Module 9 — Learning In Games

Game Theory

hbar.university

Fictitious Play

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.1

Fictitious play is the oldest learning algorithm in game theory and the first procedure by which rational agents who know nothing about each other's internal strategies can converge on a Nash equilibrium through repeated play alone. Introduced by George W. Brown in 1951 as a computational method for solving zero-sum games — it was designed as an algorithm, not as a behavioral model — it turned out to have surprising structural properties that made it the starting point for the entire learning-in-games program. The rule is radical in its minimalism: each player maintains a running count of the opponent's historical actions, treats those counts as a probability distribution, and plays a best response against that distribution. No reasoning about the opponent's rationality, no mutual consistency, no equilibrium calculation — just count and respond. Yet in two-player zero-sum games, this local rule provably produces play whose *empirical* frequencies converge to a Nash equilibrium. Fictitious play is the prototype for every learning dynamic that follows: no-regret algorithms, CFR, replicator dynamics, and reinforcement learning in games all inherit its core question — *what happens when boundedly rational agents adapt to each other over time?*

Formal Definition

Let $G = (N, \{S_i\}, \{u_i\})$ be a finite normal-form game played repeatedly at times $t = 1, 2, 3, \ldots$. Let $s_i^t \in S_i$ denote the action player i plays at time t.

Empirical frequency. For each player j and action $s_j \in S_j$, define the empirical frequency of s_j through time t:

$$\bar{\sigma}_j^t(s_j) \;=\; \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{1}[s_j^\tau = s_j].$$

This is a probability distribution $\bar{\sigma}_j^t \in \Delta(S_j)$ — the running histogram of player j's plays.

Fictitious play rule. At every time $t+1$, player i chooses a best response to the profile of opponent empirical frequencies $\bar{\sigma}_{-i}^t$:

$$s_i^{t+1} \;\in\; B_i\big(\bar{\sigma}_{-i}^t\big) \;=\; \arg\max_{s_i \in S_i} \; u_i\big(s_i, \bar{\sigma}_{-i}^t\big)\,.$$

Ties are broken by an arbitrary fixed rule. The initial condition $\bar{\sigma}_{-i}^0$ is a specified prior (commonly a unit count on some action for each opponent).

Bayesian interpretation. The rule has a clean probabilistic reading: player i believes the opponents are playing a stationary mixed strategy σ_{-i}, starts with a Dirichlet prior, updates the posterior by counting, and plays the Bayes-optimal response. The empirical frequency is the posterior mean. The definition does not need this interpretation, but it clarifies the model of the other players that fictitious play implicitly holds: they are stationary random processes rather than adaptive strategic agents.

Convergence concepts. Three distinct senses of "convergence to Nash":

1. **Empirical convergence:** $\bar{\sigma}^t$ converges (in $\prod_i \Delta(S_i)$) to a Nash equilibrium σ^*.
2. **Action convergence:** s^t eventually stabilizes on a Nash equilibrium profile.
3. **Time-averaged payoff convergence:** $\bar{u}_i^t = \frac{1}{t}\sum_\tau u_i(s^\tau)$ converges to the Nash payoff.

For fictitious play, empirical convergence holds in important classes of games, while action convergence typically fails (players keep cycling even when the frequencies settle).

Intuition

Three lenses on fictitious play, each illuminating a different facet.

The statistician. Player i is doing naive Bayesian inference about a stationary but unknown opponent. Each opponent is treated as a random process whose parameter is being estimated from the data. Fictitious play is the Laplace-style point estimate: use the empirical histogram as your best guess for the underlying mixed strategy, then optimize against it. The flaw — obvious but essential — is that the opponent is not actually a stationary process. The opponent is doing the same thing you are, and their histogram evolves in response to your play. The fictitious-play agent ignores this dependence. Remarkably, in zero-sum games this mis-specification does not prevent convergence of frequencies; in general games, it sometimes does.

The optimizer. Fictitious play is a *best-response dynamic* with memory. At each step you act optimally against the *cumulative* distribution of opponent actions, not just the most recent. This averaging across history is the structural feature that distinguishes fictitious play from simple best-response dynamics (the latter can cycle arbitrarily, e.g., in matching pennies, while the former's *averages* settle).

The computer. Brown's original motivation was numerical: use fictitious play as an iterative method for computing equilibria of zero-sum games, as an alternative to solving the LP directly. For a matrix game A, the fictitious-play iterate computes a sequence of row/column plays whose averages converge to the game's value. Brown and Robinson (1951) proved the convergence for zero-sum games, and the rate is polynomial but slow. This slowness is one reason no-regret methods (Node 9.2) and CFR (Node 9.3) eventually displaced fictitious play as practical equilibrium-computation tools.

The deep lesson of fictitious play is that *consistency is not rationality.* Each player is doing something locally reasonable — best-responding to a frequency count — but the joint behavior only aligns with Nash under specific structural conditions on the game (zero-sum, dominance-solvable, certain potential games). In the wrong games (Shapley's 3x3 counterexample), fictitious play *provably* cycles and never converges. The theoretical frontier of learning in games is carved out by asking: *which games force learning dynamics to reach equilibrium, and which do not?*

Structural Role

Fictitious play sits at the root of Module 9 as the ancestor of every learning dynamic:

- **9.2 No-regret learning** generalizes the best-response-to-empirical rule to algorithms with regret guarantees (fictitious play is not no-regret in general, which is a striking fact).
- **9.3 Counterfactual regret minimization** lifts regret-matching to extensive-form games and is the poker-solving workhorse.

- **9.4 Multiplicative weights** replaces pure best response with an exponentiated-weights soft maximum, trading off determinism for no-regret guarantees.
- **9.5 Convergence to equilibrium** catalogs when learning dynamics reach Nash and when they cycle — Shapley's counterexample to fictitious play is the motivating negative result.
- **9.6 Reinforcement learning in games** replaces the "know-the-payoff-function" assumption of fictitious play with payoff *samples*, connecting to Q-learning and actor-critic methods.

Fictitious play is also the simplest nontrivial example of a dynamic on $\prod_i \Delta(S_i)$, and its analysis introduces the differential-inclusion formalism used throughout the literature on continuous-time learning dynamics.

Mathematical Development

Continuous-time formulation

A cleaner analytical setting: pretend the time index is continuous. Let $\bar{\sigma}_i(t) \in \Delta(S_i)$ denote the empirical frequency, evolving by

$$\frac{d\bar{\sigma}_i}{dt} \;\in\; \frac{1}{t}\Big(B_i(\bar{\sigma}_{-i}(t)) - \bar{\sigma}_i(t)\Big),$$

where B_i is the (possibly set-valued) best-response correspondence. This is a *differential inclusion*, because B_i can be multi-valued. After a time rescaling $\tau = \ln t$, the dynamics become autonomous:

$$\frac{d\bar{\sigma}_i}{d\tau} \;\in\; B_i(\bar{\sigma}_{-i}) - \bar{\sigma}_i.$$

This is the **best-response dynamic**, a continuous-time limit of fictitious play. Its fixed points are exactly the Nash equilibria (because at a fixed point $\bar{\sigma}_i \in B_i(\bar{\sigma}_{-i})$ for every i), and its trajectories are the analytical tool for studying convergence.

Brown–Robinson theorem (zero-sum convergence)

Theorem (Brown 1951, Robinson 1951). In any finite two-player zero-sum game, fictitious play converges in empirical frequencies to the set of Nash equilibria. Specifically, the players' time-averaged payoffs converge to the value of the game.

Proof sketch. Let V be the value of the game. Define the minimax gap at time t:

$$\Delta^t \;=\; \max_{s_1} u_1(s_1, \bar{\sigma}_2^t) - \min_{s_2} u_1(\bar{\sigma}_1^t, s_2).$$

This is the "exploitability" of the empirical profile. By the fictitious-play rule, each player at each step picks an action that is optimal against the opponent's empirical frequency. Robinson's argument shows $\Delta^t \to 0$ by a careful induction on the matrix dimensions, using a potential function that tracks the difference between upper and lower envelopes. The convergence rate is polynomial but slow:

$$\Delta^t \;=\; O\Big(t^{-1/(m+n-2)}\Big)$$

for an $m \times n$ matrix game. For even moderately sized games this rate is impractical, which is why no-regret methods eventually replaced fictitious play in computational work.

Zero-sum is special

The Brown–Robinson proof relies heavily on the zero-sum structure. For general-sum games, fictitious play can diverge. The next results carve out the positive classes and illustrate the negative boundary.

Convergent classes beyond zero-sum

- **Dominance-solvable games.** In games solvable by iterated strict dominance, fictitious play converges to the unique surviving profile. Miyasawa (1961) proved convergence for all 2x2 games.
- **Games with a potential.** In a finite game admitting a potential function $\Phi : S \to \mathbb{R}$ such that unilateral deviations change each player's payoff by the same amount as the change in Φ, fictitious play converges to a Nash equilibrium (Monderer and Shapley 1996). This class includes coordination games and congestion games.
- **Supermodular games.** Many classes of games with strategic complementarities also admit convergence results.

Shapley's counterexample (1964)

Consider the 3x3 game with payoff bimatrix

	L	C	R
T	$(0,0)$	$(1,0)$	$(0,1)$
M	$(0,1)$	$(0,0)$	$(1,0)$
B	$(1,0)$	$(0,1)$	$(0,0)$

This is a non–zero-sum variant of Rock-Paper-Scissors. The unique Nash equilibrium is the uniform mixture $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ for both players. But **fictitious play does not converge** to this equilibrium from most initial conditions. Shapley showed the empirical frequencies cycle forever, with the sequence of best-response regions traversed in a rotational pattern: each player's best response chases the other's shifting frequency around the simplex, and the frequencies themselves trace out a limit cycle that fails to reach the centroid.

The Shapley counterexample is the canonical negative result of learning in games. It shows that "everybody best-responds to empirical opponent play" is not enough to guarantee convergence outside of special game classes — and it motivates all the structural refinements studied in 9.2–9.5.

The miscoordination problem

Even in games where fictitious play converges in empirical frequencies, *action* convergence typically fails. Consider Matching Pennies: the Nash equilibrium is $(\frac{1}{2}, \frac{1}{2})$ for both. Under fictitious play, each player alternately best-responds to whichever action their opponent has played more often, causing both players to flip their actions in a predictable pattern. The empirical frequencies converge to $\frac{1}{2}$ but the players never settle on a pure action profile. This is fine when all you care about is the distribution of play, but for "predicting what will be played next turn" fictitious play offers no consolation.

Worked Example

Fictitious play in Matching Pennies

Row chooses Heads (H) or Tails (T). Column chooses H or T. Row wins +1 on match, Column wins +1 on mismatch. Initialize Row's belief about Column as (H: 0, T: 1); initialize Column's belief about Row as (H: 1, T: 0).

Each round, each player best-responds to the current empirical belief and the count is updated. Row best-responds to Column = T by playing T; Column best-responds to Row = H by playing T. Then Row updates Column's histogram to (H: 0, T: 2), still best-responds with T; Column updates Row's histogram to (H: 1, T: 1), ties — say plays H. And so on.

Tracing the trajectory, each player follows the opponent's empirical frequency with a lag. Row eventually switches from T to H, Column eventually switches from H to T, and so on. The empirical frequencies $\bar{\sigma} \to (\frac{1}{2}, \frac{1}{2})$ for both — which is the Nash equilibrium — but the actions themselves form an infinite alternating sequence that never stabilizes.

Fictitious play in a dominance-solvable game

Consider the Prisoner's Dilemma with payoffs $(C, C) = (3, 3), (C, D) = (0, 5), (D, C) = (5, 0), (D, D) = (1, 1)$. Defect is strictly dominant for both players. Whatever the initial empirical counts, Row's best response is always D (because $u_1(D, \cdot) > u_1(C, \cdot)$ for every opponent distribution). Same for Column. So from the first step, $s_1^t = s_2^t = D$ for every t, and the empirical frequencies converge instantly to (D, D), the dominance-solvable Nash equilibrium.

Fictitious play in Shapley's 3x3 counterexample

Initialize with unit weight on (T, L). Row sees Column's L and best-responds with B. Column sees Row's T and best-responds with C. Next round: (B, C). Row sees Column's (L: 1, C: 1) and best-responds with M (tie-breaking aside). Column sees Row's (T: 1, B: 1) and best-responds with R. And so on.

Tracing the trajectory, the empirical frequencies move in a rotational pattern around the uniform distribution but never settle there. The limit set is a cycle, not a point. Simulations show the cycle persists indefinitely with amplitude that does not shrink. This is the structural reason fictitious play cannot serve as a universal equilibrium-computation method: for games like Shapley's, the rule is simply wrong.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Empirical frequency $\bar{\sigma}_j^t$	Your HUD stats on an opponent — VPIP, PFR, 3-bet%, c-bet% — running averages of observed actions
Best response $B_i(\bar{\sigma}_{-i}^t)$	The maximally exploitative strategy against the opponent's HUD profile
Fictitious play update	After each hand, update HUD stats and recompute your exploitative plan
Convergence to Nash (zero-sum)	In a heads-up match, repeated exploitation by both players drives play toward GTO
Shapley-style cycle	Two opponents stuck in a rock-paper-scissors loop: each "adjustment" spawns a counter-adjustment that leaves the cycle

Concrete situation: reg-vs-reg HUD-based adjustment

You play long sessions against a pool of regulars at mid stakes. You keep HUD stats on each regular: their preflop frequencies, c-bet frequencies, check-raise frequencies, etc. Over time these become reliable estimates of their mixed strategies at various nodes. Your own strategy is a best response to those running statistics, updated continuously as you see more hands.

This is exactly fictitious play. Your action each hand is the best response to the opponent's empirical frequencies aggregated over all prior observations. The opponents, similarly, are observing your frequencies and adjusting. If the game were two-player zero-sum (it is, modulo rake), fictitious play theory says the empirical frequencies of both players would converge to Nash — the GTO solution — over sufficiently long play. In practice, convergence is slow (Brown–Robinson rate $O(t^{-1/(m+n-2)})$, which for a realistic game tree with hundreds of info sets is glacial), which is why actual poker learning uses CFR (9.3) offline rather than waiting for the mutual-exploitation dynamics to settle at the table.

Concrete situation: the "leveling war" failure mode

In heads-up matches between two aggressive regulars, a common failure of fictitious-play-like reasoning is the "leveling war": I exploit your high c-bet rate by check-raising more, you exploit my high check-raise rate by c-betting thinner, I exploit *that* by check-raising even more, and so on. This is precisely the structural failure mode that Shapley's counterexample demonstrates in the abstract: when the game's best-response dynamics generate cycles rather than attracting to a fixed point, mutual adjustment does not reach equilibrium — it produces an oscillation whose empirical average may or may not be Nash.

The practical upshot: if you are playing someone who adjusts faster than you, pure exploitation will lose because you chase their shifts without ever getting ahead of them. You want a strategy closer to the Nash, which does not require prediction.

What this understanding buys you

Your HUD stats are a fictitious-play prior on your opponent. Recognizing that "I use HUD stats to best-respond" is *literally fictitious play* tells you two things. First, under zero-sum assumptions with a stationary opponent, your empirical exploitation will converge toward optimal — slowly. Second, if your opponents are adapting in parallel, you cannot trust the convergence theorem, because the stationarity assumption is violated. In a reg pool where everyone studies, everyone is updating their strategies on the basis of everyone else's statistics, and the "empirical frequency" you see is a lagged picture of a moving target. Fictitious play's mathematical guarantee is no longer in force.

