Introducing Multi-Stage Multiplicative-Weights Update

An Empirical Evaluation of Convergence to Correlated Equilibria

Michael Han Ashley Yu
Mentor: Noah Golowich

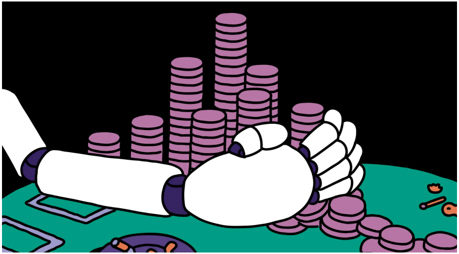
MIT PRIMES October Conference

Table of Contents

Background

Big Picture

- Algorithmic game theory: the intersection of **game theory** and **computer science**
- Recent advances in AI have led to breakthroughs in various multi-agent games: Poker, Go, Avalon, Diplomacy, etc.
- A key part of these advances is **no-regret learning**, which is currently state-of-the-art for finding equilibria



What Is a Game?

- Normal-form games
  - Representations of strategic interactions with perfect information; players choose actions simultaneously
  - Formally, we have:
    - N: the set of players
    - A_i: the set of actions played by player i
    - $u_i(a)$: the payoff received by player i if they play action a

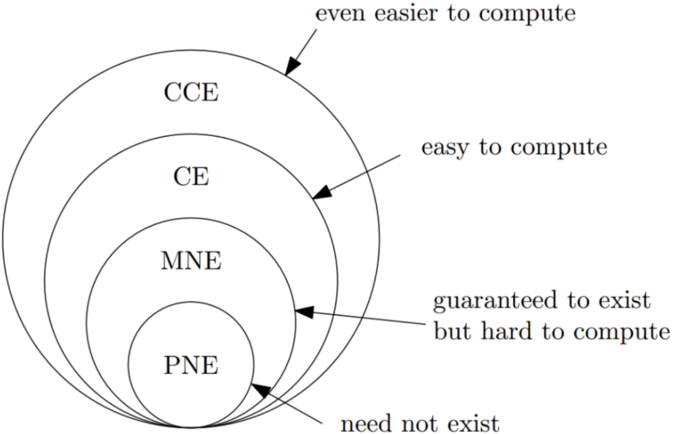
Player II

Player I		rock	paper	scissors
	rock	(0, 0)	(-1, 1)	(1, -1)
	paper	(1, -1)	(0,0)	(-1,-1)
	scissors	(-1, 1)	(1, -1)	(0, 0)

- Extensive-form games
- The goal of these games is to **compute equilibrium**

Types of Equilibrium

- Pure Nash Equilibrium
- Mixed Nash Equilibrium
- Correlated Equilibrium
- Coarse Correlated Equilibrium



How do we find equilibria?

- Finding Nash equilibrium is **computationally hard**; no known polynomial-time algorithms for an arbitrary game
  - Relax the notion of equilibrium: focus on finding correlated equilibria
- We can use **regret** to help us converge to equilibria
- What is regret?
  - **External regret**: how much better we could've done if we just played the *single* best action (in hindsight)
  - **Swap regret**: how much better we could've done if we swapped each action we played with another (better) action
  - External regret $\leq$ Swap regret

Regret Minimization

- We use **no-regret learning** to converge to an approximate coarse correlated equilibrium!!
- Basic setup for each iteration $t = 1, 2, \ldots T$ of regret minimization:
  - The player chooses a mixed strategy p^t, (a probability distribution over A)
  - The adversary chooses a loss vector $\ell^t : A \to [0, 1]$
  - The player plays an action a^t based on p^t and receives loss $\ell^t(a^t)$

Existing No-Regret Algorithms

Overview

- No external-regret algorithms
  - **Multiplicative-Weights Update (MWU)**
    - Folklore
  - **Optimistic MWU** [SALS15]
    - Last iteration convergence
- No swap-regret algorithms
  - **Blum-Mansour** [BM07]
    - External regret $\rightarrow$ swap regret
  - **TreeSwap** [DDFG24]
    - Better for very large action spaces
- Note that no swap-regret algorithms also minimize external regret

