Module 2: Computing Equilibria & Regret Minimization

CSCE 631 — Algorithmic Game Theory meets LLMs

Week 2: June 2 (Mon) – June 6 (Fri)

Contents

Learning Objectives **3**

1 Lecture 5: Computing Equilibria I (70 min) **3**
- 1.1 Support Enumeration . . . 3
- 1.2 The Lemke-Howson Algorithm . . . 4
- 1.3 PPAD-Completeness . . . 4
- 1.4 LP for Zero-Sum Games . . . 4

2 Lecture 6: Computing Equilibria II (69 min) **5**
- 2.1 LP Formulation for Zero-Sum Games (Detailed) . . . 5
- 2.2 Computing Correlated Equilibria via LP . . . 6
- 2.3 Approximate Nash Equilibrium . . . 6
- 2.4 Gambit and Practical Tools . . . 6

3 Lecture 7: Regret Minimization I (65 min) **7**
- 3.1 The Online Decision-Making Model . . . 7
- 3.2 External Regret . . . 8
- 3.3 From Halving to Weighted Majority . . . 8
- 3.4 Multiplicative Weights Update (MWU) . . . 8

4 Lecture 8: Regret Minimization II (70 min) **10**
- 4.1 Regret Matching . . . 10
- 4.2 Internal Regret and Swap Regret . . . 11
- 4.3 Convergence to Equilibria . . . 11

5 Lecture 9: Regret Minimization III (67 min) **12**
- 5.1 Zero-Sum Convergence to Nash Equilibrium . . . 12
- 5.2 RM+ (Regret Matching Plus) . . . 13
- 5.3 Last-Iterate vs. Average-Iterate Convergence . . . 13
- 5.4 Bandit Feedback and EXP3 . . . 14
- 5.5 Preview: Counterfactual Regret Minimization (CFR) . . . 14

Key Algorithms Summary **15**

Suggested Reading **16**

Learning Objectives

By the end of this module, you should be able to:

1. Compute Nash equilibria using the support enumeration algorithm and describe its worst-case complexity.
2. Formulate zero-sum Nash equilibrium computation as a linear program (LP) and explain why it is polynomial-time solvable.
3. Define external regret, internal regret, and swap regret, and explain the hierarchy among them.
4. Describe the Multiplicative Weights Update (MWU) algorithm and state its $O(\sqrt{T \ln K})$ regret bound.
5. Prove that no-external-regret play converges to a coarse correlated equilibrium, and that no-swap-regret play converges to a correlated equilibrium.

1 Lecture 5: Computing Equilibria I (70 min)

Module 1 established the existence of Nash equilibria via Nash's theorem, but existence says nothing about how to *find* one. This lecture confronts the computational question head-on: given a game specified by its payoff matrices, produce a Nash equilibrium. The answer turns out to depend crucially on the structure of the game. For zero-sum games the problem reduces to linear programming and is efficiently solvable; for general-sum games it is PPAD-complete, placing it among the hardest problems whose solutions are guaranteed to exist.

1.1 Support Enumeration

The most direct approach to computing a Nash equilibrium in a two-player game exploits the indifference principle from Lecture 3. Recall that in a non-degenerate Nash equilibrium, every pure strategy in the *support*—the set of strategies played with positive probability—must yield the same expected payoff against the opponent's mixed strategy. This observation converts the equilibrium conditions into a system of linear equations for each candidate support pair.

Definition: Support Enumeration Algorithm

Given a two-player game with strategy sets S_1 and S_2, the **support enumeration algorithm** iterates over all possible support pairs (T_1, T_2) with $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$. For each pair, it solves the linear system arising from the indifference conditions: every strategy in T_1 must yield the same expected payoff against the opponent's mixture over T_2, and vice versa. A candidate solution is accepted as a Nash equilibrium if the resulting probabilities are non-negative and no strategy outside the support is a profitable deviation.

The algorithm is conceptually simple but computationally expensive. For an $n \times m$ game, there are $2^n \cdot 2^m$ possible support pairs (minus the empty sets), so the worst-case running time is exponential. Moreover, even the subset of "interesting" support sizes—where $|T_1| = |T_2|$ in non-degenerate games—still requires $\binom{n}{k}\binom{m}{k}$ iterations for each support size k. Support enumeration is practical only for small games ($n, m \leq 15$ or so).