Static GTO baselines are the answer to non-stationary pools. If fictitious-play dynamics can cycle, what you want is a strategy whose guarantee does not depend on opponent stationarity. That strategy is Nash (GTO): it guarantees the game's value regardless of opponent behavior. This is why the shift from exploit-first to GTO-first poker coincided with the realization, at the theoretical level, that exploit-dynamics can fail to converge in non-zero-sum or non-stationary settings.

Cross-Links

- **1.4 Best response** — the core operation fictitious play iterates.
- **2.1 Nash equilibrium** — the target of convergence (when convergence holds).
- **2.3 Mixed-strategy equilibrium** — the empirical frequency is a mixed strategy.
- **9.2 No-regret learning** — the successor framework with stronger guarantees.
- **9.3 CFR** — the extensive-form descendant that dominates poker solving.
- **9.5 Convergence to equilibrium** — the general theory of which learning dynamics reach Nash.
- **Statistics 3.x Bayesian inference** — the Laplace-point-estimate interpretation.
- **Dynamical systems theory** — differential inclusions, limit cycles.

Structural Compression

Invariant: A player maintaining a running histogram of the opponent's actions and best-responding to the histogram generates an adaptive dynamic whose empirical frequencies converge to Nash equilibrium in two-player zero-sum and certain other structured games, but can cycle indefinitely in general games.

Mechanism: Treat opponent actions as samples from an unknown stationary distribution; estimate the distribution by the empirical histogram; play a best response to the estimate. Iterate. The continuous-time limit is the best-response differential inclusion.

Cross-System Echo: Fictitious play is a stateless version of Bayesian belief updating in a misspecified model: the agent assumes stationarity, estimates via Laplace point estimation, and optimizes greedily. This pattern — local rationality under a mis-specified generative model — recurs throughout statistics (empirical risk minimization), control (certainty-equivalence control), and ecology (optimal foraging under stationary

resource assumptions). In each case, the interesting phenomena happen when the mis-specification bites back, producing limit cycles, cascading failures, or equilibrium selection pathologies.

Exercises

1. Simulate fictitious play on the 2x2 matching-pennies game with initial beliefs $\bar{\sigma}_2^0 = (0.6, 0.4)$ and $\bar{\sigma}_1^0 = (0.3, 0.7)$. Verify that empirical frequencies approach $(\frac{1}{2}, \frac{1}{2})$ for both players while action sequences exhibit persistent alternation.

2. Prove Miyasawa's result: fictitious play converges to a Nash equilibrium in every finite 2x2 game. (Hint: enumerate the structural types — zero-sum, dominance-solvable, coordination, anti-coordination.)

3. Write the continuous-time best-response dynamic for Shapley's 3x3 counterexample and argue that there is a limit cycle around the uniform distribution. Compute the approximate period of the cycle from the best-response structure.

4. Show that in any finite potential game with potential Φ, the fictitious-play empirical frequencies converge to a Nash equilibrium. (Hint: Φ evaluated at the empirical profile is approximately monotone.)

5. Derive the Brown–Robinson convergence rate $O(t^{-1/(m+n-2)})$ for fictitious play on an $m \times n$ zero-sum game, using a bound on the minimax gap Δ^t. Compare to the $O(t^{-1/2})$ rate achievable by multiplicative-weights methods (Node 9.4).

6. Consider the 3x3 Rock-Paper-Scissors game (zero-sum). Prove that fictitious play converges to the uniform equilibrium. Why does the zero-sum structure rescue convergence that Shapley's non-zero-sum variant destroys?

7. **Argue, structurally,** why a learning rule that ignores its own effect on opponent play can converge in zero-sum games but not in general-sum games. What is the structural feature of zero-sum that makes "pretend the opponent is stationary" a viable approximation, and what breaks in general-sum?

No-Regret Learning

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.2

No-regret learning is the framework that repairs the defects of fictitious play by replacing "best-respond to empirical opponent play" with a weaker but more robust guarantee: *the learner should not regret, in hindsight, having played a fixed action for all time.* This shift from optimality-against-a-model to regret-minimization-against-an-arbitrary-sequence is the conceptual core of modern online learning theory, and it has two consequences for game theory that reshape the field. First, when *all* players in a finite game use no-regret algorithms, the joint empirical frequency of play converges to the set of *coarse correlated equilibria* — a set that strictly contains Nash in general but equals Nash in two-player zero-sum games. Second, the convergence rate is $O(T^{-1/2})$, exponentially faster than fictitious play's sluggish polynomial, which makes no-regret methods computationally viable. The connection between no-regret learning and equilibrium is one of the deepest structural results in the theory: it shows that a *very weak individual rationality guarantee* (no external regret) is sufficient for *collective convergence* to a well-defined equilibrium concept. Node 9.2 develops the regret framework, proves the fundamental convergence theorem, and introduces the Hannan consistency criterion that links individual learning to equilibrium theory.

Formal Definition

Fix a finite action set $A = \{1, 2, \ldots, K\}$ for a single learner. At each round $t = 1, 2, \ldots, T$, the learner picks an action $a^t \in A$ (possibly randomly), and then observes a loss vector $\ell^t \in [0,1]^K$, where ℓ^t_k is the loss that would have been incurred by action k. The learner suffers loss $\ell^t_{a^t}$.

External regret. The external regret with respect to a fixed action k is

$$R_k^T \;=\; \sum_{t=1}^{T} \ell^t_{a^t} - \sum_{t=1}^{T} \ell^t_k.$$

The **maximum external regret** is

$$R^T \;=\; \max_{k \in A} R_k^T \;=\; \sum_{t=1}^{T} \ell^t_{a^t} - \min_{k \in A} \sum_{t=1}^{T} \ell^t_k.$$

This measures how much worse the learner did than the single best fixed action in hindsight.

No-regret (Hannan consistency). An algorithm is **no-regret** (or **Hannan consistent**) if, against *any* sequence of loss vectors (even adversarially chosen),

$$\frac{R^T}{T} \;\to\; 0 \qquad \text{as } T \to \infty.$$

A stronger quantitative guarantee: the algorithm achieves **sublinear regret** if $R^T = o(T)$, and achieves $O(\sqrt{T})$ regret if $R^T \le C\sqrt{T \ln K}$ for some constant C.

Internal regret. A finer notion: the internal regret of switching action j to action k whenever j was played is

$$R^T_{j \to k} \;=\; \sum_{t : a^t = j} \left(\ell^t_j - \ell^t_k\right).$$

An algorithm has **no internal regret** if $\max_{j,k} R^T_{j\to k}/T \to 0$. No internal regret is strictly stronger than no external regret.

Coarse correlated equilibrium (CCE). A distribution $\mu \in \Delta(S)$ over joint strategy profiles is a CCE if for every player i and every fixed action $s'_i \in S_i$:

$$\mathbb{E}_{s\sim\mu}\big[u_i(s)\big] \;\geq\; \mathbb{E}_{s\sim\mu}\big[u_i(s'_i, s_{-i})\big].$$

No player can improve their expected payoff by unilaterally switching to a *fixed* alternative, where the expectation is over the correlation device.

Correlated equilibrium (CE). Stronger: μ is a CE if no player can improve by a *conditional* switch — after observing their own recommended action, no player wants to deviate. Every CE is a CCE; the converse is false in general.

Intuition

Three ways to understand what no-regret buys you.

The insurance policy. External regret asks: "Could I have done better by committing to a single action for all time?" If the answer is "not by much," you have no external regret. This is a *minimal* rationality requirement — you are not comparing yourself to the best adaptive strategy, or the best sequence of actions, only to the best fixed action. The requirement is deliberately weak because the guarantee must hold against *arbitrary* loss sequences, including adversarial ones. A stronger benchmark (e.g., competing with the best *sequence* of actions) is unachievable against adversaries.

The equilibrium factory. The most surprising consequence of no-regret is not about individual learning but about what happens when *everyone* uses it simultaneously. If all players in a finite game independently run no-regret algorithms, the empirical distribution of joint play converges to the set of coarse correlated equilibria. In two-player zero-sum games, every CCE yields the same payoff as Nash (because the CCE polytope collapses to the Nash set), so no-regret learning *computes Nash equilibria* in zero-sum games as a side effect. This is the structural link between regret and equilibrium.

The upgrade from fictitious play. Fictitious play best-responds to the empirical opponent distribution, which sounds strong but has no regret guarantee (Shapley's counterexample shows fictitious play can accumulate linear regret in non-zero-sum games). No-regret algorithms are weaker per-step — they do not necessarily best-respond — but they guarantee sublinear regret against *any* opponent, including adversarial ones. This robustness is what enables the equilibrium convergence theorem.

Structural Role

9.2 is the theoretical backbone of Module 9. It provides:

- **The regret framework** that defines the convergence criterion for all subsequent algorithms.
- **The CCE convergence theorem** that links individual regret bounds to collective equilibrium.
- **The conceptual foundation for 9.3 (CFR)**: CFR is no-regret learning applied independently to each information set in an extensive-form game.
- **The benchmark for 9.4 (MWU/Hedge)**: Hedge is a specific no-regret algorithm that achieves the optimal $O(\sqrt{T \ln K})$ regret bound.
- **The convergence criteria for 9.5**: the question "does a learning dynamic converge to Nash?" is answered by asking whether the associated regret goes to zero.

Without 9.2, the entire module is a collection of algorithms without a unifying performance criterion. With 9.2, every algorithm in Module 9 can be evaluated on the same axis: *how fast does its regret grow?*

Mathematical Development

The fundamental theorem: no-regret implies CCE convergence

Theorem. Let G be a finite n-player normal-form game. Suppose each player i uses a no-regret algorithm to select actions over rounds $t = 1, \ldots, T$. Let $\mu^T \in \Delta(S)$ denote the empirical distribution of joint play: $\mu^T(s) = \frac{1}{T}|\{t : s^t = s\}|$. Then every limit point of $\{\mu^T\}$ is a coarse correlated equilibrium.

Proof. Fix player i and a fixed action s_i'. The external regret of player i with respect to s_i' is

$$R_{i,s_i'}^T \;=\; \sum_{t=1}^{T} u_i(s_i', s_{-i}^t) - \sum_{t=1}^{T} u_i(s_i^t, s_{-i}^t).$$

(Using gains rather than losses; the sign flips.) No-regret guarantees $R_{i,s_i'}^T / T \to 0$ for every s_i'. Dividing by T:

$$\frac{1}{T}\sum_{t=1}^{T} u_i(s_i^t, s_{-i}^t) \;\geq\; \frac{1}{T}\sum_{t=1}^{T} u_i(s_i', s_{-i}^t) - \frac{R_{i,s_i'}^T}{T}.$$

As $T \to \infty$, the left side converges to $\mathbb{E}_\mu[u_i(s)]$ and the right to $\mathbb{E}_\mu[u_i(s_i', s_{-i})]$, with the regret term vanishing. So $\mathbb{E}_\mu[u_i(s)] \geq \mathbb{E}_\mu[u_i(s_i', s_{-i})]$, which is the CCE condition. $\square$

Corollary: Nash in zero-sum games

In a two-player zero-sum game, every CCE yields the minimax value to both players (because the CCE polytope is a singleton in payoff space for zero-sum games — any correlation device that prevents unilateral improvement must give each player at least their minimax value, and zero-sum forces the payoffs to sum to zero). Therefore, when both players use no-regret, the empirical distribution converges to a Nash equilibrium *in payoff*.

The exploitability of the empirical mixed strategies $\bar{\sigma}_1^T, \bar{\sigma}_2^T$ is bounded by:

$$\text{exploitability} \;\leq\; \frac{R_1^T + R_2^T}{T}.$$

With $O(\sqrt{T})$ regret for each player, exploitability drops as $O(T^{-1/2})$ — much faster than fictitious play.

No internal regret implies CE convergence

Theorem (Foster and Vohra 1997, Hart and Mas-Colell 2000). If all players use no-*internal*-regret algorithms, the empirical distribution converges to the set of correlated equilibria (not just coarse correlated equilibria). This is a strictly stronger guarantee.

Algorithms achieving no internal regret exist (e.g., Blum–Mansour calibration-based algorithms), but they are more complex than external-regret algorithms and typically have higher computational cost. For game-solving purposes, external regret (and hence CCE) is usually sufficient, especially in zero-sum games where CCE = Nash.

Regret matching (Hart and Mas-Colell 2000)

A specific no-regret algorithm that is the direct precursor to CFR:

1. Track cumulative regret for each action k: $R_k^t = \sum_{\tau=1}^{t}(u_i(k, s_{-i}^\tau) - u_i(s_i^\tau, s_{-i}^\tau))$.
2. Define the regret-positive part: $R_k^{t,+} = \max(R_k^t, 0)$.

3. At round $t+1$, play action k with probability proportional to $R_k^{t,+}$:

$$\sigma_i^{t+1}(k) \;=\; \begin{cases} \dfrac{R_k^{t,+}}{\sum_j R_j^{t,+}} & \text{if } \sum_j R_j^{t,+} > 0, \\ \dfrac{1}{K} & \text{otherwise.} \end{cases}$$

Theorem (Hart and Mas-Colell 2000). Regret matching achieves $R^T = O(\sqrt{T})$ external regret, and is therefore Hannan consistent.

Regret matching is the engine inside CFR (Node 9.3): at each information set, the player runs regret matching independently, and the global convergence follows from the per-info-set no-regret guarantee.

The Hannan set

Define the **Hannan set** as the set of payoff vectors achievable by strategies that are no-regret against any opponent. For two-player zero-sum games, the Hannan set equals the singleton $\{(V, -V)\}$ where V is the game value. For general games, the Hannan set contains the set of Nash equilibrium payoffs but may be larger. The CCE set is the equilibrium characterization of the Hannan set: the joint distributions reachable by no-regret play are exactly the CCEs.

Regret bounds and rates

Algorithm	External regret bound	Notes
Hedge / MWU (9.4)	$O(\sqrt{T \ln K})$	Optimal among oblivious algorithms
Regret matching	$O(\sqrt{T})$	Per-action tracking, precursor to CFR
Follow the perturbed leader	$O(\sqrt{T \ln K})$	Random perturbation to leader algorithm
EXP3 (bandit setting)	$O(\sqrt{TK \ln K})$	Only observes own payoff, not full vector

The $O(\sqrt{T \ln K})$ bound is optimal in the adversarial full-information setting (lower bound by Cesa-Bianchi et al.).

Worked Example

Two-player zero-sum: regret matching computes Nash

Consider Rock-Paper-Scissors with payoff matrix (for Row):

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}.$$

Both players run regret matching. Initialize all cumulative regrets to 0, so both play uniform $(1/3, 1/3, 1/3)$.

Round 1. Both play uniform. Expected payoff for Row from each action: 0 (by symmetry). Regret of each action: 0. Next-round strategy: still uniform.

By symmetry, regret matching stays at uniform forever in this game — the Nash equilibrium is reached in a single step. This is a degenerate example, but it illustrates the fixed-point property: if the current empirical play is Nash, regret matching stays there.

Now break the symmetry. Suppose Row plays R in round 1, Column plays S. Row's payoff: $u_1(R, S) = -1$. Counterfactual payoffs: $u_1(R, S) = -1$, $u_1(P, S) = -1$, $u_1(S, S) = 0$. Cumulative regret: $R_R^1 = 0$, $R_P^1 = 0$, $R_S^1 = 1$. (Wait — using the row-payoff against column's S: $u_1(R, S) = 1$, $u_1(P, S) = -1$, $u_1(S, S) = 0$.) Let me re-derive with the standard RPS matrix where Row = R beats Column = S:

Row played R, got payoff $u_1(R, S) = 1$. Regret of not playing P: $u_1(P, S) - 1 = -1 - 1 = -2$. Regret of not playing S: $u_1(S, S) - 1 = 0 - 1 = -1$. All regrets are negative, so $R_k^{1,+} = 0$ for all k. Row plays uniform next round. Column analogously updates. Over many rounds, the cumulative regrets oscillate around zero, and the empirical strategy converges to $(1/3, 1/3, 1/3)$.