Multiplicative-Weights Update (MWU)

- Maintain weights w_i which are assigned to each possible action $i \in \{1, \ldots, N\}$
  - Begin by playing a uniform distribution: $w_i^1 = 1$ for all i
- Run for many iterations: $t = 1, \ldots T$. For each iteration:
  - Receive losses ℓ_i^t from the adversary
  - Update the corresponding weights:

$$w_i^{t+1} = w_i^t \cdot (1-\eta)^{\ell_i^t}$$

  - Play a randomly chosen action based on the weights; we play action i with probability

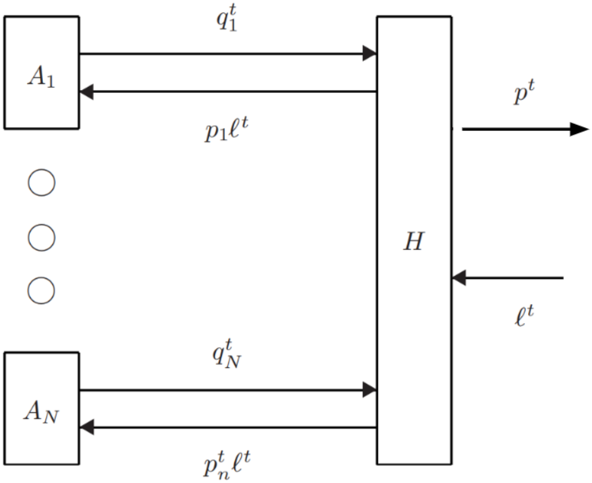
$$p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$$

MWU Pseudocode

Algorithm 1 MWU

Input: Learning rate $\eta \in (0, 1)$, number of actions N, time horizon T
Initialize: Weights $w_i^1 = 1$ for all $i \in \{1, \ldots, N\}$
for $t = 1$ to T **do**
    Normalize weights: $p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$ for all $i \in \{1, \ldots, N\}$
    Choose action: Randomly select action i with probability p_i^t
    Receive loss: ℓ_i^t for each action i
    for each action $i \in \{1, \ldots, N\}$ **do**
        Update weight: $w_i^{t+1} = w_i^t \cdot (1 - \eta)^{\ell_i^t}$
    end for
end for

- p^t = stationary distribution of the matrix made up of q_i^t
- A_i = no-external-regret algorithms



TreeSwap

- Inspired by Blum-Mansour's reduction and uses instances of no external-regret algorithms
  - The instances of the no-external-regret algorithms are updated in a blocks, meaning that they are updated after a certain number rounds, rather than for every round.
- Achieves better bounds on swap regret for larger or infinite action spaces
- Makes significant improvements in computational complexity

Our Algorithm

Multi-Stage Multiplicative-Weights Update (MS-MWU)

- This new algorithm **converges significantly faster** than any of the existing no-regret algorithms (OMWU, BM, TreeSwap) across all our experiments
- How it works:
  - Split the time horizon T into many blocks, each of length M
  - Run MWU, but each time we enter a new block, **reinitialize the weights to be the average of the weights of the previous block**
- Intuitively:
  - Optimistic MWU "predicts" the next loss, while we are "predicting" the next M losses
  - Jumping to the average = taking a shortcut that takes us closer to equilibrium

MS-MWU Pseudocode

Algorithm 2 MS-MWU

Input: Number of actions N, time horizon T, decay rate r

Initialize: block size $M \approx \sqrt{T}$, $\eta = \sqrt{\frac{\log N}{M}}$, $P_{cum} = (0, \dots, 0)$

for $t = 1$ to T **do**

  Normalize weights: $p_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_j^t}$ for all $i \in \{1, \dots, N\}$

  Choose action: Randomly select action i with probability p_i^t

  Receive loss: ℓ_i^t for each action i

  Update weights: $w_i^{t+1} = w_i^t \cdot (1 - \eta)^{\ell_i^t}$ for each action i