1.2 The Lemke-Howson Algorithm

Definition: Lemke-Howson Algorithm

The **Lemke-Howson algorithm** is a combinatorial pivoting method for two-player games. It operates on a labeled polytope derived from the game's best-response conditions and follows a path of "complementary" vertices—pairs of vertices from the two players' strategy polytopes that satisfy complementary slackness conditions. The path begins at the trivial equilibrium (or a designated starting vertex) and terminates at a Nash equilibrium.

Lemke-Howson is analogous to the simplex method for linear programming: it follows edges of a polytope and is guaranteed to terminate at a Nash equilibrium. In practice it is often fast, but Savani and von Stengel (2006) showed that the worst-case number of pivoting steps is exponential. Despite this, the algorithm remains one of the most widely used methods for moderate-sized two-player games.

1.3 PPAD-Completeness

Definition: PPAD

PPAD (*Polynomial Parity Argument, Directed version*) is the complexity class of total search problems reducible to End-of-the-Line: given an exponentially large directed graph (specified by polynomial-time successor and predecessor circuits) with a known source, find another endpoint of a directed path. The existence of such a vertex is guaranteed by a directed parity argument: in any directed graph where each vertex has in-degree and out-degree at most one, a known source (in-degree 0, out-degree 1) implies the existence of a distinct sink (in-degree 1, out-degree 0). This mirrors the structure of Brouwer's fixed-point theorem that underlies Nash's existence proof.

Theorem: PPAD-Completeness of Nash Equilibrium (Daskalakis, Goldberg, Papadimitriou 2009; Chen, Deng 2009)

Finding a Nash equilibrium in a two-player game is **PPAD-complete**. That is, the problem of computing any NE of a given bimatrix game is polynomially equivalent to every other problem in PPAD. In particular, unless PPAD $\subseteq$ P (which is widely believed to be false), there is no polynomial-time algorithm for computing Nash equilibria, even in two-player games.

It is important to understand what PPAD-completeness does and does not say. It says that *finding* any Nash equilibrium is computationally hard in the worst case. It does *not* say that verifying whether a given strategy profile is a Nash equilibrium is hard—verification is easy, requiring only a check of best-response conditions. The hardness lies entirely in the search.

1.4 LP for Zero-Sum Games

Zero-sum games are the great exception to the computational hardness of Nash equilibrium. Recall von Neumann's minimax theorem from Lecture 3: in a two-player zero-sum game, $\max_{\sigma_1} \min_{\sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_2} \max_{\sigma_1} u_1(\sigma_1, \sigma_2)$. This equality is precisely the statement of strong LP duality.

Theorem: Minimax Theorem via LP Duality

Let $A \in \mathbb{R}^{n \times m}$ be the payoff matrix for the row player in a two-player zero-sum game. The row player's maxmin problem is the primal LP:

$$\max_{x,v} \; v \quad \text{s.t.} \quad A^\top x \geq v\mathbf{1}, \quad \mathbf{1}^\top x = 1, \quad x \geq 0,$$

and the column player's minimax problem is the dual LP:

$$\min_{y,w} \; w \quad \text{s.t.} \quad Ay \leq w\mathbf{1}, \quad \mathbf{1}^\top y = 1, \quad y \geq 0.$$

By strong duality, both LPs have the same optimal value $v^* = w^*$, and the optimal solutions (x^*, y^*) constitute a Nash equilibrium.

Since linear programs can be solved in polynomial time (via the ellipsoid method or interior-point methods), computing a Nash equilibrium in a two-player zero-sum game is a polynomial-time problem. This stands in stark contrast to the PPAD-completeness of general-sum games and is one of the most elegant results in algorithmic game theory.

Common Pitfalls

- PPAD-completeness says finding *any* NE is hard. It does not say that *all* NE are hard to verify—verification is easy (just check best responses).
- The LP approach only works for zero-sum games. For general-sum games, LPs give correlated equilibria, not Nash equilibria.
- Do not confuse "PPAD-complete" with "NP-complete." PPAD is a subclass of TFNP (total function NP), and PPAD-complete problems are guaranteed to have solutions. NP-complete decision problems may have no solution.

Connection: PA1

PA1 uses support enumeration for small games. Understanding why support enumeration is exponential in general motivates the regret-minimization approach developed in Lectures 7–9.

2 Lecture 6: Computing Equilibria II (69 min)

This lecture deepens the computational picture in two directions: computing correlated equilibria (which, unlike Nash equilibria, can be found in polynomial time) and approximating Nash equilibria (which relaxes the exact-equilibrium requirement to sidestep PPAD-hardness). The practical implications are significant: if we are willing to accept a correlated equilibrium or an approximate Nash equilibrium, efficient algorithms exist.

