AI III

Garrett Barber, Matthew Guiddy,
Makye Daniels, Sam Brunacini

The History of AI in Poker

The Beginning

- It all began in when “Theory of Games and Economic Behavior” was published in 1944
    - Published by mathematician John von Neumann and economist Oskar Morgenstern
    - The goal was to model economic decision-making
    - A simplified version of poker was picked
- This is considered to have started the field of game theory.
    - Game theory is a branch of mathematics that deals with analysing strategy games
- After this paper, many games such as checkers were studied and some were even “solved”
- Poker remained a difficult problem that few were willing to tackle

Why Poker?

- There are countless challenges to making a good decision
- Your actions are heavily dependent on those of your opponents
- You can never know if you made the correct decision until afterwards

General AI Problems:

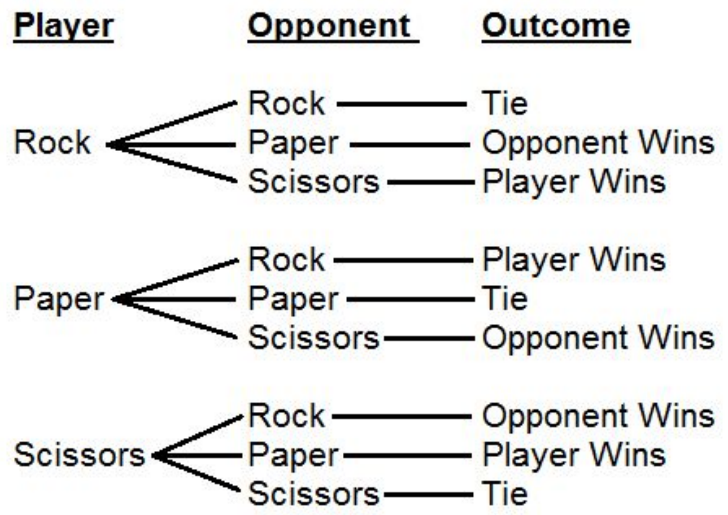
- Incomplete information
- Untrustworthy information
- Deception
- Agent modeling
- Risk Management

Poker Problems:

- The hands of the opponents are hidden
- The opponents might be bluffing
- Being able to bluff is integral
- Identify and exploit patterns
- Betting strategies and consequences

The Evolution of Poker AI

- One of the earliest attempts was developed by the University of Alberta
  - A method for modeling all decisions involved in poker
  - Simplified Texas Hold 'em
    - Only two players
    - Predetermined bet sizes
    - 316,000,000,000,000,000 decision branches
- The next development was to group hands together
  - E.g. treat pairs of 9's and 10's equally
  - The goal was to reduce the number of branches
- Counterfactual Regret Minimization
  - Groundbreaking in the field of game theory and for Poker AIs
- Game Theory Optimal (GTO) Poker



Our Easy AI

Easy AI Overview

- Goals:
    - To create an AI that is easy to beat for beginner poker players
    - To create an AI that would make decisions loosely based on the strength of their hand
- Implementation:
    - Not based on a decision tree like traditional AIs
    - The AI will primitively determine if it has a strong hand
    - Decisions are made using a random number generator
    - Different hand strengths have different decision distributions

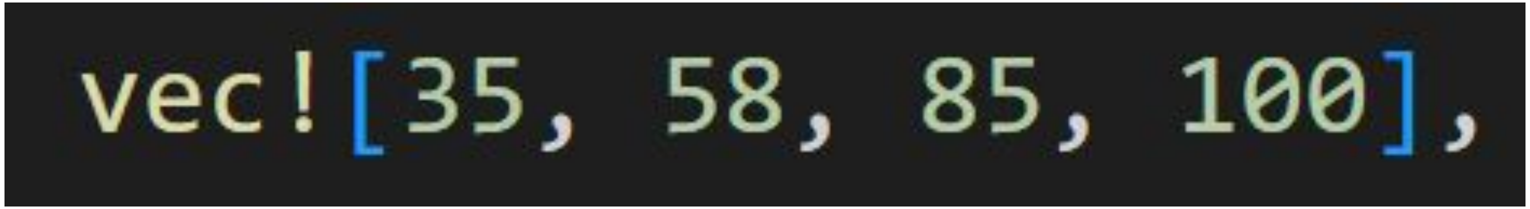
Step 1: Determining hand strength

- Pre flop hand strength:
  - The AI only has 2 cards in its hand at this point
  - Hand strength is determined by adding up the values from each card
  - E.g. A hand with a 10 of hearts and a 2 of spades will have a hand strength of 12
- Post flop hand strength:
  - From here on, the AI will have 5 cards in its hand
  - Hand strength will be determined using our hand evaluation algorithm
    - The hand strength ranks hands based on their type (pair, straight, flush, etc.)
    - Highest rank is 8 for a straight flush and the lowest is 0 for a high card

