathcal{I}\left\{a_i^t = a_i\right\}$$

and we let $p^T \in \Delta\left(A\right)$ denote the empirical distribution of play up to time T :

$$p^T\left(a\right) = \frac{1}{T}\sum_{t=0}^{T-1} \mathcal{I}\left\{a^t = a\right\}$$

Multiplicative Weight Algorithm

The algorithm maintains weights $w_i^{(t)}$, which are proportional to the probability that strategy i will be used in round t. After each round, the weights are updated by a multiplicative factor, which depends on the cost in the current round. Let $\eta \in \left(0, \frac{1}{2}\right]$; we will choose η later. - Initially, set $w_i^{(1)} = 1$, for every $i \in [N]$. - At every time t, $-$ Let $W^{(t)} = \sum_{i=1}^{N} w_i^{(t)}$ - Choose strategy i with probability $p_i^{(t)} = w_i^{(t)}/W^{(t)}$; $-$ Set $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)^{\ell_i^{(t)}}$.

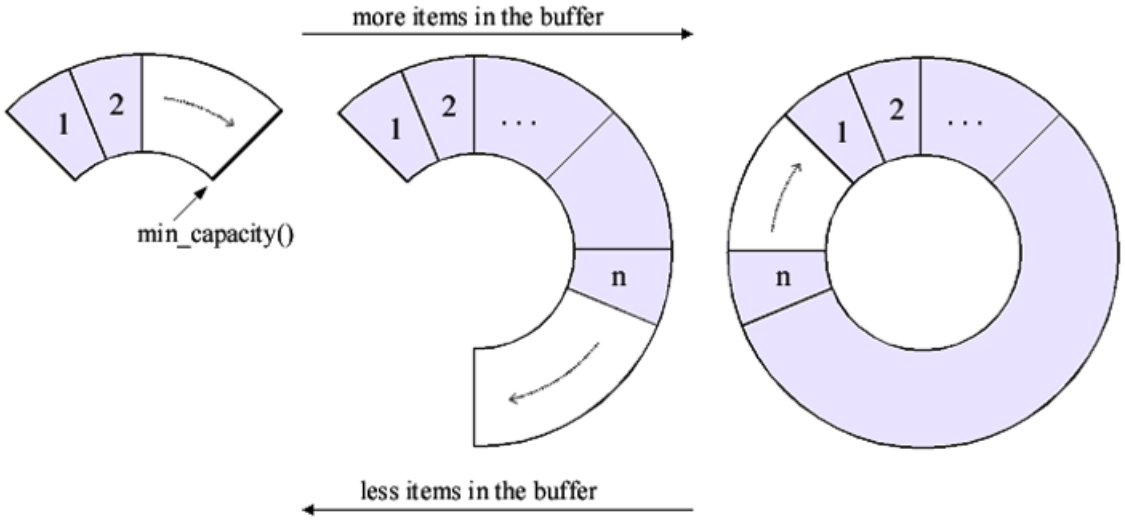
Under the MW algorithm, the individual regret ER_i of a player decays at $O(\frac{1}{t^{-3/4}})$, implying that there exists a constant K such that $(\frac{K}{\epsilon})^{\frac{4}{3}}$ rounds are necessary to minimize players regrets below any ϵ. We have $p^{t=(\frac{K}{\epsilon})^{\frac{4}{3}}} \in \epsilon - CCE$.

Following the learning phase of duration S, the players are playing according to the simple rule that I describe in the next section.

Circular Memory Buffer

Let $\mathcal{B}$ be an empty memory of fixed size $S = (\frac{K}{\epsilon})^{\frac{4}{3}}$ used for recording the action profile after each round. Players play until the memory buffer is full and **keep playing the oldest action**

profile in the memory buffer, which then enters a rotational cycle as explained in the figures.



Circular queues use first in first out logic. A property of the circular buffer is that when it is full and a subsequent write is performed, then it starts overwriting the oldest data. In our case, the content of the buffer is shifted (rotated) by one step so that the newest action profile becomes by the oldest action profile.

Implementation in C

```
enum {N = 10000};
int buf[N];      // N elements circular buffer
int end = 0;     // write index
int start = 0;   // read index

void put(int item) {
  buf[end++] = item;
  end %= N;
```

```
}

int get() {
  int item = buf[start++];
  start %= N;
  return item;
}
```