2.1 LP Formulation for Zero-Sum Games (Detailed)

We revisit the LP formulation from Lecture 5 in more detail. Consider a two-player zero-sum game with payoff matrix $A \in \mathbb{R}^{n \times m}$. The row player chooses a mixed strategy $x \in \Delta_n$ (the n-simplex), and the column player chooses $y \in \Delta_m$. The row player's expected payoff is $x^\top A y$.

The row player's problem is to maximize the guaranteed payoff:

$$\max_{x \in \Delta_n} \min_{y \in \Delta_m} x^\top A y.$$

Because $x^\top Ay$ is linear in y for fixed x, the inner minimum over the simplex is achieved at a vertex, so $\min_{y\in\Delta_m} x^\top Ay = \min_{j\in[m]}(A^\top x)_j$. Writing v for this minimum, the problem becomes the LP presented in the previous lecture. The column player's dual LP is obtained symmetrically.

2.2 Computing Correlated Equilibria via LP

Example: CE as a Linear Feasibility Problem

For a two-player game with strategy sets $S_1 = \{1,\dots,n\}$ and $S_2 = \{1,\dots,m\}$, a correlated equilibrium is a distribution $p \in \Delta(S_1 \times S_2)$ satisfying: for every player i and every pair of strategies $s_i, s_i' \in S_i$,

$$\sum_{s_{-i}} p(s_i, s_{-i})\big[u_i(s_i, s_{-i}) - u_i(s_i', s_{-i})\big] \geq 0.$$

These are $n(n-1) + m(m-1)$ linear inequalities in nm variables. The set of all correlated equilibria is therefore a convex polytope, and any linear objective (e.g., maximize social welfare $\sum_i u_i$, minimize maximum regret) can be optimized over it via a single LP.

The polynomial-time computability of correlated equilibria stands in sharp contrast to the PPAD-completeness of Nash equilibria. This computational advantage is one of the major motivations for adopting CE as a solution concept, especially in large games where Nash computation is infeasible.

2.3 Approximate Nash Equilibrium

Definition: ε-Nash Equilibrium

A strategy profile σ is an **ε-Nash equilibrium** (ε-NE) if no player can gain more than ε by unilateral deviation:

$$u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma_i', \sigma_{-i}) - \varepsilon \quad \text{for all } i \in N,\ \sigma_i' \in \Delta(S_i).$$

Approximate equilibria are practically motivated: in real strategic interactions, small gains from deviation may not justify the cost of reoptimization. They are also theoretically motivated by the following structural result.

Theorem: Lipton-Markakis-Mehta (2003)

Every two-player $n \times n$ game has an ε-Nash equilibrium in which each player's mixed strategy has support of size at most $k = O\left(\frac{\log n}{\varepsilon^2}\right)$. Consequently, an ε-NE can be found by exhaustive search over supports of size k in time $n^{O(\log n/\varepsilon^2)}$, which is quasi-polynomial.

This result implies that for any constant $\varepsilon > 0$, the problem of finding an ε-NE lies in quasi-polynomial time. Whether it can be solved in polynomial time remains a major open question. For the specific case of constant-factor approximation, the Tsaknakis-Spirakis algorithm finds a 0.3393-NE in polynomial time for two-player games.

2.4 Gambit and Practical Tools

The **Gambit** software suite provides practical implementations of multiple equilibrium-computation algorithms: support enumeration, Lemke-Howson, the global Newton method (Govindan-Wilson), and heuristic methods such as iterated polymatrix approximation. For games of moderate size

($n, m \leq 20$–50), Gambit can compute exact Nash equilibria. For larger games, one typically resorts to approximation algorithms or moves to structured-game representations (graphical games, games with bounded treewidth, or the extensive-form games of Week 3).

Common Pitfalls

- CE computation is polynomial via LP, while NE computation is PPAD-complete. Do not conflate the two: an LP over the CE polytope does *not* produce a Nash equilibrium in general.
- An ε-NE is not the same as a NE of a perturbed game. The approximation is in the deviation payoff, not in the game structure.
- The Lipton-Markakis-Mehta bound gives quasi-polynomial time, not polynomial time. The distinction matters for complexity-theoretic implications.

Connection: PA1

PA1 Tasks 1–5 cover the foundational material that support enumeration builds upon: computing expected utilities, checking best responses, verifying Nash equilibria, detecting dominated strategies, and verifying correlated equilibrium constraints. Completing PA1 will solidify your understanding of why the indifference conditions yield linear systems and why the procedure is exponential in the number of strategies.