Exploitability decay

After T rounds of regret matching by both players, the exploitability of the average strategies is bounded by $(R_1^T + R_2^T)/T \leq C/\sqrt{T}$. For RPS with 3 actions, after 10,000 rounds, exploitability is on the order of 0.01 — effectively Nash.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Action set A	Actions at a decision point: fold, call, raise (various sizes)
Loss vector ℓ^t	The EV of each action given the opponent's play this iteration
External regret	"How much better would I have done if I had always raised here instead of what I actually did?"
Regret matching	The per-info-set update rule inside every CFR solver
CCE convergence	The solver's average strategy profile converging to approximate Nash
Exploitability $O(T^{-1/2})$	The rate at which a CFR-based solver's output approaches GTO

Concrete situation: why solvers converge

When you run a solver (PioSolver, GTO+, MonkerSolver) on a postflop spot, the solver is running regret matching at every information set simultaneously. Each info set is an independent no-regret learner: it tracks cumulative regret for each available action (fold, call, bet 33%, bet 67%, etc.), and at each iteration it mixes in proportion to positive regrets. The fundamental theorem guarantees that the average strategy across all info sets converges to a CCE, which in the two-player zero-sum poker game is a Nash equilibrium. The exploitability drops as $O(T^{-1/2})$, so after enough iterations the strategy is approximately GTO.

This is the *entire* theoretical justification for why poker solvers produce Nash strategies. There is nothing else underneath — it is regret matching at every info set, plus the CCE-to-Nash collapse in zero-sum games.

Concrete situation: the meaning of "iterations" in a solver

When PioSolver reports "500 iterations, exploitability 0.3% pot," it is saying: regret matching has run for 500 rounds, and the sum of both players' maximum external regrets divided by iterations is 0.3% of pot.

The $O(T^{-1/2})$ rate means that cutting exploitability in half requires quadrupling the iterations. This is why getting from 1% to 0.1% exploitability takes 100x more compute than getting from 10% to 1%.

What this understanding buys you

Understanding regret is understanding why solvers work. If you know that the solver is running no-regret learning at every info set, you understand why convergence is guaranteed (the CCE theorem), why it is slow (the $\sqrt{T}$ rate), and why the output is Nash in zero-sum games (the CCE = Nash collapse). You also understand what the solver is *not* doing: it is not searching for a Nash equilibrium directly, it is not doing backward induction, and it is not reasoning about opponent rationality. It is just minimizing regret locally, and the Nash equilibrium emerges from the global interaction of local no-regret learners.

Regret matching is the link between 9.2 and 9.3. CFR (Node 9.3) is regret matching lifted to extensive-form games via the counterfactual regret decomposition. If you understand regret matching in the normal-form setting of 9.2, the only new ingredient in CFR is the trick that decomposes global regret into per-info-set counterfactual regrets.

Cross-Links

- **9.1 Fictitious play** — predecessor with no regret guarantee; motivates the move to no-regret.
- **9.3 CFR** — no-regret learning applied to extensive-form games via counterfactual regret.
- **9.4 Multiplicative weights / Hedge** — the canonical no-regret algorithm with optimal bounds.
- **9.5 Convergence to equilibrium** — CCE convergence is the main positive result.
- **2.1 Nash equilibrium** — the target in zero-sum games.
- **2.5 Correlated equilibrium** — the target under no-internal-regret.
- **Online learning / optimization** — the broader field in which regret minimization originates.

Structural Compression

Invariant: If every player in a finite game independently minimizes external regret (achieving sublinear $R^T = o(T)$), the empirical distribution of joint play converges to the set of coarse correlated equilibria; in two-player zero-sum games this set collapses to Nash, giving a $O(T^{-1/2})$ equilibrium computation method.

Mechanism: Each player tracks cumulative regret per action and mixes in proportion to positive regrets (regret matching) or exponential weights (Hedge). The CCE convergence theorem translates individual sublinear-regret guarantees into collective equilibrium convergence via a telescoping-sum argument on the regret definitions.

Cross-System Echo: The regret framework is the game-theoretic instantiation of **online convex optimization**: the learner picks a point in a convex set, nature reveals a loss function, and the learner's goal is sublinear regret against the best fixed point. In machine learning, this is the foundation of online gradient descent, AdaGrad, and the entire adaptive-learning-rate literature. In statistics, it connects to sequential prediction and the minimax theory of estimation. The structural invariant — *local sublinear regret implies global consistency* — is the same across all these domains.

Exercises

1. Prove that regret matching achieves $O(\sqrt{T})$ external regret. (Hint: use a potential function $\Phi^t = \sum_k (R_k^{t,+})^2$ and bound the per-round change.)
2. Show that every Nash equilibrium payoff vector is a CCE payoff vector, but construct a 2-player game where the set of CCE payoff vectors strictly contains the set of Nash payoff vectors.

3. Verify the CCE-to-Nash collapse in zero-sum games: prove that if μ is a CCE of a two-player zero-sum game, then the marginals μ_1, μ_2 form a Nash equilibrium.

4. An algorithm has regret bound $R^T \leq 3\sqrt{T \ln K}$. After 10,000 rounds in a zero-sum game with $K = 5$ actions per player, bound the exploitability of the average strategy profile.

5. Construct an adversarial loss sequence against which fictitious play accumulates linear regret (i.e., $R^T = \Omega(T)$), showing that fictitious play is *not* Hannan consistent in general.

6. Explain why no-*internal*-regret algorithms converge to correlated equilibria rather than coarse correlated equilibria. What is the structural difference between the two regret concepts that produces the tighter equilibrium set?

7. **Argue, structurally,** why the $O(\sqrt{T})$ regret bound is tight — no algorithm can guarantee $o(\sqrt{T})$ regret against adversarial loss sequences — and what this implies about the fundamental speed limit of equilibrium computation via self-play.

Counterfactual Regret Minimization (CFR)

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.3

Counterfactual regret minimization is the algorithm that solved poker. Not approximately, not heuristically — *solved*, in the game-theoretic sense of computing an epsilon-Nash equilibrium for games with 10^{13} to 10^{161} information sets. CFR, introduced by Zinkevich, Johanson, Bowling, and Piccione in 2007, is the structural bridge between the abstract no-regret learning framework of 9.2 and the concrete, astronomically large extensive-form games that define real poker. Its key insight is a decomposition theorem: the total regret of a strategy in an extensive-form game can be written as a sum of *counterfactual regrets* at individual information sets, and if each information set independently minimizes its own counterfactual regret (e.g., via regret matching), the global average strategy converges to a Nash equilibrium of the full game. This decomposition is the structural move that makes enormous games tractable — instead of one monstrous optimization, you get millions of tiny independent ones, each over a handful of actions at a single decision point.

Every major poker AI milestone — Cepheus (2015), DeepStack (2017), Libratus (2017), Pluribus (2019) — is built on CFR or one of its descendants. The algorithm is to poker what the simplex method is to linear programming or what backpropagation is to neural networks: the canonical computational engine that makes the theory useful. This node develops CFR from its formal foundations through its variants (CFR+, Monte Carlo CFR) to its landmark applications, and explains why it is *the* algorithm for imperfect-information games.

Formal Definition

Extensive-form game notation

Let Γ be a finite two-player zero-sum extensive-form game with imperfect information. Define:

- H: the set of all histories (sequences of actions from the root).
- $Z \subseteq H$: the set of terminal histories.
- $A(h)$: the set of actions available at non-terminal history h.
- $P(h) \in \{1, 2, c\}$: the player to act at h (or chance c).
- I_i: the set of information sets of player i. Each $I \in I_i$ is a set of histories that player i cannot distinguish.
- $A(I)$: the actions available at information set I (well-defined because all $h \in I$ have the same action set).
- $u_i(z)$: player i's payoff at terminal history z.

A **behavioral strategy** for player i is a function σ_i that assigns to each $I \in I_i$ a probability distribution over $A(I)$. A strategy profile $\sigma = (\sigma_1, \sigma_2)$ induces a probability distribution over terminal histories, with expected payoff $u_i(\sigma) = \sum_{z \in Z} \pi^\sigma(z)\, u_i(z)$, where $\pi^\sigma(z)$ is the probability of reaching z under σ.

Reach probabilities

For a history h, the **reach probability** under strategy profile σ decomposes as:

$$\pi^\sigma(h) \;=\; \pi_1^\sigma(h) \cdot \pi_2^\sigma(h) \cdot \pi_c(h),$$

where $\pi_i^\sigma(h)$ is the product of player i's action probabilities along the path to h, and $\pi_c(h)$ is the product of chance probabilities. Define the **counterfactual reach probability** for player i at h:

$$\pi_{-i}^\sigma(h) \;=\; \pi^\sigma(h) / \pi_i^\sigma(h),$$

the probability of reaching h given that player i plays to reach h (i.e., all factors except player i's own choices).

Counterfactual value

The **counterfactual value** of an information set $I \in I_i$ under strategy profile σ is:

$$v_i^\sigma(I) \;=\; \sum_{h \in I} \pi_{-i}^\sigma(h) \sum_{z \in Z(h)} \pi^\sigma(h, z)\, u_i(z),$$

where $Z(h)$ is the set of terminal histories extending h, and $\pi^\sigma(h, z)$ is the probability of going from h to z under σ. Intuitively: weight each history in the info set by how likely the *opponents and chance* made it, then compute expected payoff under the continuation strategy.

The counterfactual value of a specific action $a \in A(I)$ is:

$$v_i^\sigma(I, a) \;=\; \sum_{h \in I} \pi_{-i}^\sigma(h) \sum_{z \in Z(h,a)} \pi^\sigma(h \cdot a, z)\, u_i(z),$$

where $Z(h, a)$ is the set of terminal histories through h taking action a.

Counterfactual regret

The **instantaneous counterfactual regret** at information set I for action a at iteration t is:

$$r_i^t(I, a) \;=\; v_i^{\sigma^t}(I, a) - v_i^{\sigma^t}(I).$$

The **cumulative counterfactual regret** through iteration T is:

$$R_i^T(I, a) \;=\; \sum_{t=1}^{T} r_i^t(I, a).$$

The CFR algorithm

For each iteration $t = 1, 2, \ldots, T$:

1. **Traverse the game tree** under current strategy profile σ^t. Compute counterfactual values $v_i^{\sigma^t}(I)$ and $v_i^{\sigma^t}(I, a)$ for every info set I of player i and every action $a \in A(I)$.
2. **Update cumulative regrets:** $R_i^t(I, a) \leftarrow R_i^{t-1}(I, a) + r_i^t(I, a)$.
3. **Compute next strategy via regret matching:**

$$\sigma_i^{t+1}(I)(a) \;=\; \begin{cases} \dfrac{R_i^{t,+}(I, a)}{\sum_{a'} R_i^{t,+}(I, a')} & \text{if } \sum_{a'} R_i^{t,+}(I, a') > 0, \\[2ex] \dfrac{1}{|A(I)|} & \text{otherwise,} \end{cases}$$

where $R_i^{t,+}(I, a) = \max(R_i^t(I, a), 0)$.

4. **Accumulate the average strategy:**

$$\bar{\sigma}_i^T(I)(a) \;=\; \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I) \cdot \sigma_i^t(I)(a)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}.$$

Output: The average strategy profile $\bar{\sigma}^T = (\bar{\sigma}_1^T, \bar{\sigma}_2^T)$.

Intuition

CFR is best understood as an assembly of tiny independent no-regret learners, one per information set, coordinated by the game tree.

The decomposition trick. In a normal-form game, regret matching (9.2) works on a single player's action set. In an extensive-form game, the "action set" is the space of all behavioral strategies — exponentially large. CFR's core insight is that global regret *decomposes* into a sum of counterfactual regrets at individual info sets. If each info set independently achieves no-regret (via regret matching), the global strategy achieves no-regret, and by the fundamental theorem of 9.2, the average strategy converges to Nash in zero-sum games.

Counterfactual values as the right weighting. The "counterfactual" in CFR refers to the fact that each info set's regret is computed *as if the player had tried to reach that info set*. The opponents' and chance's contributions to reach probability are included, but the player's own reach probability is factored out. This is the correct weighting because it ensures that the decomposition of global regret into per-info-set regrets is exact — a non-trivial algebraic identity that Zinkevich et al. proved.

Why "average" strategy, not "current" strategy. At any given iteration, the current strategy σ^t can be wild — it swings from one extreme to another as regrets accumulate and sign-change. The *average* strategy, weighted by reach probabilities, smooths out these oscillations and is the one that converges to Nash. This is analogous to the distinction between last-iterate and average-iterate convergence in optimization: the average iterates of gradient descent converge even when the last iterates cycle.

Why it works for poker. A heads-up no-limit hold'em game has on the order of 10^{161} game states and 10^{14} information sets (after bucketing). No algorithm can enumerate these states explicitly. CFR's per-info-set decomposition means you only need to store regret values for each (info set, action) pair, not for the global strategy space. With abstraction (clustering similar hands and bet sizes), the number of info sets drops to 10^{12} or fewer, which fits in memory. Each iteration traverses the abstracted tree once, updating regrets. After billions of iterations, the average strategy is an epsilon-Nash equilibrium of the abstract game.

Structural Role

9.3 is the keystone node of the entire game theory course from a poker standpoint. It is where abstract theory meets the concrete reality of billion-dollar poker competition:

- **From 9.2:** CFR inherits the no-regret guarantee and the CCE convergence theorem. The only new ingredient is the counterfactual regret decomposition that lifts normal-form regret matching to extensive-form games.
- **From 3.2 (information sets):** CFR operates on info sets, not game nodes. The entire apparatus of imperfect information — what you know, what you do not, what your strategy must be consistent across — is the structural substrate.
- **From 3.4 (backward induction):** CFR traverses the game tree recursively, computing values bottom-up like backward induction. The difference is that CFR handles imperfect information and produces mixed strategies rather than pure ones.
- **From 4.3 (Bayesian games):** The counterfactual reach probability is a belief-like object — "how likely is the opponent to have put me in this info set?" — linking CFR to Bayesian reasoning.
- **To 9.5 (convergence):** CFR is the constructive proof that learning dynamics converge to Nash in extensive-form zero-sum games.
- **To 10.3 (strategic systems):** CFR-based solvers are the computational tools that make GTO poker strategies real, not merely theoretical.

Mathematical Development

The counterfactual regret decomposition theorem

Theorem (Zinkevich et al. 2007). For any player i in a finite extensive-form game, the overall external regret of player i through T iterations is bounded by the sum of positive counterfactual regrets at each information set:

$$R_i^T \;\le\; \sum_{I \in I_i} R_i^{T,+}(I),$$

where $R_i^{T,+}(I) = \max_{a \in A(I)} R_i^T(I,a)$ is the maximum positive counterfactual regret at I, and R_i^T is the overall external regret.

Consequence. If each info set I achieves sublinear counterfactual regret — i.e., $R_i^{T,+}(I) = O(\sqrt{T})$ — then the overall regret is $R_i^T \le |I_i| \cdot O(\sqrt{T}) = O(|I_i|\sqrt{T})$. The average strategy's exploitability is:

$$\epsilon^T \;\le\; \frac{R_1^T + R_2^T}{T} \;=\; O\!\left(\frac{|I_1| + |I_2|}{\sqrt{T}}\right).$$

To achieve exploitability ϵ, you need $T = O((|I_1| + |I_2|)^2/\epsilon^2)$ iterations. For a game with 10^{12} info sets and target exploitability 10^{-3} of the pot, this is astronomical — which is where Monte Carlo CFR and abstraction enter.