  Accumulate strategies: $P_{cum} = P_{cum} + p^t$

  if $t \pmod{M} = 0$ **then**

    $w^t = \frac{P_{cum}}{M}$, $\eta = \frac{\eta}{r}$, $P_{cum} = 0$

  end if

end for

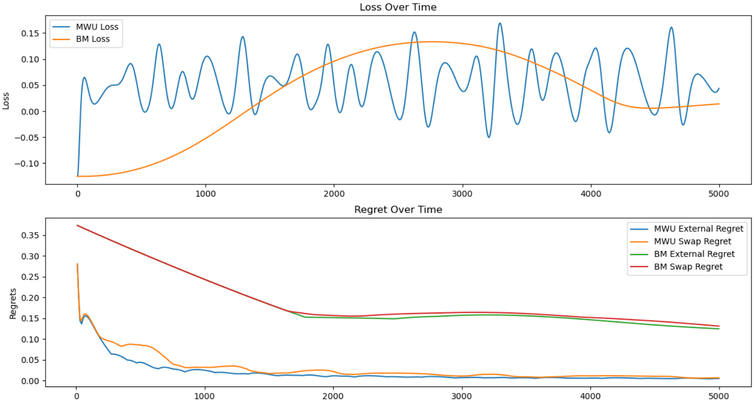
Experimental Results

Methodology

- We implemented MWU, OMWU, MS-MWU, Treeswap, and BM
- These were run on random games, Kuhn Poker, and normal-form subgames of the extensive-form game Diplomacy
- In all of these games, MS-MWU performed the best experimentally