3 Lecture 7: Regret Minimization I (65 min)

This lecture marks a major conceptual shift. Rather than computing equilibria directly (a PPAD-hard problem for general-sum games), we approach equilibria indirectly through the lens of *online learning*. The key insight is that when all players independently minimize their own regret, the resulting joint play converges to an equilibrium—without any player needing to solve the game explicitly. This connection between learning and equilibrium is one of the deepest results in algorithmic game theory.

3.1 The Online Decision-Making Model

The online learning framework models a repeated interaction between a learner (or decision-maker) and an environment. At each round $t = 1, 2, \ldots, T$:

1. The learner selects an action $a_t \in [K] = \{1, \ldots, K\}$ (or, more generally, a distribution $\sigma_t \in \Delta([K])$).
2. The environment reveals a loss vector $\ell_t \in [0, 1]^K$, and the learner incurs loss $\ell_t(a_t)$.

Crucially, the environment may be *adversarial*: it can choose ℓ_t based on the learner's past actions. This adversarial model is essential for the connection to game theory, where the "environment" is the collection of other players who are also strategic agents.

3.2 External Regret

Definition: External Regret

The **external regret** of a sequence of actions $a_1, \ldots, a_T$ is the difference between the learner's cumulative loss and the cumulative loss of the best fixed action in hindsight:

$$R_T = \sum_{t=1}^{T} \ell_t(a_t) - \min_{a^* \in [K]} \sum_{t=1}^{T} \ell_t(a^*).$$

An algorithm is **no-regret** (or **Hannan consistent**) if $R_T/T \to 0$ as $T \to \infty$, i.e., the average regret vanishes. Equivalently, the algorithm achieves sublinear regret: $R_T = o(T)$.

External regret compares the learner to the best *fixed* action. It does not account for the possibility that different actions might have been best at different times. Despite this seemingly weak benchmark, external regret is powerful enough to drive convergence to coarse correlated equilibria (see Section 4.3).

A fundamental information-theoretic lower bound states that any algorithm faces $R_T = \Omega(\sqrt{T \ln K})$ against a worst-case adversary. MWU matches this bound up to a constant factor of 2.

3.3 From Halving to Weighted Majority

The **halving algorithm** is the simplest no-regret method: it maintains a set of "consistent" experts (actions that have not yet erred) and predicts with the majority. If one expert is perfect, the halving algorithm makes at most $\log_2 K$ mistakes. However, it fails when no expert is perfect.

The **Weighted Majority algorithm** generalizes halving by assigning weights to experts and multiplicatively downweighting those that make mistakes, rather than eliminating them entirely. This yields robustness: even when all experts err occasionally, the algorithm's cumulative loss remains competitive with the best expert.

3.4 Multiplicative Weights Update (MWU)

MWU extends Weighted Majority to handle fractional losses (rather than binary mistakes). It is one of the most fundamental algorithms in theoretical computer science, with applications ranging from game theory to combinatorial optimization to machine learning.

Definition: MWU Update Rule

The **Multiplicative Weights Update** algorithm maintains a weight vector $w_t \in \mathbb{R}^K_{>0}$. Starting with $w_1(a) = 1$ for all $a \in [K]$, at each round t:

1. Play the distribution $\sigma_t(a) = w_t(a) / \sum_{a'} w_t(a')$.
2. Observe the loss vector $\ell_t \in [0,1]^K$.
3. Update weights: $w_{t+1}(a) = w_t(a) \cdot (1 - \eta\, \ell_t(a))$, where $\eta \in (0,1)$ is the learning rate.

An equivalent formulation uses the exponential update $w_{t+1}(a) = w_t(a) \cdot e^{-\eta\, \ell_t(a)}$.

Algorithm 1 Multiplicative Weights Update (MWU)

Require: Learning rate $\eta \in (0, 1)$; number of actions K; horizon T

Initialize $w_1(a) \leftarrow 1$ for all $a \in [K]$
for $t = 1, 2, \ldots, T$ **do**
  $W_t \leftarrow \sum_{a=1}^{K} w_t(a)$
  $\sigma_t(a) \leftarrow w_t(a)/W_t$ for all $a \in [K]$
  Play action $a_t \sim \sigma_t$ (or play the distribution σ_t)
  Observe loss vector $\ell_t \in [0, 1]^K$
  for $a = 1, \ldots, K$ **do**
    $w_{t+1}(a) \leftarrow w_t(a) \cdot (1 - \eta\, \ell_t(a))$
  end for
end for