Proof sketch of the decomposition

The key identity: for any pair of strategies σ, σ' for player i, the difference in expected payoff can be written as:

$$u_i(\sigma_i', \sigma_{-i}) - u_i(\sigma_i, \sigma_{-i}) \;=\; \sum_{I \in I_i} \pi_{-i}^{\sigma}(I) \sum_{a \in A(I)} [\sigma_i'(I)(a) - \sigma_i(I)(a)]\, v_i^{\sigma}(I,a).$$

This follows from the factored form of reach probabilities and the law of total expectation over the game tree. The counterfactual regret of not playing action a at info set I is exactly the per-info-set contribution to the global regret of not switching to a everywhere at I. The decomposition is exact, not an approximation.

CFR+ (Tammelin 2014)

CFR+ is a practical improvement that accelerates convergence significantly:

1. **Regret flooring:** Instead of allowing cumulative regrets to go arbitrarily negative, CFR+ floors them at zero: $R_i^t(I,a) \leftarrow \max(R_i^{t-1}(I,a) + r_i^t(I,a),\ 0)$. This prevents an action that was bad early on from accumulating a large negative regret that takes many iterations to "pay off" before the action can be reconsidered.
2. **Linear averaging:** Instead of uniform averaging of past strategies, CFR+ weights iteration t by t in the average strategy:

$$\bar{\sigma}_i^T(I)(a) \;\propto\; \sum_{t=1}^{T} t \cdot \pi_i^{\sigma^t}(I) \cdot \sigma_i^t(I)(a).$$

This gives more weight to later iterations, which are closer to equilibrium, and speeds up convergence in practice.

Empirical performance. CFR+ converges roughly 10x faster than vanilla CFR in heads-up limit hold'em, and was the algorithm used to solve that game (Cepheus, Bowling et al. 2015). The theoretical convergence guarantee is the same — $O(T^{-1/2})$ exploitability — but the constants are dramatically better.

Monte Carlo CFR (MCCFR)

The bottleneck of vanilla CFR is that each iteration requires a full traversal of the game tree, computing counterfactual values at every info set. For large games (HUNL has $\sim 10^{161}$ game states), full traversal is impossible. Monte Carlo CFR replaces full traversal with *sampled* traversals, updating only the info sets encountered on a sampled path through the tree.

External sampling. Sample the opponent's actions and chance outcomes; traverse the tree fully for the updating player's actions. At each iteration, only the info sets along the sampled path are updated. This reduces per-iteration cost from $O(|H|)$ to $O(|H_i|)$ (the tree restricted to player i's decisions with opponent/chance actions sampled).

Outcome sampling. Sample *all* actions — both players and chance — to get a single trajectory through the tree. Update regrets only along that trajectory. Per-iteration cost is $O(d)$ where d is the depth of the tree, but variance is high.

Chance sampling. Sample only chance outcomes (e.g., which cards are dealt), then traverse the resulting perfect-information game fully. This is the most common variant for poker, because chance sampling eliminates the card-dealing combinatorics (which dominate the branching factor) while still doing exact computation within each deal.

Convergence. All MCCFR variants converge to Nash in the same $O(T^{-1/2})$ sense, but with higher variance. The effective convergence speed depends on the sampling scheme's variance-to-cost tradeoff. In practice, chance sampling with pruning is the standard for poker applications.

Pruning

Regret-based pruning (Brown and Sandholm 2015). If all actions at an info set have cumulative regret below $-C$ for some threshold, skip that info set entirely for the next several iterations — the regrets cannot become positive quickly, so the strategy will not change. This can reduce per-iteration cost by 10-100x in practice without affecting convergence.

Abstraction

Real poker games are too large for CFR even with Monte Carlo sampling. **Abstraction** reduces the game to a tractable size:

- **Card abstraction:** Cluster similar hands into "buckets" based on equity distribution, hand strength, or learned features. E.g., all flush draws on a monotone flop might be one bucket.
- **Action abstraction:** Restrict bet sizes to a finite grid (e.g., 33%, 67%, 100%, 150% of pot). The continuous bet-size space is discretized.

CFR is run on the abstract game; the resulting strategy is an approximate equilibrium of the abstract game. Whether it is a good approximation to the *real* game depends on the quality of the abstraction. Pathological abstractions can introduce significant error (the "abstraction pathology" — Waugh et al. 2009).

Subgame solving

Rather than solving the entire game once, decompose the game into subgames and solve each independently:

- **Unsafe subgame solving:** Fix the strategy in the rest of the tree and solve a subgame in isolation. This can produce exploitable strategies because the subgame solution may not be consistent with the trunk strategy.

- **Safe subgame solving (Burch, Johanson, Bowling 2014; Brown, Sandholm 2017):** Maintain a "gift" or "floor" value for each subgame entry point, ensuring that the subgame solution does not make the overall strategy worse. Libratus and Pluribus use safe subgame solving to resolve postflop situations in real time during play.

Depth-limited solving. Instead of solving all the way to terminal nodes, solve to a limited depth and use a learned value function to estimate continuation values at the leaves of the subtree. DeepStack uses this approach with neural-network value estimation.

Worked Example

CFR on Kuhn Poker

Kuhn Poker is the canonical toy poker game for demonstrating CFR. Three cards: Jack (J), Queen (Q), King (K). Two players. Each antes 1 chip. One card dealt to each player. Player 1 acts: check or bet 1. If check, Player 2 acts: check or bet 1. If Player 2 bets after check, Player 1 acts: fold or call. If Player 1 bets initially, Player 2 acts: fold or call. Higher card wins at showdown.

Information sets. Player 1 has 6 info sets: $\{J, Q, K\} \times \{\text{first action}, \text{facing bet after check}\}$. Player 2 has 6 info sets: $\{J, Q, K\} \times \{\text{facing check}, \text{facing bet}\}$.

CFR iteration 1. Initialize all strategies to uniform. Traverse the tree for all 6 possible card deals (J-Q, J-K, Q-J, Q-K, K-J, K-Q, each equally likely). For each deal, compute counterfactual values at each info set using the uniform strategy. Update cumulative regrets. Compute new strategy via regret matching.

After a few hundred iterations, the average strategy converges to the known Nash equilibrium of Kuhn Poker:

Player 1 (Nash equilibrium): - Holding J: check 100%. If facing bet: fold 2/3, call 1/3. - Holding Q: check 100%. If facing bet: call 1/3, fold 2/3. - Holding K: bet $3 \times \alpha$ for parameter $\alpha \in [0, 1/3]$; check otherwise. If facing bet after check: call 100%.

Player 2 (Nash equilibrium): - Holding J, facing check: bet 1/3. Facing bet: fold 100%. - Holding Q, facing check: check 100%. Facing bet: call 1/3, fold 2/3. - Holding K: always bet or call.

The game value is $-1/18$ for Player 1 (Player 2 has a slight positional advantage). CFR reaches exploitability below 10^{-4} within a few thousand iterations on this toy game.

Exploitability decay in Kuhn Poker

Iterations	Exploitability (chips)
10	0.15
100	0.04
1,000	0.012
10,000	0.003

The decay tracks the theoretical $O(T^{-1/2})$ rate. CFR+ on the same game converges roughly 5x faster.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Information set I	A specific decision point: your hand, the board, the action history so far
Action set $A(I)$	Fold, call, raise (various sizes) at that decision point
Counterfactual value $v_i^\sigma(I)$	The EV of reaching this spot, weighted by opponents' and chance's contribution to getting here
Counterfactual regret $r_i^t(I,a)$	How much better action a would have been than the current mixed strategy at this info set, this iteration
Cumulative regret $R_i^T(I,a)$	The running total of how much action a has been "missed" — positive means the action is underplayed
Regret matching	Mix over actions in proportion to how much you "regret" not playing them — the more you regret not raising, the more you raise next iteration
Average strategy $\bar{\sigma}^T$	The GTO output: the strategy the solver reports after T iterations
Exploitability	How much an opponent can win by best-responding to your strategy — the solver's error metric
Abstraction	Bucketing hands and discretizing bet sizes to make the game tree tractable
Subgame solving	"Solve this river spot in real time" — what Libratus and Pluribus do during play

Canonical poker applications

Cepheus (2015, Bowling et al., University of Alberta). Heads-up limit Texas hold'em (HULHE) was "essentially solved" using CFR+. The game has approximately 3.19×10^{14} information sets after isomorphic reduction and 3.16×10^{17} game states. Bowling et al. ran CFR+ for over 1,500 CPU-years of computation, producing a strategy with exploitability less than 1 milli-big-blind per hand — so small that a human playing 60 hands per hour for a 70-year lifetime would not be able to statistically detect the deviation from Nash play. Cepheus was the first *non-trivial* game to be "essentially weakly solved" in the sense of computational game theory: the strategy's exploitability is within the noise floor of statistical significance.

Libratus (2017, Brown and Sandholm, CMU). Libratus defeated four top professionals in a 120,000-hand heads-up no-limit Texas hold'em (HUNL) match ("Brains vs. Artificial Intelligence"), winning by a combined 1.77 million chips (14.7 bb/100 — a decisive margin). Libratus used three modules:

1. **Blueprint computation:** An abstracted HUNL game with $\sim 10^{12}$ info sets solved by Monte Carlo CFR to produce a coarse strategy for the entire game tree.
2. **Subgame solving:** When a specific postflop situation was reached, Libratus re-solved the relevant subgame in real time using safe subgame solving with a finer action abstraction. This is the move that made HUNL tractable: the blueprint handles the macro-structure, and real-time subgame solving provides precision at the micro-level.
3. **Self-improvement:** After each day of play, the blueprint was refined to close exploits the human opponents had found, using a targeted regret-update procedure.

Libratus's victory was the first time an AI beat top human players in HUNL, a game with $\sim 10^{161}$ game states.

DeepStack (2017, Moravcik et al.). DeepStack took a different approach: instead of computing a full blueprint, it used *depth-limited solving* with a deep neural network to estimate continuation values beyond the solving horizon. At each decision point, DeepStack re-solves the next few actions using CFR with neural-net leaf values. The neural nets were trained on millions of randomly generated poker situations, providing a fast approximate value function. DeepStack defeated professional poker players in HULHE matches, and its approach was the first to integrate deep learning with game-theoretic solving.

Pluribus (2019, Brown and Sandholm). Pluribus extended the Libratus approach to *six-player* no-limit hold'em — a much harder problem because (a) the game is no longer two-player zero-sum, (b) the number of possible opponent interactions grows combinatorially, and (c) Nash equilibrium is no longer the unique "safe" solution concept (CCE is the natural target). Pluribus used:

1. **Monte Carlo CFR for blueprint computation** in the six-player game, with action abstraction and card abstraction.
2. **Depth-limited search** during play, solving subgames to a limited depth with blueprint-based continuation values.
3. **A modified search that considers correlated opponent responses** — notably, Pluribus does *not* assume opponents play independently, and its real-time search considers opponent action profiles that are correlated in response to Pluribus's deviations.

Pluribus defeated 12 elite professionals (including several World Series of Poker bracelet winners) over 10,000 hands of 6-max no-limit hold'em, with a win rate statistically significant at $p < 0.001$. Its training cost was approximately $150 of cloud compute — roughly 8 days on a 64-core server — a striking demonstration that CFR-based methods do not require supercomputer-scale resources.

Why CFR is THE algorithm for poker

Five structural reasons:

1. **Handles imperfect information natively.** Unlike backward induction or minimax search, CFR operates on information sets, not game nodes. It does not need to know the opponent's cards — it works with the beliefs induced by the info-set structure.

2. **Decomposes over info sets.** The counterfactual regret decomposition means the algorithm scales with the number of *info sets*, not the number of game *states*. In HUNL, the number of game states is 10^{161}, but with abstraction, the number of info sets can be reduced to 10^{12} or fewer.

3. **Guaranteed convergence to Nash in zero-sum games.** The no-regret guarantee at each info set implies that the average strategy converges to a Nash equilibrium. This is not a heuristic — it is a theorem.

4. **Compatible with Monte Carlo sampling.** Full tree traversal is impossible for large games, but MCCFR variants allow sampled traversals that still converge, making CFR practical for games with 10^{14}+ info sets.

5. **Compatible with subgame solving and depth-limited search.** The blueprint-plus-refinement architecture used by Libratus and Pluribus is a natural extension of CFR: compute a coarse solution offline, then refine specific subgames online. This two-level structure is what makes real-time play feasible.

No other known algorithm family combines all five properties. Alpha-beta search (perfect-information games only), linear programming (does not scale beyond small games), fictitious play (no per-info-set decomposition, slow convergence), and reinforcement learning (no convergence guarantees in multi-agent settings) each fail on at least one of these axes.

Concrete situation: what your solver is doing

When you open PioSolver and input a flop spot — say, BTN vs BB single-raised pot on $A\heartsuit 7\diamondsuit 2\clubsuit$ — the solver is running CFR (or a variant) on the game tree rooted at that flop node. It initializes uniform strategies at every info set. Each iteration:

1. Traverses the tree from the flop through all turns and rivers (with chance sampling for card deals).
2. At each info set (your hand + board + action history), computes counterfactual values for each action.
3. Updates cumulative regrets.
4. Computes next-iteration strategy via regret matching.
5. Accumulates the average strategy.

After the configured number of iterations (typically 200-2000 for a subgame), the solver reports the average strategy and its exploitability. The output — "bet 67% pot with this range, check with that range" — is the average strategy $\bar{\sigma}^T$, and the exploitability number is $(R_1^T + R_2^T)/T$.

Concrete situation: real-time subgame re-solving at the table

Imagine you are playing a HUNL match and reach the turn on a board that was not in your pre-computed solution's abstraction grid. A Libratus-style approach would:

1. Take the current reach probabilities (your range and opponent's range at this turn node, implied by the blueprint strategy).
2. Construct a subgame from this turn node through all rivers to showdown, using a fine action abstraction.
3. Run CFR on this subgame for enough iterations to reach low exploitability.
4. Play according to the resulting strategy for this specific subgame.

This is what makes modern poker AI superhuman: it does not memorize a single fixed strategy. It recomputes, on the fly, the Nash equilibrium of each specific subgame it encounters, using CFR as the computational engine.

What this understanding buys you

Every solver output is a CFR average strategy. When you study a solver solution, you are looking at the output of a no-regret learning process that ran for many iterations. The strategy is not "the answer" — it is the average of thousands of iterations of a self-play process, and its quality is bounded by the exploitability metric. If the exploitability is 1% of pot, the strategy is within 1% of Nash. If it is 10%, you are still far from convergence.

Abstraction quality matters more than iteration count. The biggest source of error in solver output is not "not enough iterations" (which produces a known, bounded error) but "bad abstraction" (which can produce arbitrarily bad strategies). If you bucket hands poorly or discretize bet sizes too coarsely, CFR will converge to the Nash of the *wrong game* — perfectly optimal in the abstract game, but exploitable in the real game. This is why serious solvers spend significant effort on abstraction design.

CFR explains why GTO is hard to compute but easy to approximate. The exact Nash equilibrium of HUNL is intractable (the game is too large). But an epsilon-Nash with epsilon = 1 mbb/hand is "essentially solved." CFR is the algorithm that makes this possible: it does not find the exact Nash, but it finds a strategy that is so close to Nash that no opponent can exploit the gap in a human lifetime of play.

