PokerAI

Agriya Yadav, Rachit Korrapati

November 2025

1 Introduction to Kuhn Poker

Kuhn Poker is a very small, simplified poker game introduced by Harold Kuhn in 1950. It uses only three cards (Jack, Queen, King) and two players. Each player antes one chip, receives one private card, and can either check or bet. The player with the higher card wins at showdown.

Despite its size, the game captures key elements of real poker: hidden information, sequential decisions, and bluffing. Because of its simplicity, it is widely used as a testing ground for algorithms that deal with imperfect information.

2 What is Counterfactual Regret Minimization (CFR)?

CFR is an algorithm that learns good strategies by repeatedly simulating the game and measuring *regret*—how much better a different action would have performed in hindsight. Over many iterations:

- The algorithm tries different actions in each decision point.
- After each simulated game, it checks which actions would have led to better outcomes.
- Actions that would have done better accumulate positive regret.
- The strategy gradually shifts toward actions with higher regret.

In simple terms, CFR improves by learning *what it should have done*, and adjusts future decisions accordingly. As regret decreases, the average strategy approaches a Nash equilibrium.

3 Why CFR Works Here

Bad actions accumulate regret, good actions build positive preference. Across many training rounds, the algorithm naturally learns a stable, hard-to-exploit strategy. This is why CFR is the foundation of this poker AI.

4 RL-CFR: Making CFR Scalable

Traditional CFR explores the entire game tree every iteration. This is fine for small games like Kuhn Poker but impossible for huge games such as Texas Hold'em.

RL-CFR overcomes this using reinforcement learning techniques:

4.1 Monte Carlo Sampling

Instead of traversing the entire tree, RL-CFR:

- Considers all actions for the player being trained.
- Samples only one action for the opponent, based on their current strategy.

This makes each iteration much faster.

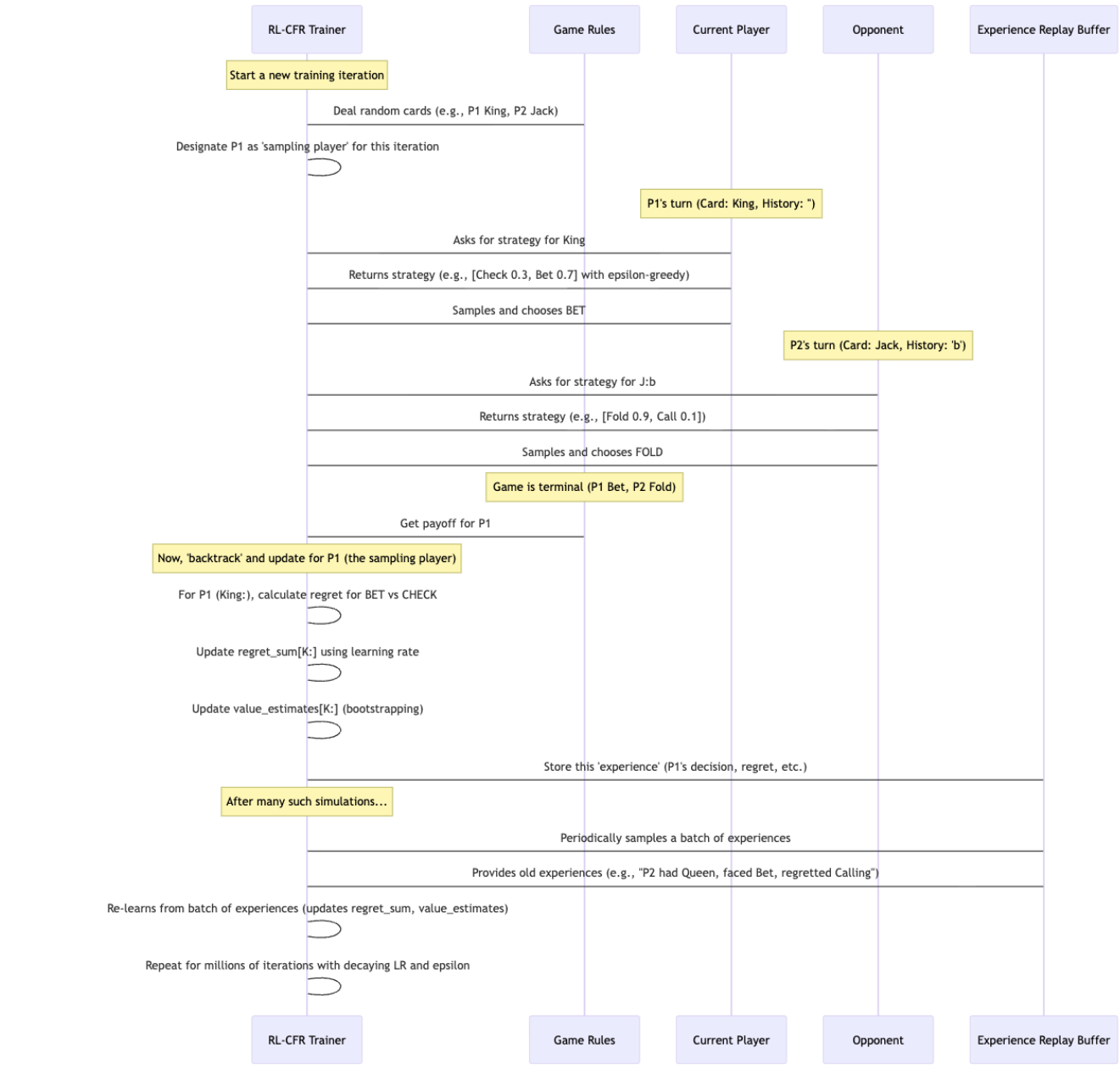

Figure 1: simplified view of the RL-CFR process

4.2 Experience Replay

Similar to modern reinforcement learning, RL-CFR stores past decision points in a replay buffer. These stored experiences are reused multiple times, improving sample efficiency and stabilizing training.