Theorem: MWU Regret Bound

For losses $\ell_t \in [0, 1]^K$ and learning rate $\eta \in (0, 1)$, the external regret of MWU satisfies:

$$R_T \leq \eta\, T + \frac{\ln K}{\eta}.$$

Setting $\eta = \sqrt{\ln K / T}$ yields the optimal bound:

$$R_T \leq 2\sqrt{T \ln K}.$$

This matches the $\Omega(\sqrt{T \ln K})$ lower bound up to a constant factor of 2. The average regret is $R_T/T = O(\sqrt{\ln K/T})$, which vanishes as $T \to \infty$.

MWU can be viewed as an instance of **Follow the Regularized Leader (FTRL)** with the negative-entropy regularizer, or equivalently as *entropic mirror descent*. This connection places MWU within the broader framework of online convex optimization and reveals why the entropy regularizer is natural: it corresponds to the geometry of the probability simplex.

Common Pitfalls

- External regret compares to the best *fixed* action. It does not guarantee good performance against a shifting benchmark (that would require dynamic regret or adaptive regret notions).
- The adversary in the regret bounds can be adaptive—it can observe the learner's past actions before choosing the current loss vector. Some bounds require an oblivious adversary; always check the assumptions.
- The learning rate $\eta = \sqrt{\ln K/T}$ requires knowledge of the horizon T. For the anytime setting (unknown T), use the *doubling trick*: run MWU in epochs of geometrically increasing length, resetting η at each epoch.

Connection: Course Project

MWU is the foundation for several course project topics. Topic 1 (Convergence of RM Variants) involves implementing MWU alongside regret matching and comparing convergence rates. Understanding the MWU regret bound is essential background for any project involving online learning or equilibrium computation.

4 Lecture 8: Regret Minimization II (70 min)

This lecture develops two critical ideas. First, we introduce regret matching—a simpler alternative to MWU that is the workhorse algorithm behind Counterfactual Regret Minimization (CFR) in Week 3. Second, we establish the deep connection between no-regret learning and equilibrium concepts: no-external-regret play converges to coarse correlated equilibria, and no-swap-regret play converges to correlated equilibria.

4.1 Regret Matching

Definition: Regret Matching

The **regret matching** algorithm maintains cumulative regret values $r_t(a)$ for each action $a \in [K]$, representing how much better action a would have performed compared to the actions actually played. At each round, the algorithm plays proportionally to the positive part of the cumulative regrets:

$$\sigma_t(a) = \frac{[r_t(a)]^+}{\sum_{a'} [r_t(a')]^+},$$

where $[x]^+ = \max(0, x)$. If all cumulative regrets are non-positive (i.e., $\sum_{a'} [r_t(a')]^+ = 0$), the algorithm plays the uniform distribution over all actions.

Algorithm 2 Regret Matching

Require: Number of actions K; horizon T

Initialize $r_0(a) \leftarrow 0$ for all $a \in [K]$
for $t = 1, 2, \ldots, T$ **do**
  if $\sum_a [r_{t-1}(a)]^+ > 0$ **then**
    $\sigma_t(a) \leftarrow [r_{t-1}(a)]^+ \,/\, \sum_{a'} [r_{t-1}(a')]^+$ for all a
  else
    $\sigma_t(a) \leftarrow 1/K$ for all a
  end if
  Play action $a_t \sim \sigma_t$
  Observe loss vector $\ell_t \in [0,1]^K$
  for $a = 1, \ldots, K$ **do**
    $r_t(a) \leftarrow r_{t-1}(a) + \ell_t(\sigma_t) - \ell_t(a)$ ▷ $\ell_t(\sigma_t) = \sum_{a'} \sigma_t(a') \ell_t(a')$
  end for
end for

Regret matching achieves an $O(\sqrt{T})$ external regret bound (Hart & Mas-Colell, 2000). Note that this bound lacks the $\ln K$ factor that appears in the MWU bound of $O(\sqrt{T \ln K})$; the two algorithms have different dependence on the number of actions. Regret matching has a different character from MWU: it is a "lazy" algorithm that concentrates probability on the actions that have performed well relative to what was actually played, rather than using multiplicative updates on absolute weights. This makes it particularly well-suited for decomposition across information sets in extensive-form games (Week 3).

4.2 Internal Regret and Swap Regret

External regret asks: "Would I have done better by committing to a single fixed action?" This is the weakest notion in a natural hierarchy of regret concepts.