CFR explains the structure of solver outputs. Why do solver solutions often involve mixing between actions at specific frequencies? Because regret matching at each info set produces a mixed strategy that balances the regrets of each action. The frequencies are not arbitrary — they are the steady-state of a no-regret learning process. When a solver says "bet 67% pot 40% of the time, check 60% of the time," it is reporting the regret-matching equilibrium at that info set.

Cross-Links

- **9.1 Fictitious play** — the ancestor algorithm; CFR improves on it with per-info-set decomposition and no-regret guarantees.
- **9.2 No-regret learning** — the theoretical foundation; CFR is regret matching applied to extensive-form games.
- **9.4 Multiplicative weights** — an alternative no-regret algorithm; could replace regret matching at each info set in CFR (rarely done in practice).
- **9.5 Convergence to equilibrium** — CFR is the constructive proof that no-regret dynamics converge to Nash in extensive-form zero-sum games.
- **3.2 Information sets** — the structural substrate on which CFR operates.

- **3.4 Backward induction** — CFR's tree traversal is structurally analogous, but handles imperfect information.
- **4.3 Bayesian games** — counterfactual reach probabilities encode beliefs about opponent types.
- **2.1 Nash equilibrium** — the convergence target.
- **Online learning / optimization** — regret minimization in the extensive-form setting.

Structural Compression

Invariant: In a finite two-player zero-sum extensive-form game, the total external regret decomposes exactly into a sum of counterfactual regrets at individual information sets; independent no-regret learning (regret matching) at each info set therefore drives the average strategy profile to a Nash equilibrium, with exploitability decaying as $O(T^{-1/2})$.

Mechanism: Counterfactual reach probabilities factor out the acting player's contribution, enabling a per-info-set decomposition of global regret. Regret matching at each info set produces a mixed strategy proportional to positive cumulative regrets. The average strategy across iterations is the output. Monte Carlo sampling, pruning, and subgame solving extend the method to games too large for full traversal.

Cross-System Echo: CFR is the game-theoretic analogue of **coordinate descent in optimization**: instead of optimizing a single massive objective, decompose it into independent per-coordinate subproblems and cycle through them. In machine learning, stochastic gradient descent decomposes the empirical risk over data points; in CFR, the decomposition is over information sets. The structural insight is the same — *global convergence from local updates* — and the mathematical engine is the same — *each local update guarantees bounded local regret, and the global bound is the sum.* The decomposition theorem (Zinkevich 2007) is the game-theoretic counterpart of the separability lemma in coordinate descent theory.

Exercises

1. Implement vanilla CFR for Kuhn Poker. Run 10,000 iterations and verify that the average strategy's exploitability is below 0.005 chips. Compare the output to the known analytic Nash equilibrium.
2. Prove the counterfactual regret decomposition theorem: show that $R_i^T \leq \sum_{I \in I_i} R_i^{T,+}(I)$ for any extensive-form game. (Hint: expand the definition of global external regret using the reach-probability factorization.)
3. Derive the exploitability bound $\epsilon^T \leq (|I_1| + |I_2|) \cdot C/\sqrt{T}$ from the per-info-set $O(\sqrt{T})$ regret bound. What does this imply about the number of iterations needed to solve a game with 10^{12} info sets to exploitability 10^{-3}?
4. Explain why CFR+ (regret flooring + linear averaging) converges faster than vanilla CFR in practice. Construct a simple example (e.g., a modified Kuhn Poker) where an action accumulates large negative regret under vanilla CFR that delays convergence, and show that CFR+ avoids this.
5. Compare external sampling and outcome sampling in MCCFR. For a game tree of depth d with branching factor b, what is the per-iteration cost of each? Which has higher variance? Why is chance sampling the standard in poker?
6. Explain the abstraction pathology: construct a simple extensive-form game where applying a hand abstraction (merging two info sets into one) causes CFR to converge to a strategy that is more exploitable than the original game's Nash equilibrium. What structural property of the abstraction causes this failure?
7. **Argue, structurally,** why CFR's per-info-set decomposition is the essential algorithmic innovation for imperfect-information games, and why no algorithm based on full-game optimization (e.g., linear

programming for the sequence form) can scale to games of poker's size. What is the computational complexity barrier that decomposition overcomes?

Multiplicative Weights and the Hedge Algorithm

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.4

The Multiplicative Weights Update (MWU) method is the workhorse no-regret algorithm: simple, optimal, and ubiquitous. Where regret matching (9.2) sets action probabilities proportional to positive cumulative regrets, MWU sets them proportional to *exponential* weights on cumulative gains — a "soft-max" rather than a "hard-max." This yields the optimal $O(\sqrt{T \ln K})$ external regret bound, the theoretical gold standard. The Hedge algorithm (Freund and Schapire 1997) is the decision-theoretic formulation of MWU: a learner choosing among K "experts" updates weights multiplicatively based on their performance. The same algorithm appears under different names in a dozen fields — Weighted Majority (Littlestone and Warmuth 1994), exponentiated gradient descent (Kivinen and Warmuth 1997), entropic mirror descent, the matrix multiplicative weights method — and each instantiation reveals a different facet of the same structural principle: *downweight losers exponentially, upweight winners exponentially, and the resulting mixture tracks the best fixed expert with sublinear regret.* In the game-theoretic context, MWU is the standard algorithm used to compute Nash equilibria of zero-sum games via the CCE convergence theorem of 9.2.

Formal Definition

Setting. A learner chooses among K actions (or "experts") over T rounds. At round t, the learner picks a distribution $w^t \in \Delta_K$ over actions. Nature (or an adversary) reveals a loss vector $\ell^t \in [0,1]^K$. The learner incurs expected loss $\langle w^t, \ell^t \rangle = \sum_k w_k^t \ell_k^t$.

Multiplicative Weights Update (MWU). Fix a learning rate $\eta > 0$. Initialize $w_k^1 = 1/K$ for all k. Update:

$$\tilde{w}_k^{t+1} \;=\; w_k^t \cdot e^{-\eta \ell_k^t}, \qquad w_k^{t+1} \;=\; \frac{\tilde{w}_k^{t+1}}{\sum_j \tilde{w}_j^{t+1}}.$$

Equivalently, the log-weight of action k decreases linearly in the cumulative loss: $\ln \tilde{w}_k^{t+1} = \ln w_k^1 - \eta \sum_{\tau=1}^{t} \ell_k^\tau$. The normalization ensures the weights form a valid probability distribution.

Hedge algorithm (Freund and Schapire 1997). Identical to MWU with losses in $[-1, 1]$ (or equivalently, gains). The name "Hedge" comes from the financial hedging interpretation: the learner "hedges" their bets across experts.

Weighted Majority (Littlestone and Warmuth 1994). The discrete predecessor: each expert makes a binary prediction, and wrong experts have their weight multiplied by $\beta \in (0, 1)$. The continuous generalization is MWU with $\beta = e^{-\eta}$.

Intuition

Exponential punishment. An action that incurs high loss in a round has its weight multiplied by $e^{-\eta \ell_k^t} < 1$, shrinking it. An action with low loss keeps its weight or grows relative to losers. Over time, the weight distribution concentrates on actions that have performed well historically. The exponential form is key: it penalizes consistently bad actions much faster than linear reweighting would. An action that incurs loss 1 every round has its weight shrink by $e^{-\eta}$ per round — exponential decay, which is why consistently bad actions are pruned quickly.

The tuning knob η. Large η means aggressive learning: bad actions are killed quickly, but the algorithm over-commits and has high regret if the environment shifts. Small η means conservative learning: slow

to abandon bad actions, but robust to shifts. The optimal choice $\eta = \sqrt{\ln K/T}$ balances exploration and exploitation, yielding the $O(\sqrt{T \ln K})$ bound.

Entropic regularization. MWU can be derived as "follow the leader" regularized by the negative entropy $\sum_k w_k \ln w_k$. Without regularization, "follow the leader" (play the action with best cumulative performance) is unstable — it can switch abruptly and incur high regret. Adding the entropy penalty keeps the weight distribution from collapsing to a single action, providing the smoothness that enables low regret. This is the "mirror descent" interpretation: MWU is gradient descent in the space of probability distributions, with the KL divergence as the Bregman distance.

Structural Role

9.4 supplies the canonical algorithm for the no-regret framework of 9.2:

- **9.2 establishes the criterion** (sublinear regret ⇒ CCE convergence). 9.4 exhibits the *optimal* algorithm achieving that criterion.
- **9.1 fictitious play** is a hard-max rule (best-respond to empirical distribution). MWU is a soft-max rule (weight by exponential of cumulative gains). The softness is exactly what enables no-regret guarantees.
- **9.3 CFR** uses regret matching, not MWU, at each info set — but MWU could substitute with the same convergence guarantee. The choice of regret matching in CFR is historical and practical (simpler to implement, no learning-rate tuning), not fundamental.
- **9.5 convergence to equilibrium** uses MWU's regret bound as the engine for constructive equilibrium proofs.
- **9.6 reinforcement learning** connects to policy-gradient methods that use entropy regularization, which is the continuous-action analogue of MWU.

Beyond game theory, MWU appears in:

- **Machine learning:** Boosting (AdaBoost is MWU applied to training examples).
- **Online optimization:** Online mirror descent with entropy regularizer.
- **Combinatorial optimization:** The Plotkin–Shmoys–Tardos framework for approximately solving LPs and SDPs using MWU as a separation oracle.
- **Quantum computing:** The matrix MWU method for semidefinite programming (Arora and Kale 2016).

Mathematical Development

The regret bound

Theorem (Freund and Schapire 1997). For MWU with learning rate $\eta \in (0, 1/2]$ and losses $\ell_k^t \in [0, 1]$, the cumulative regret against any fixed action k^* is:

$$\sum_{t=1}^{T} \langle w^t, \ell^t \rangle - \sum_{t=1}^{T} \ell_{k^*}^t \;\leq\; \frac{\ln K}{\eta} + \eta T.$$

Proof. Define the potential $\Phi^t = \ln(\sum_k \tilde{w}_k^t)$. We have:

$$\Phi^{t+1} - \Phi^t \;=\; \ln \frac{\sum_k w_k^t e^{-\eta \ell_k^t}}{\sum_k w_k^t}.$$

Using the inequality $e^{-x} \leq 1 - x + x^2$ for $x \in [0, 1]$ (valid when $\eta \leq 1/2$):

$$\sum_k w_k^t e^{-\eta \ell_k^t} \;\le\; \sum_k w_k^t (1 - \eta \ell_k^t + \eta^2 (\ell_k^t)^2) \;=\; 1 - \eta \langle w^t, \ell^t \rangle + \eta^2 \langle w^t, (\ell^t)^2 \rangle.$$

Taking logarithms and using $\ln(1-x) \le -x$:

$$\Phi^{t+1} - \Phi^t \;\le\; -\eta \langle w^t, \ell^t \rangle + \eta^2.$$

Summing over $t = 1, \ldots, T$:

$$\Phi^{T+1} - \Phi^1 \;\le\; -\eta \sum_t \langle w^t, \ell^t \rangle + \eta^2 T.$$

Now, $\Phi^1 = \ln K$ (uniform initialization), and $\Phi^{T+1} \ge \ln \tilde{w}_{k^*}^{T+1} = \ln(1/K) - \eta \sum_t \ell_{k^*}^t$ (lower-bounding the sum by any single term). Combining:

$$-\eta \sum_t \ell_{k^*}^t - \ln K \;\le\; -\eta \sum_t \langle w^t, \ell^t \rangle + \eta^2 T + \ln K.$$

Rearranging and dividing by η:

$$\sum_t \langle w^t, \ell^t \rangle - \sum_t \ell_{k^*}^t \;\le\; \frac{2 \ln K}{\eta} + \eta T. \qquad \square$$

(The tighter bound $\ln K/\eta + \eta T$ follows from a sharper use of the exponential inequality.)

Optimal η. Setting $\eta = \sqrt{\ln K / T}$:

$$R^T \;\le\; 2\sqrt{T \ln K}.$$

This is $O(\sqrt{T \ln K})$ and matches the known lower bound up to constants.

MWU for zero-sum game solving

Given a two-player zero-sum game with $m \times n$ payoff matrix A, both players run MWU:

- Row player maintains weights over rows, updated by column player's mixed strategy.
- Column player maintains weights over columns, updated by row player's mixed strategy.

After T rounds, the average mixed strategies $\bar{w}_1^T, \bar{w}_2^T$ satisfy:

$$\text{exploitability} \;\le\; \frac{R_1^T + R_2^T}{T} \;\le\; \frac{4\sqrt{\ln(\max(m,n))}}{\sqrt{T}}.$$

This gives an $O(\ln(mn)/\epsilon^2)$ algorithm for computing ϵ-Nash equilibria of zero-sum games — polynomial in the matrix dimensions and $1/\epsilon$. Compare to fictitious play's rate of $O(1/\epsilon^{m+n-2})$, which is exponentially worse.

Connection to mirror descent

MWU is an instance of **mirror descent** on the probability simplex with the *negative entropy* as the mirror map:

$$\psi(w) \;=\; \sum_k w_k \ln w_k.$$

The Bregman divergence associated with this mirror map is the KL divergence:

$$D_\psi(w\|v) \;=\; \sum_k w_k \ln \frac{w_k}{v_k}.$$

Mirror descent updates: $w^{t+1} = \arg\min_{w\in\Delta_K} \{\eta\langle w, \ell^t\rangle + D_\psi(w\|w^t)\}$. The solution is exactly the MWU update. This derivation shows that MWU is the "natural" gradient descent on the simplex, where "natural" means using the geometry induced by the entropy function rather than the Euclidean distance.

Exponentiated gradient descent

For the online linear optimization variant where the learner picks $w^t \in \Delta_K$ and observes a linear loss $\langle w^t, \ell^t\rangle$, MWU is equivalent to **exponentiated gradient descent** (Kivinen and Warmuth 1997):

$$w_k^{t+1} \;\propto\; w_k^t \cdot \exp\!\left(-\eta\,\frac{\partial}{\partial w_k}\langle w^t, \ell^t\rangle\right) \;=\; w_k^t \cdot e^{-\eta\ell_k^t}.$$

This is the same update as MWU, confirming that MWU is gradient descent in the multiplicative (log) parameterization.

The Weighted Majority algorithm

The discrete version (Littlestone and Warmuth 1994): each expert k makes a binary prediction. The learner follows the weighted majority vote. After each round, multiply the weight of each expert who was wrong by $\beta \in (0,1)$. After T rounds, the number of mistakes is at most:

$$M \;\le\; \frac{\ln K + M^* \ln(1/\beta)}{1-\beta},$$

where M^* is the number of mistakes of the best expert. Setting $\beta = 1/2$: $M \le 2.4(M^* + \ln K)$. This was the precursor to Hedge and MWU.

Worked Example

MWU on a 3x3 zero-sum game

Row player has 3 actions: $\{R, P, S\}$ (Rock, Paper, Scissors). Payoff matrix (Row's gain):

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}.$$

Both run MWU with $\eta = 0.1$. Initialize $w^1 = (1/3, 1/3, 1/3)$ for both.

Round 1. Both play $(1/3, 1/3, 1/3)$. Expected loss for Row against Column's uniform: each action has expected payoff 0 (by symmetry). Loss vector: $\ell^1 = (0, 0, 0)$ (normalized from the payoff). Weights unchanged. Both remain uniform.

This symmetric game is a fixed point for MWU — the Nash equilibrium is reached immediately because the game's symmetry makes every action equally good against uniform play.