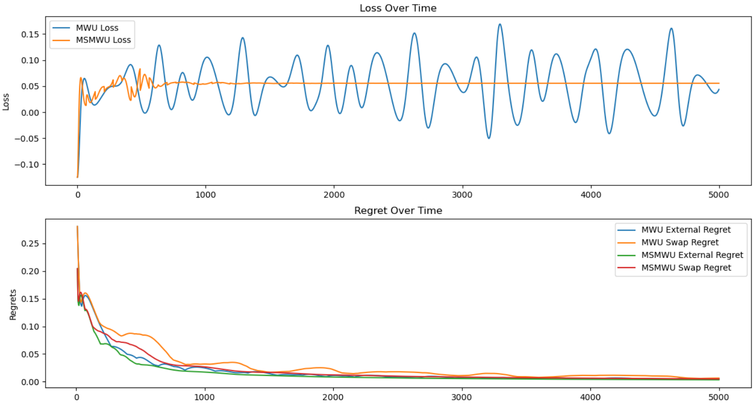
Kuhn Poker: MWU vs. BM

- MWU converges much faster than BM and minimizes swap regret



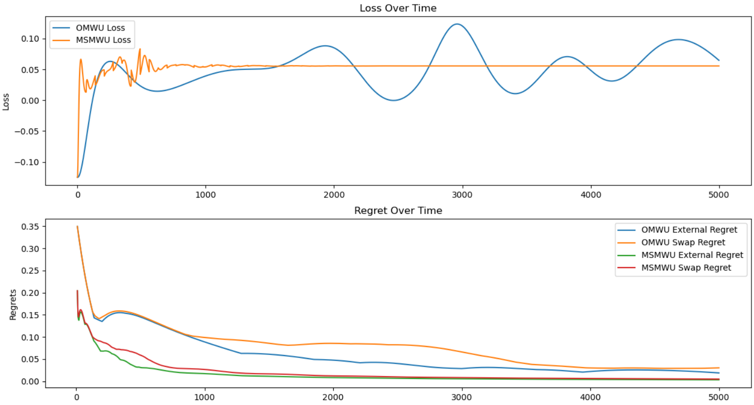
Kuhn Poker: MWU vs. MS-MWU

- MS-MWU has last iteration convergence and converges faster than MWU

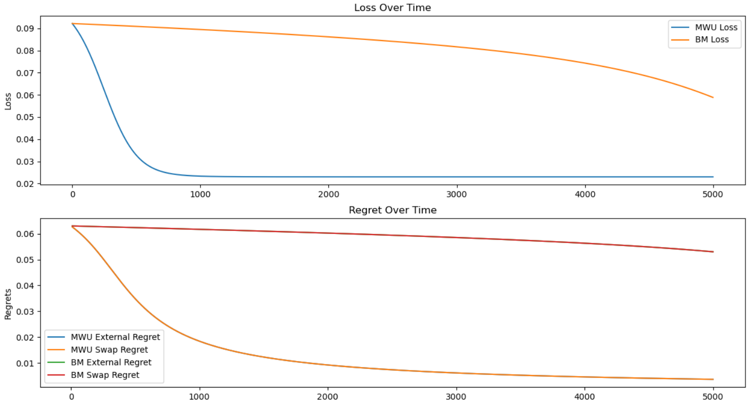


Kuhn Poker: OMWU vs. MS-MWU

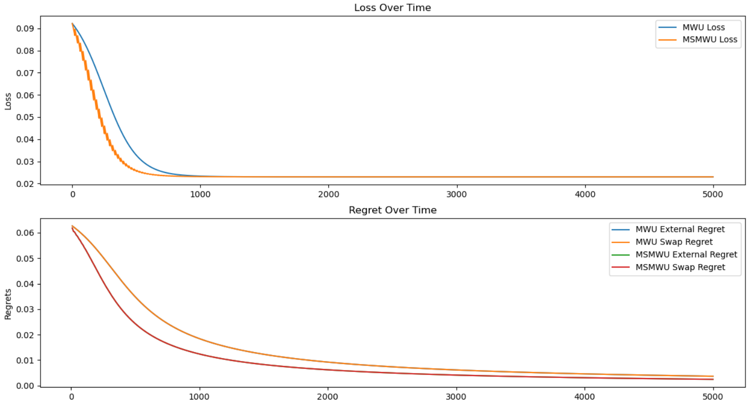
- MS-MWU also converges faster than OMWU



Diplomacy Subgame: MWU vs. BM



Diplomacy Subgame: MWU vs. MS-MWU



Further Work

- **Two main questions to focus on:**
  - Why does MS-MWU perform better than all the existing algorithms?
  - Why does MWU also have good swap regret?
- Prove a theoretical error bound for the MS-MWU algorithm
- Experiments with more algorithms
  - Regret matching, Counterfactual regret minimization
- Experiments with more games: extended-form, multi-player (3+)
  - GAMUT, OpenSpiel

Acknowledgements

We would like to thank our excellent mentor, Noah Golowich, for his continued guidance and advice. We also would like to thank MIT PRIMES and its organizers for providing us with this incredible opportunity. Finally, we would like to thank our parents for supporting us throughout this journey.

References

- Grigoriadis, M. D., & Khachiyan, L. G. (1995). A sublinear-time randomized approximation algorithm for matrix games. Operations Research Letters, 18(2), 53–58. https://doi.org/10.1016/0167-6377(95)00032-0
- Blum, A., & Mansour, Y. (2007). From external to internal regret. Journal of Machine Learning Research, 8(47), 1307–1324. https://www.jmlr.org/papers/volume8/blum07a/blum07a.pdf
- Syrgkanis, V., Agarwal, A., Luo, H., & Schapire, R. E. (2015). Fast convergence of regularized learning in games. Neural Information Processing Systems, 28, 2989–2997. https://proceedings.neurips.cc/paper/2015/file/7fea637fd6d02b8f0adf6f7dc36aed93-Paper.pdf

References

- Nisan, N., Nisan, N., Papadimitriou, C. H., Nisan, N., Tardos, É., Papadimitriou, C. H., Von Stengel, B., Blum, A., Vazirani, V. V., Codenotti, B., Kearns, M., Dodis, Y., Nisan, N., Schummer, J., Blumrosen, L., Lavi, R., Hartline, J. D., Feigenbaum, J., Jain, K., . . . Johari, R. (2007). Algorithmic Game Theory. In Cambridge University Press eBooks. https://doi.org/10.1017/cbo9780511800481