Definition: Internal Regret

The **internal regret** for a pair of actions (a, a') is the regret of not having switched from action a to action a' whenever action a was played:

$$R_T^{a \to a'} = \sum_{t=1}^{T} \sigma_t(a)\big[\ell_t(a) - \ell_t(a')\big].$$

An algorithm has **no internal regret** if $\max_{a,a'} R_T^{a \to a'} = o(T)$.

Definition: Swap Regret

The **swap regret** generalizes internal regret to arbitrary modification functions. For any function $F\colon [K] \to [K]$, the swap regret with respect to F is:

$$R_T^F = \sum_{t=1}^{T} \sum_{a=1}^{K} \sigma_t(a)\big[\ell_t(a) - \ell_t(F(a))\big].$$

An algorithm has **no swap regret** if $\max_{F\colon [K]\to[K]} R_T^F = o(T)$.

The hierarchy is strict: swap regret $\geq$ internal regret $\geq$ external regret. External regret corresponds to the special case of swap functions that are constant ($F(a) = a^*$ for all a). Internal regret corresponds to swap functions that modify at most one action. The full swap regret allows arbitrary remappings.

No-swap-regret algorithms can be constructed by combining K copies of a no-external-regret learner (one per action), using the technique of Blum and Mansour (2007). Each copy a learns a distribution over actions to "switch to" whenever action a is recommended; the overall strategy is obtained as the stationary distribution of the resulting Markov chain. The resulting swap regret is $O(K\sqrt{T \ln K})$.

4.3 Convergence to Equilibria

The central results of this lecture connect the regret hierarchy to the equilibrium hierarchy from Module 1.

Theorem: Convergence to CCE

If all n players in a finite game use no-external-regret algorithms with regret bound R_T, then the **empirical distribution of joint play**

$$\bar{p}(s) = \frac{1}{T} \sum_{t=1}^{T} \prod_{i=1}^{n} \sigma_t^i(s_i)$$

converges to an ε-coarse correlated equilibrium with $\varepsilon = \max_i R_T^i / T$. With MWU, $\varepsilon = O(\sqrt{\ln K / T})$; with regret matching, $\varepsilon = O(1/\sqrt{T})$.

Theorem: Convergence to CE

If all n players use no-swap-regret algorithms with swap regret bound R_T^{swap}, then the empirical distribution of joint play converges to an ε-correlated equilibrium with $\varepsilon = \max_i R_T^{\text{swap},i}/T$.

These theorems provide a *decentralized, polynomial-time* path to equilibrium computation: each player runs their own no-regret algorithm using only their own payoff observations, and the resulting joint play converges to a CCE or CE. No central coordinator is needed. The convergence is of the *empirical distribution* (the time-average of play), not of the last iterate—the last iterate can oscillate.

Common Pitfalls

- Regret matching gives no-*external*-regret guarantees, not no-swap-regret guarantees. For convergence to CE (not merely CCE), you need the more expensive no-swap-regret construction.
- Convergence is of the *average* strategy profile, not the *last iterate*. The last iterate can oscillate indefinitely, even as the average converges. This distinction is practically important when implementing these algorithms.
- With two no-regret learners in a *general-sum* game, the average play converges to a CCE but *not* necessarily to a Nash equilibrium. NE convergence requires additional structure (e.g., zero-sum games, as in Lecture 9).

Connection: PA1

PA1 covers Nash equilibrium verification and dominated strategy analysis for small games. The regret-minimization framework provides an alternative, scalable approach: rather than solving for NE directly, let learning dynamics converge to approximate equilibria. This contrast is the bridge from Module 1's exact methods to Module 2's iterative methods.

5 Lecture 9: Regret Minimization III (67 min)

The final lecture of this module exploits special structure—zero-sum games, algorithmic variants, and limited feedback—to push beyond the general-purpose results of Lectures 7–8. The culmination is the connection to Counterfactual Regret Minimization (CFR), the algorithmic backbone of modern poker AI, which will be the focus of Week 3.

5.1 Zero-Sum Convergence to Nash Equilibrium

In general-sum games, no-regret learning converges only to CCE (or CE with no-swap-regret). Zero-sum games are special: the minimax theorem ensures that CCE, CE, and NE all coincide in terms of their marginal strategies. This yields a much stronger convergence result.