Breaking symmetry. Suppose Column deviates to $(0.5, 0.3, 0.2)$ in round 1. Row's loss vector (negated payoff, since we minimize loss): $\ell^1_R = -(0{\cdot}0.5+(-1){\cdot}0.3+1{\cdot}0.2) = 0.1$, $\ell^1_P = -(1{\cdot}0.5+0{\cdot}0.3+(-1){\cdot}0.2) = -0.3$, $\ell^1_S = -((-1)\cdot 0.5 + 1\cdot 0.3 + 0\cdot 0.2) = 0.2$.

Updated weights: $\tilde{w}^2_R = (1/3)e^{-0.1\cdot 0.1} = 0.3300$, $\tilde{w}^2_P = (1/3)e^{0.1\cdot 0.3} = 0.3434$, $\tilde{w}^2_S = (1/3)e^{-0.1\cdot 0.2} = 0.3267$. After normalization: $w^2 \approx (0.330, 0.343, 0.327)$. Row shifts toward Paper (which exploits Column's overweighting of Rock). Over subsequent rounds, both players adjust, and the average strategies converge to the Nash $(1/3, 1/3, 1/3)$ from any initial condition.

Convergence rate comparison

For the 3x3 RPS game ($K = 3$), after T rounds:

T	MWU exploitability bound	Fictitious play exploitability bound
100	$2\sqrt{\ln 3/100} \approx 0.21$	$O(100^{-1/4}) \approx 0.32$
10,000	$2\sqrt{\ln 3/10000} \approx 0.021$	$O(10000^{-1/4}) \approx 0.10$

MWU is faster by a substantial margin, and the gap widens with game size.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Expert k	An action at a decision point (fold, call, raise 50%, raise 100%, etc.)
Weight w^t_k	The probability assigned to action k at iteration t of a solver
Loss ℓ^t_k	The negative counterfactual value of action k at this info set, this iteration
Learning rate η	Controls how aggressively the solver shifts between actions across iterations
Average strategy $\bar{w}^T$	The solver's GTO output at this info set

Concrete situation: MWU vs regret matching in solvers

Most poker solvers use regret matching rather than MWU at each info set. The reason is practical: regret matching requires no learning-rate parameter (it is parameter-free), while MWU requires choosing η, which depends on the time horizon T — a quantity not always known in advance. Adaptive learning-rate variants of MWU (e.g., AdaHedge) exist but add complexity. Regret matching achieves slightly worse theoretical bounds ($O(\sqrt{T})$ vs $O(\sqrt{T \ln K})$), but in practice the difference is negligible because the number of actions at each info set is small (typically 2-5).

However, MWU's theoretical optimality matters for understanding *why* the $O(T^{-1/2})$ convergence rate is fundamental. No algorithm — not regret matching, not MWU, not any future invention — can beat $\Omega(\sqrt{T})$

regret against adversarial losses, so the $O(T^{-1/2})$ exploitability rate is a hard lower bound on how fast any self-play method can converge to Nash.

What this understanding buys you

The learning rate controls the speed-stability tradeoff. In solver tuning, if a solver offers MWU-based updates, a higher η converges faster initially but may oscillate; a lower η is more stable but slower. Understanding MWU tells you this is not a bug — it is a fundamental tradeoff, and the optimal η depends on how many iterations you plan to run.

MWU explains why solver outputs are smooth. The exponential weighting ensures that no action is ever assigned probability exactly 0 until it has been consistently dominated for many iterations. This smoothness is why solver strategies involve many mixed frequencies rather than sharp pure-strategy prescriptions — the no-regret machinery produces genuinely mixed strategies as output, not approximations to pure ones.

Cross-Links

- **9.1 Fictitious play** — hard-max predecessor; MWU is the soft-max improvement.
- **9.2 No-regret learning** — MWU achieves the optimal bound within this framework.
- **9.3 CFR** — uses regret matching instead of MWU; the same convergence theory applies.
- **9.5 Convergence to equilibrium** — MWU's rate is the engine for quantitative convergence results.
- **Online convex optimization** — MWU is mirror descent with entropy regularizer.
- **Machine learning: boosting** — AdaBoost is MWU over training examples.
- **Optimization: LP/SDP solving** — MWU as separation oracle (Plotkin-Shmoys-Tardos, Arora-Kale).

Structural Compression

Invariant: Multiplicative (exponential) reweighting of actions based on cumulative losses achieves $O(\sqrt{T \ln K})$ external regret — optimal among all online learning algorithms in the adversarial full-information setting — and when used by all players in a zero-sum game, computes an ϵ-Nash equilibrium in $O(\ln K/\epsilon^2)$ iterations.

Mechanism: Each action's weight is multiplied by $e^{-\eta \cdot \text{loss}}$ per round and renormalized. The potential function $\Phi^t = \ln \sum_k \tilde{w}_k^t$ telescopes to give the regret bound. The learning rate $\eta = \sqrt{\ln K/T}$ optimally balances the two terms.

Cross-System Echo: MWU is the discrete analogue of **gradient flow on the probability simplex with KL-divergence geometry** (the "information geometry" of Amari). In statistical mechanics, the Gibbs distribution $p_k \propto e^{-\beta E_k}$ assigns probabilities proportional to $e^{-\text{energy}}$, which is exactly the MWU steady state when losses are constant. In biology, the replicator dynamic — the continuous-time version of evolutionary selection — is the continuous-time limit of MWU (Taylor and Jonker 1978, Losert and Akin 1983). The exponential reweighting principle — penalize bad states exponentially, favor good states exponentially — is one of the most universal computational principles in nature and mathematics.

Exercises

1. Prove the MWU regret bound $R^T \leq \ln K/\eta + \eta T$ from the potential function argument. Verify the optimal $\eta = \sqrt{\ln K/T}$ yields $R^T \leq 2\sqrt{T \ln K}$.
2. Implement MWU for a 3x3 zero-sum game (both players). Run for 10,000 iterations and plot the exploitability as a function of T. Verify the $O(T^{-1/2})$ rate empirically.
3. Show that the Weighted Majority algorithm is a special case of MWU with binary losses. Derive the mistake bound $M \leq 2.4(M^* + \ln K)$ from the general regret bound.

4. Derive the MWU update as the solution to the mirror-descent optimization problem: $w^{t+1} = \arg\min_{w \in \Delta_K} \{\eta \langle w, \ell^t \rangle + D_{KL}(w \| w^t)\}$. (Hint: use Lagrange multipliers on the simplex constraint.)

5. Compare MWU and regret matching on a 2-action game with adversarial losses that oscillate between $(0, 1)$ and $(1, 0)$. Which converges faster? Which has smoother weight trajectories? Why?

6. Show that AdaBoost can be derived as MWU applied to a game between a learner (choosing among weak classifiers) and an adversary (choosing training examples). What is the payoff matrix? What is the equilibrium interpretation of the boosted classifier?

7. **Argue, structurally,** why exponential reweighting (MWU) achieves better regret than linear reweighting (fictitious play) in general, but both converge in zero-sum games. What structural property of zero-sum games makes linear reweighting sufficient, and what do general games demand that only exponential reweighting provides?

Convergence to Equilibrium

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.5

This node asks the central question of learning in games: **when do learning dynamics converge to equilibrium, and to which equilibrium?** Nodes 9.1-9.4 developed four learning algorithms – fictitious play, no-regret learning, CFR, and multiplicative weights – each with its own convergence guarantees. Node 9.5 unifies these results, identifies the structural conditions under which convergence occurs, and maps out the landscape of equilibrium concepts that different learning dynamics target. The key insight, and the thesis's contribution to this node, is that **convergence is a property of the dynamics, not of the equilibrium concept**: the same equilibrium set can be approached by dynamics that converge, cycle, or diverge, and the choice of learning dynamics is a design decision analogous to the choice of optimizer in VQA. The separation of "what we are converging to" (the equilibrium concept) from "how we get there" (the learning dynamics) is the exact analogue of the thesis's separation of objective from dynamics in VQA optimization.

Formal Definition

Equilibrium concepts (hierarchy). Consider a finite n-player game $G = (N, \{S_i\}, \{u_i\})$ and a sequence of mixed strategies $(\sigma^1, \sigma^2, \ldots)$ generated by a learning algorithm.

- **Nash equilibrium (NE).** A profile σ^* where no player can improve by unilateral deviation. The strictest standard concept.
- **Correlated equilibrium (CE).** A distribution $\mu \in \Delta(S)$ such that each player's recommended action is optimal given the recommendation: $\mathbb{E}_{s_{-i}|s_i}[u_i(s_i, s_{-i})] \geq \mathbb{E}_{s_{-i}|s_i}[u_i(s_i', s_{-i})]$ for all s_i'. Strictly weaker than NE: NE $\subset$ CE.
- **Coarse correlated equilibrium (CCE).** A distribution μ such that no player can improve by committing to a fixed action *before* seeing their recommendation: $\mathbb{E}_{s\sim\mu}[u_i(s_i, s_{-i})] \geq \mathbb{E}_{s\sim\mu}[u_i(s_i', s_{-i})]$. Strictly weaker than CE: CE $\subset$ CCE.

The hierarchy is NE $\subset$ CE $\subset$ CCE, with strict inclusions in general.

Convergence notions.

- **Time-average convergence.** The empirical distribution of play $\bar{\sigma}^T = \frac{1}{T}\sum_{t=1}^{T} \sigma^t$ converges to an equilibrium set (e.g., to a CE or CCE).
- **Last-iterate convergence.** The actual strategy σ^T (not the average) converges to an equilibrium.
- **Regret convergence.** The total regret R^T grows sublinearly ($R^T = o(T)$), which implies time-average convergence to CCE. Internal regret $R^T_{\text{int}} = o(T)$ implies convergence to CE.

Main convergence results.

Learning algorithm	Convergence guarantee	Target equilibrium
Fictitious play (9.1)	Converges in 2-player zero-sum, 2x2, potential games; may cycle in general	NE (when it converges)
No-regret / MWU (9.2, 9.4)	Time-average converges always; last-iterate may cycle	CCE (external regret), CE (internal regret)
CFR (9.3)	Time-average converges in 2-player zero-sum extensive-form games	NE of the extensive-form game
Optimistic MWU / gradient descent-ascent	Last-iterate converges in 2-player zero-sum	NE

Intuition

The landscape of convergence results is shaped by a fundamental tension: **learning dynamics that are powerful enough to guarantee convergence in general games can only guarantee convergence to weak equilibrium concepts (CCE), while dynamics that converge to strong concepts (NE) only work in restricted game classes.**

This is not a failure of specific algorithms – it is a structural limitation. Hart and Mas-Colell (2003) showed that any "uncoupled" learning dynamics (where each player's strategy update depends only on their own payoffs, not on others' payoff functions) cannot converge to Nash equilibrium in all games. The impossibility is fundamental: Nash equilibrium requires mutual best-response consistency, but uncoupled learning cannot coordinate on this consistency without access to the other players' payoff structure.

The practical consequence is a design tradeoff:

1. **If you want guaranteed convergence in all games**, use no-regret learning (MWU, Hedge). You get CCE convergence. This is enough for many applications but does not pin down a unique outcome.
2. **If you want Nash convergence**, restrict to a well-behaved game class. Two-player zero-sum games are the most important class: CFR and optimistic gradient descent-ascent both converge to NE in last iterate (or time-average) for these games. Potential games and supermodular games also have strong convergence results.
3. **If you want Nash convergence in general games**, you need coupled dynamics (where each player knows or can infer others' payoffs). This is the mechanism-design route: design the game so that the learning dynamics produce the desired outcome.

The thesis's contribution is to frame this tradeoff as a **separation of objective from dynamics**. The equilibrium concept (NE, CE, CCE) is the objective – the target set the dynamics are trying to reach. The learning algorithm is the dynamics – the update rule that generates the trajectory. The convergence theorem specifies which (dynamics, objective) pairs are compatible. This is exactly the VQA framework: the cost function is the objective, the optimizer is the dynamics, and the convergence analysis asks which (optimizer, cost function) pairs yield convergence.

The deeper parallel: in VQA, the thesis argues that the right approach is to design the optimizer (dynamics) to match the structure of the cost landscape (objective). In learning-in-games, the same principle holds: the right approach is to choose the learning algorithm (dynamics) to match the structure of the game (objective). CFR is matched to extensive-form zero-sum games; MWU is matched to general games with external-regret targets; fictitious play is matched to zero-sum and potential games. Mismatch leads to cycling, divergence, or convergence to the wrong concept.

Structural Role

9.5 is the synthesis node of Module 9. It collects the convergence results of 9.1-9.4 into a unified framework and maps out the landscape of what is possible and what is not. It also serves as the bridge to Module 10:

- **9.1-9.4** provide the algorithms and their individual convergence results.
- **9.5** provides the unified view: which algorithms converge to which concepts in which game classes.
- **9.6 (RL in games)** extends to the setting where players do not even know their own payoff functions and must learn from realized rewards.
- **10.5-10.6** use the convergence landscape to discuss institutional design and meta-games: if you want a particular equilibrium, you need to choose (or design) the game and the learning dynamics together.

The thesis voice is strongest here. The framing of convergence as "separation of objective from dynamics" is a direct import from VQA Modules 6-7. The claim is that this framing is not just an analogy – it is a structural isomorphism. The same mathematical objects appear in both settings: gradient-like dynamics on strategy/parameter spaces, state-dependent objectives (payoffs depend on all players' strategies, cost

landscapes depend on the quantum state), and convergence conditions that depend on the interaction between the dynamics and the objective's curvature.

Mathematical Development

No-regret implies CCE convergence

Theorem. If all players use no-regret learning (external regret $R_i^T = o(T)$ for each i), then the empirical distribution of play $\bar{\mu}^T = \frac{1}{T}\sum_{t=1}^{T} \delta_{s^t}$ converges to the set of CCE as $T \to \infty$.

Proof sketch. The CCE condition for distribution μ is: for every i and every alternative action s_i':

$$\mathbb{E}_{s\sim\mu}[u_i(s_i, s_{-i})] \geq \mathbb{E}_{s\sim\mu}[u_i(s_i', s_{-i})].$$

This is equivalent to: no player's external regret is positive. If each player's regret is $o(T)$, then the time-average $\bar{\mu}^T$ approximately satisfies the CCE condition with error $O(R^T/T) \to 0$. The empirical distribution converges to the CCE set.

Internal regret implies CE convergence

Theorem. If all players use no-internal-regret learning (internal regret $R_i^{T,\text{int}} = o(T)$ for each i), then $\bar{\mu}^T$ converges to the set of CE.

Internal regret measures: the maximum improvement from swapping every instance of action a to action b in the history, for all pairs (a, b). CE requires that conditional on each recommended action, no switch is profitable – which is precisely the no-internal-regret condition.

Hart-Mas-Colell impossibility

Theorem (Hart-Mas-Colell 2003). There is no uncoupled learning dynamics that converges to Nash equilibrium in all finite games.

Uncoupled means: player i's update depends only on the history of play and their own payoff function u_i, not on others' payoff functions. This is the natural restriction for decentralized learning.

The proof constructs a three-player game where any uncoupled dynamics must cycle or diverge. The structural reason: Nash equilibrium requires mutual consistency of best responses, which is a global property of the game. Uncoupled dynamics can only detect local information (own payoffs), which is insufficient to verify global consistency.

Convergence in two-player zero-sum games

Two-player zero-sum games are the exception to the impossibility. The minimax theorem (von Neumann 1928) ensures that the Nash equilibrium is equivalent to both players minimizing regret, and any pair of no-regret learners converges to NE in time-average.