Step 2: Assigning a Move Distribution

- The move distributions change the odds that each move gets called by the random number generator
- Each hand strength from pre and post flop have its own move distribution

Check | Fold | Call | Raise



```
vec![35, 58, 85, 100],
```

Step 3: Making a Decision

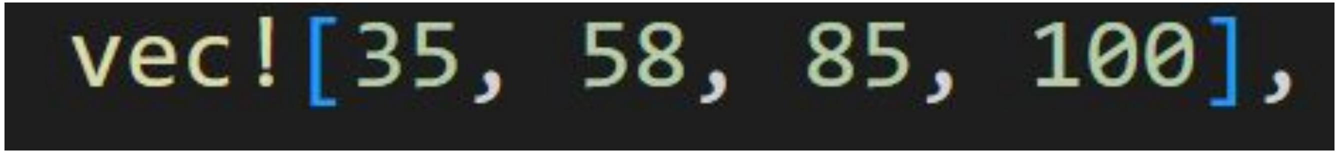
Pair of 2s

- When 0 <= x >= 35, the AI checks (35.6%)
- When 36 <= x >= 58, the AI folds (22.8%)
- When 59 <= x >= 85, the AI calls (26.7%)
- When 86 <= x >= 100, the AI raises (14.9%)

Pair of Aces

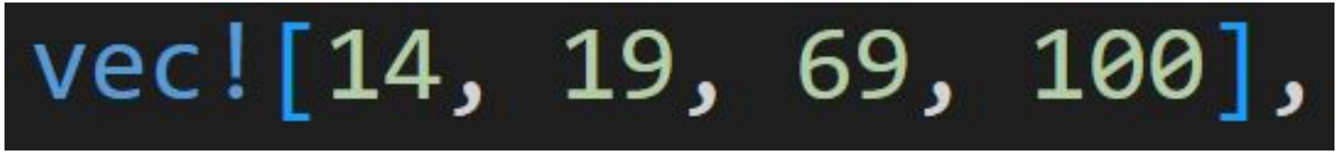
- When 0 <= x >= 14, the AI checks (14.9%)
- When 15 <= x >= 19, the AI folds (4.9%)
- When 20 <= x >= 69, the AI calls (49.5%)
- When 70 <= x >= 100, the AI raises (30.7%)

Pair of 2s:



```
vec![35, 58, 85, 100],
```

Pair of Aces:



```
vec![14, 19, 69, 100],
```

Step 3 Cont: Making a Decision

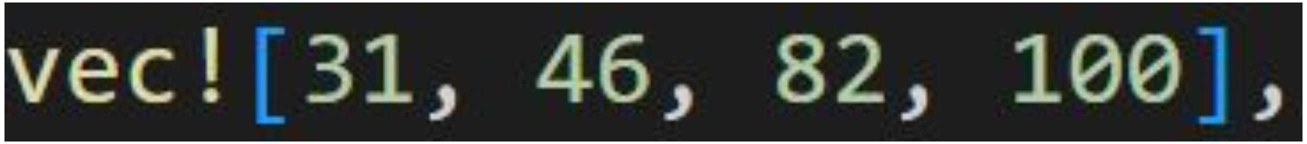
Hand strength 12

- When 0 <= x >= 31, the AI checks (31.7%)
- When 32 <= x >= 46, the AI folds (14.9%)
- When 47 <= x >= 82, the AI calls (35.6%)
- When 83 <= x >= 100, the AI raises (17.8%)