4.3 Learning Rates and Bootstrapping

Instead of updating regrets exactly every time, RL-CFR uses a learning rate to smooth updates. For situations not recently encountered, the algorithm estimates values based on past information. This helps it handle much larger game trees.

4.4 Exploration

RL-CFR includes explicit exploration (e.g. ϵ-greedy), ensuring the algorithm does not get stuck repeating the same actions. Over time, exploration decreases as the strategy becomes confident.

5 Comparison

Feature	**Standard CFR**	**RL-CFR**
Tree Traversal	Full tree	Sampled paths
Memory Use	Higher	Lower (plus replay buffer)
Speed per Iteration	Slower	Faster
Exploration	Implicit	Explicit control
Scalability	Limited	Much better

Table 1: Key differences between CFR and RL-CFR.

In Kuhn Poker, both algorithms eventually reach the known Nash equilibrium. RL-CFR, however, is far more scalable and is well-suited to large games like Texas Hold'em, where the game tree is astronomically large.

6 Inferences

Our simplified RL-CFR implementation for Kuhn Poker establishes a practical starting point for building regret-based decision-making systems. Moving forward, the goal is to extend these ideas to a heads-up No-Limit Hold'em (HUNL) poker AI. This transition is motivated by several factors:

- **HUNL is vastly more complex**: it contains enormous game trees, hidden private information, and long betting sequences that make exhaustive search or brute-force methods infeasible.
- **CFR naturally scales to imperfect-information settings**: it does not require exploring every possible state and instead focuses on learning from regret across repeated simulations.
- **RL-CFR further improves scalability**: Monte Carlo sampling, experience replay, and adaptive regret updates enable the algorithm to operate efficiently in very large decision spaces.
- **The approach converges toward minimally exploitable strategies**: this is essential for poker, where even small strategic leaks can be heavily punished by strong opponents.

Kuhn Poker therefore acts not as the final objective but as a controlled experimental sandbox. The ideas refined here—regret matching, sampling-based updates, and strategy averaging—form the backbone of modern poker AIs. They provide a clear and theoretically grounded pathway toward constructing a competitive heads-up No-Limit Hold'em agent.

7 The Core Mathematical Difference

7.1 Q-Learning: Single-Agent Optimization

Q-Learning estimates a value $Q(s, a)$ for every state–action pair:

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

This yields a deterministic policy:

$$\pi(s) = \arg\max_{a} Q(s,a).$$

Limitation for Poker: the opponent is assumed to be part of a stationary environment. In poker, an intelligent opponent adapts to your behaviour, making deterministic policies trivially exploitable.

7.2 CFR: Regret Minimization and Mixed Strategies

CFR instead minimizes *regret*:

$$R^T(I,a) = \sum_{t=1}^{T} \left(v^t(I,a) - v^t(I) \right).$$

The strategy uses regret matching:

$$\sigma(I,a) = \frac{\max(0, R^T(I,a))}{\sum_{a'} \max(0, R^T(I,a'))}.$$

Why this matters: CFR naturally produces **mixed strategies**. This is essential for poker because optimal strategy requires deliberate randomization (e.g., bluffing with weak hands, slow-playing strong hands).

7.3 Nash Equilibrium Convergence

CFR guarantees:

$$\epsilon^T \leq O\left(\frac{1}{\sqrt{T}} \right),$$

where ϵ^T is the exploitability of the average strategy. As $T \to \infty$, exploitability goes to zero. Q-Learning has no such guarantee.

8 Why Q-Learning Fails for HUNL Poker

8.1 Issue 1: Deterministic Policies

Q-Learning outputs deterministic actions. Opponents quickly adapt:

- If you always bet medium hands, opponents only call with strong hands.
- If you always check, opponents bluff relentlessly.

8.2 Issue 2: Non-Stationarity in Self-Play

Self-play causes:

1. Agent A learns against B.
2. B adapts.
3. A's Q-values become obsolete.
4. Training destabilizes and oscillates.

CFR sidesteps this by accumulating regret across all iterations.

8.3 Issue 3: Hidden Information

Q-Learning must define a *state*. In poker, the true state includes opponent cards—which are hidden. CFR instead uses *information sets*, which is the mathematically correct abstraction.

9 Empirical Evidence

Leading poker AIs use CFR variants:

- **Libratus (2017)** — CFR+
- **Pluribus (2019)** — Monte Carlo CFR

No top poker AI relies primarily on Q-Learning.

10 When Q-Learning Can Still Help

Q-Learning is useful for:

- Exploiting weaker opponents
- Real-time adaptation beyond equilibrium play

But it must sit *on top of* a CFR-trained Nash baseline.

11 Inferences

For Heads-Up No-Limit Hold'em, CFR is the superior foundation because:

1. **It converges to Nash equilibrium (unexploitable play).**
2. **It produces mixed strategies essential for poker.**
3. **It correctly handles hidden information.**
4. **It offers far greater stability in self-play.**

CFR provides the strategic backbone of every modern competitive poker AI. Q-Learning alone simply cannot meet the requirements of a game as adversarial and information-constrained as poker.