Stronger results:

- **CFR (9.3)** converges to NE in time-average in extensive-form zero-sum games, with rate $O(1/\sqrt{T})$.
- **Optimistic MWU (Rakhlin-Sridharan 2013, Daskalakis et al. 2015)** achieves last-iterate convergence in matrix zero-sum games.
- **Gradient descent-ascent with extragradient (Korpelevich 1976, Tseng 1995)** converges in continuous zero-sum games.

Convergence in potential games

Theorem. In finite potential games (games where a single potential function $\Phi(s)$ satisfies $u_i(s_i', s_{-i}) - u_i(s_i, s_{-i}) = \Phi(s_i', s_{-i}) - \Phi(s_i, s_{-i})$ for all i, s_i, s_i'), fictitious play converges to NE, best-response dynamics converge to NE, and any no-regret dynamics converges to the set of NE (in time-average).

The reason: the potential function is a Lyapunov function for the dynamics. Every best-response improvement increases Φ, and the dynamics converge to a local maximum of Φ, which is a Nash equilibrium.

Convergence in supermodular games

Theorem (Milgrom-Roberts 1990). In finite supermodular games (8.6), any adaptive learning dynamics converges to the interval $[\underline{s}, \bar{s}]$ between the least and greatest Nash equilibria.

The monotonicity of best responses ensures that the dynamics are sandwiched between two converging sequences (from the top and bottom of the strategy space), both of which converge to the equilibrium set.

The thesis framing: convergence as dynamics-objective compatibility

The convergence landscape can be organized as a compatibility matrix:

Game class	No-regret (MWU)	Fictitious play	CFR	Optimistic GDA
General	CCE ✓	May cycle	CCE ✓	May cycle
2P zero-sum	NE (avg) ✓	NE ✓	NE (avg) ✓	NE (last-iterate) ✓
Potential	NE (avg) ✓	NE ✓	N/A	NE ✓
Supermodular	NE interval ✓	NE ✓	N/A	NE ✓

The thesis claim: the right way to read this table is as a design guide. Given a game class (the "plant"), choose the learning algorithm (the "controller") that is compatible. Do not use fictitious play in general games (it may cycle). Do not use generic no-regret in two-player zero-sum if you want last-iterate NE (use optimistic methods instead). The choice is a control-design problem, not a mathematical inevitability.

Worked Example

Learning dynamics in a 2x2 coordination game

Game:

	A	B
A	$(3,3)$	$(0,0)$
B	$(0,0)$	$(2,2)$

Two pure NE: (A, A) and (B, B). One mixed NE: $(2/5, 2/5)$ (each plays A with probability $2/5$).

Fictitious play. Start with uniform prior. Period 1: both play A (best response to uniform). Period 2: empirical frequency is (A), so both play A again. Convergence to (A, A) immediately. If started with prior slightly favoring B: convergence to (B, B). Fictitious play converges to a pure NE determined by initial conditions. It does *not* converge to the mixed NE.

No-regret (MWU). Both players run MWU with learning rate η. The dynamics cycle near the mixed NE: weights oscillate around $(2/5, 3/5)$ but the time-average converges to the mixed NE (or to a CCE that includes it). Last-iterate does not converge – it cycles.

Interpretation. Different dynamics converge to different equilibria of the same game. The "right" equilibrium depends on the dynamics, not on the game alone. This is the selection-by-dynamics principle.

Learning in a 2-player zero-sum game (Matching Pennies)

Game: $(1,-1),(-1,1),(-1,1),(1,-1)$. Unique NE: $(1/2,1/2)$.

Fictitious play. Converges in time-average to $(1/2,1/2)$. Last-iterate cycles (Shapley 1964 showed this).

MWU. Time-average converges to NE. Last-iterate cycles (proven by Mertikopoulos et al. 2018).

Optimistic MWU. Last-iterate converges to $(1/2,1/2)$. This is the state of the art for zero-sum last-iterate convergence.

CFR. Time-average converges to NE. This is the algorithm that solved heads-up limit Texas hold'em (Bowling et al. 2015).

Poker Anchor

Concept mapping

Formal object	Poker instantiation
Nash equilibrium	GTO strategy – the mutual-best-response profile
Correlated equilibrium	A "correlated play" recommendation that no player deviates from
CCE	A weaker notion: no player can improve by committing to a fixed strategy regardless of what they're told
No-regret learning	A player who minimizes regret over the long run by adjusting their strategy
CFR convergence	The solver algorithm that computes NE in poker (two-player zero-sum extensive form)
Last-iterate convergence	Whether the solver's *current* strategy (not the average) is close to GTO
Time-average convergence	Whether the solver's *average* strategy over all iterations is close to GTO

Concrete situation: why solvers use CFR and not MWU

Poker solvers (PioSolver, GTO+, Libratus) use CFR or its variants (CFR+, DCFR, LCFR) rather than generic no-regret algorithms like MWU. The reason is structural compatibility: poker is a two-player zero-sum extensive-form game, and CFR is specifically designed for this game class. CFR exploits the extensive-form structure (traversing the game tree, computing counterfactual values at each information set) in a way that MWU cannot. The convergence rate of CFR in poker-sized games is practical ($O(1/\sqrt{T})$ for time-average NE, with DCFR achieving faster practical convergence); the convergence rate of generic MWU would be impractical for the same game sizes.

This is the thesis's point: the choice of learning algorithm (optimizer) must match the structure of the game (objective). CFR matched to extensive-form zero-sum is the right control design. MWU matched to the same game would technically converge but at impractical rates.

What this understanding buys you

Solver output is a time-average, not a last-iterate. When you load a PioSolver solution, you are seeing the time-averaged strategy over many CFR iterations, not the strategy from the last iteration. This matters because the time-average converges to NE while the last iterate may cycle. If you export a "partially converged" solver solution, the time-average is much closer to NE than the current-iteration strategy.

Convergence tells you what you can compute, not what opponents play. The convergence theory says that CFR finds NE of two-player zero-sum poker. It does not say that your opponents play NE. The gap between "what the solver computes" and "what opponents do" is the gap between equilibrium and learning dynamics. Understanding the convergence landscape helps you understand where solver output is reliable (in well-structured game classes) and where it is not (in multi-player or non-zero-sum settings).

Cross-Links

- **9.1-9.4** – the algorithms whose convergence this node unifies.
- **7.5 Stochastic stability** – evolutionary selection among equilibria.
- **7.6 Adaptive dynamics** – the gradient-flow analogue in evolutionary settings.
- **8.6 Strategic complementarities** – the game class with the strongest convergence results.
- **5.4 Folk theorem** – the multiplicity that convergence theory must address.
- **10.6 Strategic ecosystems** – meta-level design of games and learning dynamics together.
- **VQA 6-7 (Optimization dynamics, control view)** – the thesis's framework for separating objective from dynamics.

Structural Compression

Invariant: No-regret learning converges (in time-average) to coarse correlated equilibrium in all games, to correlated equilibrium with internal-regret minimization, and to Nash equilibrium in two-player zero-sum and potential games. No uncoupled dynamics converges to Nash in all games (Hart-Mas-Colell impossibility). The convergence target depends on the pairing of learning dynamics with game structure.

Mechanism: Regret bounds imply time-average convergence via the equivalence between no-regret and approximate CCE/CE conditions; game-class-specific structure (zero-sum, potential, supermodular) provides additional convergence guarantees by supplying Lyapunov functions, minimax duality, or monotonicity.

Cross-System Echo: The convergence landscape of learning in games is structurally isomorphic to the convergence landscape of optimizers on cost functions in VQA. The game class (zero-sum, potential, supermodular) maps to the cost-landscape class (convex, non-convex, noisy). The learning algorithm (MWU, CFR, fictitious play) maps to the optimizer (Adam, HOPSO, natural gradient). The convergence theorem specifies which (optimizer, landscape) pairs converge. The thesis's claim is that this isomorphism is not accidental: both are instances of the same abstract problem – a controller navigating a state-dependent objective using local feedback – and the design principles for good controllers are the same in both settings.

Exercises

1. Show that if both players in a two-player zero-sum game use no-regret learning, the time-average strategy converges to a Nash equilibrium. (Hint: use the minimax theorem and the definition of regret.)
2. Construct a 3-player game where fictitious play cycles forever. (Hint: use Shapley's example or a modification.)
3. Prove the Hart-Mas-Colell impossibility: no uncoupled dynamics converges to NE in all finite games. (Outline the proof structure and identify the key counterexample.)
4. In a 2x2 coordination game with two pure NE, run MWU for both players for 1000 iterations (numerically). Plot the last-iterate strategies and the time-average strategies. Verify that the time-average converges to a CCE while the last-iterate cycles.
5. Show that in a finite potential game, best-response dynamics converge to a Nash equilibrium in finite time. (Hint: the potential function increases at each step and the strategy space is finite.)

6. For the linear-quadratic network game of 8.6, show that iterated best-response dynamics converge to the unique Nash equilibrium. (Hint: the best-response map is a contraction for small λ.)

7. **Argue, structurally,** why the thesis's "separation of objective from dynamics" principle applies to learning in games. Identify the objective, the dynamics, and the feedback loop. What does the Hart-Mas-Colell impossibility tell us about the limits of this framework, and how does game-class restriction (zero-sum, potential) overcome those limits?

Reinforcement Learning in Games

Position in Knowledge Graph

System: Game Theory **Structural Domain:** Learning in Games **Node:** 9.6

Reinforcement learning in games drops the most restrictive assumption of the classical learning framework: that players know the payoff function. In fictitious play (9.1), regret matching (9.2), and CFR (9.3), each player observes the full loss vector — the payoff of every action, not just the one chosen. RL agents observe only the payoff of the action they took (the "bandit" feedback model), and must balance *exploration* (trying suboptimal actions to learn about them) with *exploitation* (playing the currently best-known action). This shift from full-information to partial-information learning changes everything: convergence is harder to guarantee, rates are slower, and the multi-agent setting introduces non-stationarity that breaks single-agent RL guarantees. Despite these challenges, RL in games has produced some of the most spectacular AI results of the past decade: AlphaGo and AlphaZero (perfect-information board games), AlphaStar (StarCraft), OpenAI Five (Dota 2), and actor-critic methods applied to poker and other imperfect-information domains. This node develops the theory of RL in games — Q-learning, policy gradient, self-play, multi-agent RL — and maps the structural connections between RL and the game-theoretic learning dynamics of the rest of Module 9.

Formal Definition

Single-agent RL (background)

An agent interacts with an environment modeled as a Markov Decision Process (MDP): (S, A, P, R, γ) where S is the state space, A the action space, $P(s'|s,a)$ the transition kernel, $R(s,a)$ the reward function, and $\gamma \in [0,1)$ the discount factor. A policy $\pi : S \to \Delta(A)$ maps states to action distributions. The value function:

$$V^\pi(s) \;=\; \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \;\middle|\; s_0 = s\right],$$

and the action-value function:

$$Q^\pi(s,a) \;=\; R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s').$$

The optimal policy π^* satisfies $V^{\pi^*}(s) = \max_\pi V^\pi(s)$ for all s.

Q-learning (Watkins 1989)

A model-free algorithm that learns Q^* directly from experience:

$$Q(s,a) \;\leftarrow\; Q(s,a) + \alpha\left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right],$$

where α is the learning rate and (s, a, R, s') is an observed transition. Under standard assumptions (every state-action pair visited infinitely often, α decaying appropriately), Q-learning converges to Q^* in single-agent settings.

Multi-agent extension: stochastic games

A **stochastic game** (Shapley 1953) generalizes MDPs to n players: $(S, \{A_i\}, P, \{R_i\}, \gamma)$, where the transition and rewards depend on the joint action profile. Each player's MDP is non-stationary from their perspective,

because the other players' policies change over time. This non-stationarity is the fundamental obstacle: single-agent RL convergence theorems assume a stationary environment, which multi-agent play violates.

Independent Q-learning

Each player runs Q-learning independently, treating other players as part of the environment:

$$Q_i(s, a_i) \;\leftarrow\; Q_i(s, a_i) + \alpha \left[R_i(s, a_i, a_{-i}) + \gamma \max_{a_i'} Q_i(s', a_i') - Q_i(s, a_i) \right].$$

No convergence guarantee. Since each player's "environment" changes as others learn, the stationarity assumption of Q-learning is violated. Independent Q-learning can diverge, cycle, or converge to suboptimal equilibria. Empirically, it sometimes works in simple games but frequently fails in complex ones.

Self-play

An agent plays against copies of itself (or historical versions of itself). The resulting training process generates a sequence of policies $\pi^0, \pi^1, \pi^2, \ldots$ where each π^{t+1} is trained against π^t (or a mixture of past policies). In two-player zero-sum games, self-play with appropriate averaging converges to Nash — this is because self-play is effectively a form of fictitious play or no-regret learning where the "opponent model" is the agent's own past.

Policy gradient methods

Instead of learning Q-values, directly parameterize the policy π_θ and update via gradient ascent on expected reward:

$$\theta \;\leftarrow\; \theta + \alpha \nabla_\theta J(\theta), \qquad J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_t \gamma^t R_t \right].$$

The REINFORCE estimator (Williams 1992):

$$\nabla_\theta J(\theta) \;=\; \mathbb{E} \left[\sum_t \nabla_\theta \ln \pi_\theta(a_t | s_t) \cdot G_t \right],$$

where $G_t = \sum_{\tau \geq t} \gamma^{\tau - t} R_\tau$ is the return from time t.

Actor-critic methods

Combine a policy (the "actor") with a value-function estimator (the "critic"). The actor updates via policy gradient using the critic's value estimates as a baseline, reducing variance:

$$\theta \;\leftarrow\; \theta + \alpha \nabla_\theta \ln \pi_\theta(a_t | s_t) \cdot \left(R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \right),$$

where V_ϕ is the critic's learned value function. In multi-agent settings, each player can maintain their own actor-critic, leading to independent actor-critic (IAC), or a centralized critic can observe the joint state while actors observe only local information (centralized training, decentralized execution — CTDE).

Intuition

The exploration-exploitation tradeoff. In full-information settings (9.2–9.4), the learner sees the payoff of every action at each round, so there is no need to explore — you can evaluate all actions without trying them. In RL, you only observe the payoff of the action you took. To estimate the value of an untried action, you must try it — potentially sacrificing payoff in the process. This tradeoff is fundamental and does not exist in the full-information regime.

Non-stationarity is the killer. The single-agent RL convergence theory (Q-learning converges, policy gradient is unbiased, etc.) assumes a *stationary* environment. In a game, the environment is the other players, who are also learning. From each player's perspective, the reward function and transition kernel change over time as opponents change their policies. This makes the "MDP" that each player faces non-stationary, breaking every single-agent guarantee. The multi-agent RL (MARL) field is largely about finding conditions under which convergence can be restored despite this non-stationarity.

Self-play as a structural fix. In two-player zero-sum games, self-play sidesteps the non-stationarity problem by making the opponent a function of the learner's own history. If the learner improves, the opponent improves at the same rate (because it is the same agent). This creates a "co-evolution" dynamic that, under appropriate conditions, converges to Nash. Self-play is the RL analogue of the no-regret convergence theorem: in zero-sum games, the structure of the game forces convergence even though each individual learner faces a moving target.

Deep RL: function approximation changes the game. When Q-functions or policies are represented by neural networks rather than tables, new phenomena arise: approximation error, catastrophic forgetting, training instability, and mode collapse. These are not game-theoretic issues per se, but they interact with the game-theoretic convergence properties in complex ways. AlphaZero's success shows that deep RL self-play can work spectacularly in practice (in perfect-information games); AlphaStar shows that it can work in imperfect-information games with massive state spaces; but theoretical guarantees for deep RL in games remain sparse.