Hand strength 13

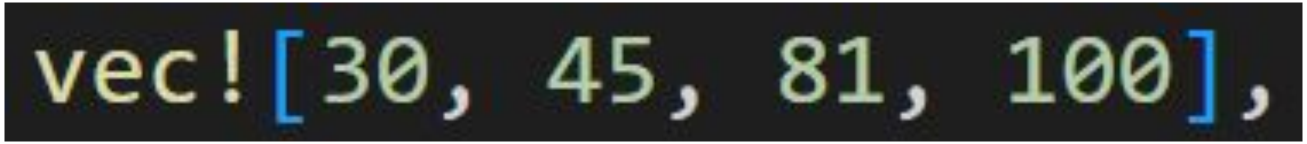
- When 0 <= x >= 30, the AI checks (30.7%)
- When 31 <= x >= 45, the AI folds (14.9%)
- When 46 <= x >= 81, the AI calls (35.6%)
- When 82 <= x >= 100, the AI raises (18.8%)

12:



```
vec![31, 46, 82, 100],
```

13:



```
vec![30, 45, 81, 100],
```

Background: Game Theory

What is Game Theory

- Mathematical way of modeling games among rational players
- The words “game” and “players” in this context are not used as in everyday speech
- In our case, players are assumed to be rational and competitive with each other
  - “Non-cooperative”
- The goal of a player is to maximize their expected utility, or play with a best response strategy
  - These terms will be explained later.
- You can think of “utility” in general as some quantification of “reward” or “payoff” for taking a certain action.

Normal form games

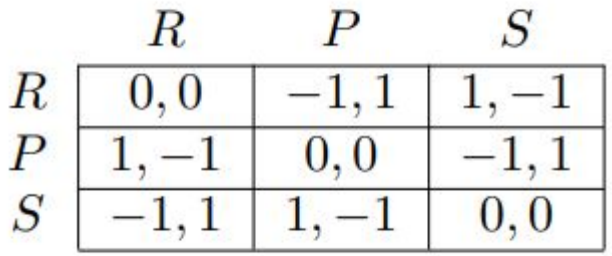
A normal form game is a 3-tuple (N, A, u) where:

- $N = \{1, \dots, n\}$ where $n \in \mathbb{Z}^+$ denotes the set of players.
- S_i is a **finite** set of moves for the ith player.
- $A := S_1 \times \cdots \times S_n$ is the set of action profiles.
    - Action profile: a combination of simultaneous player moves.
- $V \subset \mathbb{R}^n$ denotes the set of all possible utility vectors for the n players.
- u is a map $u : A \to V$ defined by $A \ni a \mapsto (u_1, \dots, u_n) \in V$.

  So u_i denotes the utility for player i if the move combination a is performed by the players.

Normal form games (cont.)

- The map u can be represented as a (discrete) vector field embedded in n dimensional affine space, which in the n=2 case can be displayed as a table with headers in A and values in V.
- u is a fixed map. This represents that normal form games are single turn.
- Example: RPS



	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Image: [1]

Expected utility

We want a way to evaluate how good a player's strategy is. In this case, "strategy" refers to how the players choose their moves. For mixed strategies, the probability that player i selects move $s \in S_i$ is denoted by $\sigma_i(s)$. Player i's opponents are denoted by $-i$. The expected utility of the strategy σ_i for a two-player game is given by

$$\sum_{s \in S_i} \sum_{s' \in S_{-i}} \sigma_i(s)\sigma_{-i}(s')u_i(s, s')$$

Expected utility (cont.)

Here is the formula in full generality:

$$\sum_{a \in A} u_i(a)\sigma_i(a_i) \prod_{j \neq i} \sigma_j(a_j)$$

Key takeaway: the limit of the running average of player i's utility converges the expected utility as the number of rounds approaches infinity.

Equilibrium

Equilibrium is reached when no player has an incentive to switch their strategy.

If you assume all players are perfectly rational, the optimal strategy for each player will be an equilibrium strategy.

In terms of what we just discussed, this occurs when all players are using a best response strategy.

Background for Hard AI: Regret Matching

Regret matching

Regret matching is an algorithm for maximizing expected utility in simultaneous games.

Example: RPS.

Poker is **not** a simultaneous game, but regret matching serves as the basis for the algorithm behind our Hard AI.

“Regret” measures what would’ve been the best move in retrospect. The regret for doing a move s instead of s’ is the difference of their utilities (given identical opponent moves). We seek to minimize regret over time.