Theorem: Zero-Sum NE Convergence

If both players in a two-player zero-sum game use no-external-regret algorithms with regret bound R_T, then their **average strategies** $\bar{\sigma}_i = \frac{1}{T}\sum_{t=1}^{T} \sigma_t^i$ form an ε-Nash equilibrium with $\varepsilon = (R_T^1 + R_T^2)/T$. With MWU or regret matching, this gives an ε-NE after $O(1/\varepsilon^2)$ rounds of play.

This result provides a simple, decentralized algorithm for computing approximate Nash equilibria

in zero-sum games: both players run MWU (or regret matching) independently, and their average strategies converge. The convergence rate of $O(1/\varepsilon^2)$ rounds for an ε-NE is polynomial in $1/\varepsilon$ and logarithmic in the number of strategies.

5.2 RM+ (Regret Matching Plus)

Definition: RM+ (Regret Matching Plus)

Regret Matching Plus (RM+) is a variant of regret matching that *clips* cumulative regrets at zero after each update, rather than allowing them to become negative:

$$r_t(a) = \left[r_{t-1}(a) + \ell_t(\sigma_t) - \ell_t(a)\right]^+.$$

The strategy is then computed as in standard regret matching: $\sigma_{t+1}(a) \propto [r_t(a)]^+$. The clipping prevents regret from accumulating large negative values that would delay recovery, leading to faster empirical convergence.

Algorithm 3 Regret Matching Plus (RM+)

Require: Number of actions K; horizon T

Initialize $r_0(a) \leftarrow 0$ for all $a \in [K]$
for $t = 1, 2, \ldots, T$ **do**
  if $\sum_a r_{t-1}(a) > 0$ **then**
    $\sigma_t(a) \leftarrow r_{t-1}(a) \,/\, \sum_{a'} r_{t-1}(a')$ for all a
  else
    $\sigma_t(a) \leftarrow 1/K$ for all a
  end if
  Play action $a_t \sim \sigma_t$
  Observe loss vector $\ell_t \in [0,1]^K$
  for $a = 1, \ldots, K$ **do**
    $r_t(a) \leftarrow \left[r_{t-1}(a) + \ell_t(\sigma_t) - \ell_t(a)\right]^+$ ▷ Clip at zero
  end for
end for

RM+ maintains the same theoretical $O(\sqrt{T})$ regret bound as standard regret matching, but empirically converges 10–100× faster on poker games. Tammelin (2014) introduced RM+ as the per-information-set algorithm in **CFR+**, which solved heads-up limit Texas hold'em. The practical speedup comes from the clipping: in standard RM, an action that performs poorly early on accumulates large negative regret and must "dig out" of this deficit before it can be played again, even if conditions have changed. RM+ avoids this by resetting negative regrets to zero.

5.3 Last-Iterate vs. Average-Iterate Convergence

Standard no-regret algorithms guarantee convergence of the *average* strategy to an equilibrium. The *last iterate*—the strategy actually played at time T—may oscillate indefinitely, cycling through the strategy space without settling down. This is a significant practical limitation: if you need to deploy a single strategy at the end of training, the average iterate requires storing and updating the running average throughout.

Recent work has shown that *optimistic* variants of no-regret algorithms can achieve last-iterate convergence in zero-sum games, eliminating the need for averaging.

Definition: Optimistic MWU (OMWU)

Optimistic Multiplicative Weights Update modifies MWU by incorporating a prediction of the current loss based on the previous loss. The update rule becomes:

$$\hat{w}_{t+1}(a) = w_t(a) \cdot e^{-\eta\, \ell_t(a)},$$
$$w_{t+1}(a) = \hat{w}_{t+1}(a) \cdot e^{-\eta\, \ell_t(a)},$$

where the loss ℓ_t is used as a predictor for ℓ_{t+1}. The key idea is that in a game, the opponent's strategy changes slowly, so the previous loss is a good predictor of the current loss.

OMWU achieves $O(1/T)$ last-iterate convergence to Nash equilibrium in two-player zero-sum games, compared to the $O(1/\sqrt{T})$ rate for average-iterate convergence with standard MWU. This quadratic speedup in the convergence rate, combined with the elimination of averaging, makes OMWU the method of choice when last-iterate convergence is needed.

5.4 Bandit Feedback and EXP3

All algorithms discussed so far assume **full-information feedback**: after each round, the learner observes the entire loss vector $\ell_t \in [0,1]^K$. In many applications—including games where a player observes only their own payoff, not the payoffs of unchosen actions—the learner has only **bandit feedback**: it observes $\ell_t(a_t)$ for the action actually played.