Structural Role

9.6 is the bridge from classical game-theoretic learning (9.1–9.5) to modern AI:

- **From 9.1 (fictitious play):** Self-play is RL's version of fictitious play — iterate best responses against the opponent's historical play.
- **From 9.2 (no-regret):** Policy gradient with entropy regularization can be interpreted as running a no-regret algorithm (MWU variant) in policy space.
- **From 9.3 (CFR):** CFR requires a model of the game tree; RL does not. RL can learn in games where the tree is unknown or too large to enumerate.
- **From 9.5 (convergence):** RL inherits all the convergence difficulties of the full-information setting, plus the additional problems of partial information and function approximation.

9.6 also connects outward to the AI milestones that brought game-theoretic concepts to public attention:

- **AlphaGo (2016, Silver et al.):** Deep RL + Monte Carlo tree search in the perfect-information game of Go.
- **AlphaZero (2017, Silver et al.):** Self-play RL from scratch (no human data) in Go, chess, and shogi. Demonstrates that self-play + deep RL + MCTS can rediscover and surpass centuries of human knowledge in perfect-information games.
- **AlphaStar (2019, Vinyals et al.):** Deep RL self-play in StarCraft II, an imperfect-information, real-time strategy game with massive state and action spaces. Used a league of agents with diverse strategies to avoid self-play convergence failures.
- **OpenAI Five (2019):** Deep RL in Dota 2 (5v5 team game), trained via self-play with long-horizon RL.

Mathematical Development

Failures of independent Q-learning

Example: coordination game. Two players, two actions each: (A, A) = (1, 1), (B, B) = (1, 1), (A, B) = (0, 0), (B, A) = (0, 0). Two Nash equilibria: (A, A) and (B, B). Independent Q-learning with ϵ-greedy exploration: each player estimates Q-values for A and B based on observed rewards. If player 1 happens to explore B while player 2 plays A, both observe zero reward, and player 1 decreases $Q_1(B)$. The learning dynamics are path-dependent: the equilibrium reached depends on the random exploration sequence. In some runs, the players converge to (A, A); in others, to (B, B); and in degenerate cases, they cycle between the two.

Example: Prisoner's Dilemma. Independent Q-learning converges to (D, D) — the stage Nash — because defection dominates. No exploration can discover that (C, C) is Pareto-superior, because each player's Q-values correctly reflect the fact that cooperating against a defector is worse than defecting. This is not a failure of the algorithm; it is a correct equilibrium computation. But it shows that RL finds Nash, not welfare optima.

Example: matching pennies. Independent Q-learning can fail to converge entirely. The Q-values oscillate as each player chases the other's strategy. With function approximation (deep Q-learning), the oscillations can be amplified, leading to divergence.

Nash Q-learning (Hu and Wellman 2003)

A modification of Q-learning where each player, instead of maximizing their own Q-value, computes a Nash equilibrium of the stage game defined by the current Q-values:

$$Q_i(s, a_i, a_{-i}) \leftarrow Q_i(s, a_i, a_{-i}) + \alpha \left[R_i + \gamma \, \text{Nash}_i(Q(s', \cdot)) - Q_i(s, a_i, a_{-i}) \right],$$

where $\text{Nash}_i(Q(s', \cdot))$ is player i's payoff in the Nash equilibrium of the matrix game $Q(s', \cdot, \cdot)$.

Convergence. Nash Q-learning converges in two-player zero-sum stochastic games (because Nash of a zero-sum matrix game is unique and efficiently computable) and in certain special classes of general-sum games. It does not converge in general (because computing Nash of a general matrix game is PPAD-hard and may not be unique).

Policy gradient in multi-agent settings

Each player i parameterizes their policy as π_{θ_i} and updates via:

$$\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J_i(\theta_1, \ldots, \theta_n),$$

where J_i is player i's expected reward under the joint policy. This is **gradient play** — each player follows the gradient of their own objective. The fixed points of gradient play are the first-order stationary points of the game, which include Nash equilibria but also non-Nash stationary points.

Convergence in zero-sum games. In two-player zero-sum games, policy gradient with appropriate learning rates converges to Nash in time-averaged play (connecting to the gradient-dynamics analysis of 9.5).

Divergence in general-sum games. In general-sum games, gradient play can cycle (9.5), and policy gradient inherits this problem. Adding entropy regularization (SAC, maximum-entropy RL) can stabilize the dynamics by making the objective strongly concave, but changes the target equilibrium (the entropy-regularized Nash is not the same as Nash).

Self-play convergence

Theorem (informal, Heinrich and Silver 2016). In two-player zero-sum games, neural fictitious self-play (NFSP) — where each player trains a best-response network via RL and an average-strategy network via

supervised learning on its own past play — converges to an approximate Nash equilibrium. The rate depends on the function approximation quality and the mixing of best-response and average-strategy play.

NFSP is the RL analogue of CFR: it approximates the per-info-set regret-matching process of CFR using neural networks to represent the strategy, and substitutes RL (DQN) for the exact value computation. The convergence guarantee is weaker than CFR's (no exploitability bound in the general case), but NFSP can handle games too large for explicit CFR.

AlphaZero: the deep RL paradigm for perfect-information games

AlphaZero (Silver et al. 2017) combines:

1. **Monte Carlo tree search (MCTS):** At each position, build a search tree by simulating many rollouts, guided by a neural network that estimates value and policy.
2. **Self-play:** Generate training data by playing games against itself.
3. **Neural network training:** Train the network to predict (a) the MCTS policy output and (b) the game outcome from each position.

The training loop is: self-play → generate data → train network → use improved network for next round of self-play. After many iterations, the policy and value estimates converge to near-optimal play. In chess, Go, and shogi, AlphaZero achieved superhuman performance within hours of training.

Game-theoretic interpretation. AlphaZero's self-play is a form of policy iteration in a perfect-information zero-sum game. Each iteration of self-play computes an improved best response to the current policy (via MCTS), and the network training aggregates these improvements into a single policy. This is structurally similar to fictitious play, but with deep function approximation replacing the empirical frequency.

Limitation: perfect information only. AlphaZero relies on MCTS, which requires the ability to simulate the game from any state. In imperfect-information games (poker), you cannot simulate from a single state because you do not know the opponent's private information. This is why poker AI uses CFR-based methods rather than AlphaZero-style MCTS.

AlphaStar and league training

AlphaStar (Vinyals et al. 2019) tackled StarCraft II, an imperfect-information, real-time strategy game with $\sim 10^{26}$ possible actions per step. Key innovations:

1. **League training:** Instead of pure self-play (which can converge to a narrow strategy that beats itself but loses to diverse opponents), AlphaStar maintains a "league" of agents with diverse strategies. New agents are trained against the league, and the league is updated to include strong and diverse policies.
2. **Population-based training:** Multiple agents are trained in parallel with different hyperparameters, and the best performers seed the next generation.
3. **Exploiter agents:** Special agents trained to exploit weaknesses in the current league members, forcing the league to be robust.

The league structure is an engineering solution to the non-convergence problem in general-sum games: by maintaining a diverse population, AlphaStar avoids the cycling and instability that plague simple self-play in non-zero-sum settings.

Worked Example

Independent Q-learning in Rock-Paper-Scissors

Two players, each running ϵ-greedy Q-learning with $\epsilon = 0.1$, $\alpha = 0.01$, against each other. Payoff matrix for Row:

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}.$$

Initialize $Q_i(a) = 0$ for all i, a. Each round: both players choose actions (ϵ-greedy on Q-values), observe rewards, update Q-values.

After 10,000 rounds, the Q-values oscillate without converging to a fixed point. The empirical action frequencies converge (slowly) to approximately $(1/3, 1/3, 1/3)$ for both players — the Nash equilibrium — because the time-averaging effect smooths out the oscillations. But the Q-values themselves do not stabilize: $Q_i(R)$, $Q_i(P)$, $Q_i(S)$ fluctuate around 0 as the opponents' strategies shift.

Compare to regret matching (9.2), which converges cleanly and monotonically. The RL version achieves the same long-run result but with much higher variance and no per-iteration exploitability guarantee.

Self-play policy gradient in a simple poker game

Consider Kuhn Poker (the toy game from 9.3). Train two policy-gradient agents via self-play:

1. Parameterize each player's policy by a small neural network mapping info-set features to action probabilities.
2. Play 1,000 hands of self-play, collecting trajectories.
3. Compute policy-gradient updates for both players.
4. Repeat.

After 100,000 episodes, the policies approximate the Nash equilibrium of Kuhn Poker (see 9.3 for the analytic solution). The approximation quality depends on the network architecture and learning rate. Typical exploitability: 0.01-0.05 chips, compared to CFR's 0.001 after the same number of game evaluations. The RL approach is less sample-efficient but does not require knowledge of the game tree.

Poker Anchor

Concept mapping

Formal object	Poker instantiation
MDP / stochastic game	The poker game as a sequential decision process with unknown opponent policy
Q-function $Q(s, a)$	EV of taking action a at game state s (hand + board + history)
Policy π_θ	A neural-network strategy mapping info sets to action distributions
Self-play	Training by playing against yourself (or past versions of yourself)
Exploration (ϵ-greedy)	Occasionally making "random" plays to discover new strategic information
Function approximation	Using a neural net to generalize across similar poker situations
Non-stationarity	Your opponent is also learning, so the "environment" changes over time

Concrete situation: why poker AI uses CFR, not pure RL

CFR (9.3) has two advantages over RL for poker:

1. **Full-information feedback.** CFR's tree traversal computes counterfactual values for *every* action at *every* info set, even ones not taken. RL only observes the reward of the action taken. This makes CFR far more sample-efficient: one CFR iteration updates regrets at every info set, while one RL episode updates the policy only along the single trajectory played.
2. **Convergence guarantees.** CFR has a provable $O(T^{-1/2})$ exploitability bound. RL in multi-agent settings has no comparable guarantee. The non-stationarity of multi-agent learning means that RL can cycle, diverge, or converge to exploitable strategies.

However, RL has advantages in settings where CFR is impractical:

- **Unknown game rules.** CFR requires a complete model of the game tree. RL can learn from interaction without knowing the rules.
- **Continuous action spaces.** CFR requires discretizing actions; RL can handle continuous actions via policy gradient.
- **Very large games.** When the game tree is too large even for MCCFR with abstraction, RL with function approximation can still learn (AlphaStar, OpenAI Five).

The state of the art in poker AI is a hybrid: CFR for the core equilibrium computation (where convergence guarantees matter), RL-style techniques for value estimation in subgame solving (DeepStack's neural-net value function).

Concrete situation: training a poker bot via self-play RL

If you were to train a HUNL poker bot using pure RL self-play (no CFR, no game tree model), you would:

1. Parameterize the bot's strategy as a neural network mapping (hand features, board features, action history) to action probabilities.
2. Play millions of hands against itself.
3. After each hand, compute policy-gradient updates using the hand's reward.
4. Repeat, periodically evaluating exploitability against a known GTO strategy.

This approach works in principle but is enormously less efficient than CFR. The key bottleneck is exploration: in HUNL, the space of possible game states is 10^{161}, and each self-play hand explores only one trajectory. CFR, by contrast, updates all info sets in each iteration. Empirically, pure RL self-play in HUNL requires orders of magnitude more compute than CFR to reach the same exploitability.

What this understanding buys you

CFR and RL are complementary, not competing. CFR gives you convergence guarantees and sample efficiency in games with known structure. RL gives you flexibility in games with unknown structure or continuous actions. Modern poker AI (DeepStack, Libratus, Pluribus) uses both: CFR for equilibrium computation, neural networks (trained by supervised learning or RL) for value estimation in real-time subgame solving.

Self-play is not magic. Self-play works well in two-player zero-sum games because of the minimax theorem: improving against yourself improves against everyone. In general-sum or multiplayer games, self-play can converge to narrow strategies that exploit their own weaknesses without being robust to diverse opponents. This is why AlphaStar uses league training rather than pure self-play, and why Pluribus's self-play blueprint is supplemented with online subgame solving.

The exploration-exploitation tradeoff in poker study. When you sit down to play and decide between "play GTO" (exploit) and "try a new line to see what happens" (explore), you are instantiating the exploration-exploitation tradeoff of RL. GTO play exploits your current knowledge; deviating from GTO explores new strategic territory. The theory says some exploration is necessary for long-run improvement, but too much exploration costs money in the short run. The optimal balance depends on your current skill level, the stakes, and how far your strategy is from optimal.

Cross-Links

- **9.1 Fictitious play** — self-play is the RL analogue.
- **9.2 No-regret learning** — policy gradient with entropy is an approximate no-regret method.
- **9.3 CFR** — the model-based alternative with convergence guarantees.
- **9.5 Convergence to equilibrium** — RL inherits all convergence difficulties plus sampling noise.
- **7.3 Replicator dynamics** — population-based RL training echoes evolutionary dynamics.
- **3.4 Backward induction** — AlphaZero's MCTS is a learned version of backward induction.
- **MDP and control theory** — single-agent RL foundations.
- **Deep learning** — function approximation for large state spaces.

Structural Compression

Invariant: Reinforcement learning in games replaces full-information feedback with bandit feedback (observing only the payoff of the chosen action), introducing exploration-exploitation tradeoffs and non-stationarity from other learners; self-play in two-player zero-sum games converges to Nash under appropriate conditions, but independent RL in general games has no convergence guarantee to Nash.

Mechanism: Q-learning or policy gradient updates estimate action values or policy gradients from sampled trajectories. Self-play generates training data against the agent's own (evolving) policy. Multi-agent non-stationarity is mitigated by population-based training (league), centralized critics (CTDE), or averaging over policy histories (NFSP). Function approximation (neural networks) enables scaling to large state spaces at the cost of losing convergence guarantees.

Cross-System Echo: Multi-agent RL is the computational instantiation of **co-evolution in biology**: two populations adapting to each other in real time, with no fixed fitness landscape. The Red Queen effect — "you have to keep running just to stay in the same place" — is the biological analogue of non-stationarity in multi-agent learning. League training (AlphaStar) is the computational analogue of maintaining biodiversity to prevent evolutionary dead ends. The structural lesson is universal: in any system where multiple adaptive agents interact, the "environment" is not fixed, and single-agent optimization principles must be augmented with game-theoretic reasoning about the co-adaptation of others.

Exercises

1. Implement independent Q-learning for two players in Rock-Paper-Scissors. Run for 100,000 rounds and plot the Q-values and empirical action frequencies over time. Does the empirical frequency converge to Nash? Do the Q-values converge?

2. Show that independent Q-learning can converge to different Nash equilibria in a coordination game depending on the exploration sequence. Run 100 independent trials and report the distribution of outcomes.

3. Derive the policy-gradient theorem: $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_t \nabla_\theta \ln \pi_\theta(a_t|s_t) \cdot G_t]$. Why is this an unbiased estimator of the gradient of expected reward?

4. Explain why AlphaZero's approach (MCTS + self-play + neural networks) cannot be directly applied to poker. What specific structural feature of poker (imperfect information) prevents MCTS from being used without modification?

5. Compare the sample efficiency of CFR and self-play RL on Kuhn Poker. How many game evaluations does each require to reach exploitability below 0.01 chips? What accounts for the difference?

6. Describe the non-stationarity problem in multi-agent RL and explain how league training (AlphaStar) mitigates it. What is the game-theoretic interpretation of maintaining a diverse league?

7. **Argue, structurally,** why the combination of RL (for value estimation) and CFR (for equilibrium computation) is more powerful than either alone for solving large imperfect-information games. What does each method contribute that the other cannot?