Definition: EXP3

The **EXP3** (*Exponential-weight algorithm for Exploration and Exploitation*) algorithm handles bandit feedback by constructing unbiased estimates of the full loss vector using importance weighting:

$$\hat{\ell}_t(a) = \frac{\ell_t(a) \cdot \mathbf{1}[a_t = a]}{\sigma_t(a)},$$

and then applying MWU to these estimated losses. To ensure bounded variance, EXP3 mixes the MWU distribution with a uniform exploration component.

EXP3 achieves regret $O(\sqrt{TK \ln K})$—worse than the full-information bound of $O(\sqrt{T \ln K})$ by a factor of $\sqrt{K}$. This $\sqrt{K}$ penalty is the price of partial information: with K actions and bandit feedback, each action is observed only a $1/K$ fraction of the time, and the importance-weighted estimator has variance proportional to K.

5.5 Preview: Counterfactual Regret Minimization (CFR)

The bridge to Week 3 is Counterfactual Regret Minimization (CFR), which applies regret matching independently at each information set of an extensive-form game. The **decomposition theorem** of Zinkevich et al. (2007) shows that if the regret at each information set is sublinear, then the overall regret in the game is sublinear. Combined with the zero-sum NE convergence theorem, this gives a polynomial-time algorithm for computing ε-Nash equilibria in two-player zero-sum extensive-form games.

The connection is: CFR = regret matching (or RM+) at each information set + the decomposition theorem. Week 3 will develop CFR in full detail, including its application to poker.

Common Pitfalls

- In zero-sum games, no-regret learners converge to NE in the average iterate. In general-sum games, they converge only to CCE—not to NE. Do not assume NE convergence without the zero-sum assumption.
- RM+ clips regrets at zero, which improves practical convergence but does not change the worst-case theoretical bound. The speedup is empirical, not asymptotic.
- Last-iterate convergence requires optimistic algorithms (OMWU) or other modifications. Standard MWU and regret matching do *not* exhibit last-iterate convergence, even in zero-sum games.
- EXP3's $\sqrt{K}$ penalty is unavoidable with bandit feedback. If the full loss vector is available, always prefer full-information algorithms.

Connection: Course Project

Topics 1 (Convergence of RM Variants) and 9 (Equilibrium Computation at Scale) directly build on this module's material. If choosing Topic 1, you would implement RM, RM+, and predictive RM (or OMWU) and compare their convergence rates on standard games (e.g., Kuhn poker, Leduc poker). The empirical 10–100× speedup of RM+ over RM is one of the key phenomena to reproduce and analyze.

Key Algorithms Summary

Algorithm	Setting	Regret Bound	Notes
MWU	Full info, adversarial	$O(\sqrt{T \ln K})$	Tight; instance of FTRL with entropy regularizer
Regret Matching	Full info, adversarial	$O(\sqrt{T})$	Simpler; used as the base algorithm in CFR
RM+	Full info, adversarial	$O(\sqrt{T})$	Clips negative regrets; 10–100× faster in practice
Optimistic MWU	Full info, zero-sum	$O(1/T)$ last-iterate	Requires predictability; quadratic speedup over average iterate
EXP3	Bandit, adversarial	$O(\sqrt{TK \ln K})$	$\sqrt{K}$ penalty for partial information

Key Takeaways

1. Computing Nash equilibria is PPAD-complete for general-sum games but polynomial for zero-sum games (via LP duality). Correlated equilibria are always polynomial-time computable via LP.
2. The Multiplicative Weights Update algorithm achieves $O(\sqrt{T \ln K})$ external regret, matching the information-theoretic lower bound up to constants.
3. No-external-regret play converges to coarse correlated equilibria; no-swap-regret play con-

verges to correlated equilibria. In zero-sum games, no-external-regret play converges to Nash equilibria.
4. RM+ (regret matching with clipping) provides dramatic empirical speedups over standard regret matching and is the engine behind CFR+ for solving extensive-form games (Week 3).
5. The regret hierarchy (external $\leq$ internal $\leq$ swap) mirrors the equilibrium hierarchy (NE $\subseteq$ CE $\subseteq$ CCE), creating a unified framework for learning and equilibrium.

Suggested Reading

- Cesa-Bianchi & Lugosi: *Prediction, Learning, and Games*, Chapters 2–4.
- Nisan, Roughgarden, Tardos, Vazirani: *Algorithmic Game Theory*, Chapter 4 (on computation of equilibria).
- Hart & Mas-Colell (2000): "A Simple Adaptive Procedure Leading to Correlated Equilibrium," *Econometrica*.