Regret matching (cont.)

1. For each player, initialize two 0-vectors of dimension equal to the number of possible moves (to store cumulative regret and strategy respectively).
2. “Train” for some number of iterations N:
    a. **Regret match**: Compute and normalize the vector v containing max(0, cumulative regret for not playing move s) for each move s. If this results in the zero vector, use a uniform strategy.
    b. Add v to each player’s cumulative strategy.
    c. Select a move using v for each player.
    d. Compute regrets and add it to the cumulative regret vector.
3. The final strategy is the cumulative strategy divided by N (the average strategy used across training).

Our Hard AI

What algorithm is the Hard AI using?

- Why can't we just use Regret Matching by itself?
  - Not meant for sequential games
  - Based on knowing exact utilities for other moves

- Counterfactual Regret (CFR) Minimization algorithm
  - An application of Regret Matching for sequential incomplete information games
  - Counterfactual - relating to or expressing what has not happened or is not the case.
  - At every decision point the regrets for other options not taken are considered

Hard AI Overview

- Goals
  - To create an AI that is more difficult and made for more seasoned players.
  - To create an AI using the CFR to make a much more realistic decisions.
- Implementation
  - Uses the CFR algorithm
  - Similar to the Easy AI still probabilities for what action is taken
  - All hands have their own CFR data to make decisions more realistic.

Step 1: Initialize CFR Data

- Each of the 5 card hand strengths have their own CFR data
- CFR data includes two main parts:
  - Strategy table: Maps actions to the probability for that action
  - Regret table: Maps actions to the total regret for that action
- Strategy table is initialized with ¼ for all actions

Fold	Raise	Check	Call
.25	.25	.25	.25

- Regret table is initialized to 0 for all actions

Fold	Raise	Check	Call
0	0	0	0

Step 2: Pick Initial Action and identify results

- Using the Strategy Table, randomly generate a number and select which action to take
- After the action is taken determine the utility for all actions. I.e. You folded and that had a utility of 0, raising could've had a utility of 50, etc.
  - How do we determine these values since we don't have complete information?
    - We'll get to that later

Step 3: Update regrets and strategies

- Finally take the new utility values and update the regrets and strategies.
    - Fold would be a baseline utility of 0
    - Checking is 20 because it would've at least kept the AI in the hand
    - Calling also 20 for similar reasons
    - Raising was 50 due to maybe getting the other player to quit early or better hand
- Subtract from baseline utility and add values to regret table (only positive values)
- Finally renormalize all probabilities for strategy table using regret values
- Steps 2-3 can be repeated infinitely

Example

Strategy Table for High Card

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for High Card

Fold	Raise	Check	Call
0	0	0	0

Before any actions are taken

Hand:
Ace of Spades, Queen of Hearts

Hand Strength:
High Card

Action Taken:
None

Example (Cont.)

Strategy Table for High Card

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for High Card

Fold	Raise	Check	Call
0	0	0	0

Hard AI has raised bet by 50!

Hand:
Ace of Spades, Queen of Hearts

Hand Strength:
High Card

Action Taken:
Raise 50

Example (Cont.)

Determined Utilities:

Raise 50: 50

Fold: 0

Call: 20

Check: 20

Hand:
Ace of Spades, Queen of Hearts

Hand Strength:
High Card

Action Taken:
Raise 50

Hard AI has raised bet by 50!

Example (Cont.) before Strategy table update

Strategy Table for High Card

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for High Card

Fold	Raise	Check	Call
0	0	0	0

Hard AI has raised bet by 50!

Hand:
Ace of Spades, Queen of Hearts

Hand Strength:
High Card

Action Taken:
Raise 50

Example (Cont.) before Community Cards dealt

Strategy Table for High Card

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for High Card

Fold	Raise	Check	Call
0	0	0	0

Now dealing flop

Hand:
Ace of Spades, Queen of Hearts

Hand Strength:
High Card

Action Taken:
None Yet

Example (Cont.) after flop

Strategy Table for Two Pair

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for Two Pair

Fold	Raise	Check	Call
0	0	0	0

Waiting for Hard AI action

Hand:
Ace of Spades, Queen of Hearts

Flop: Ace of Hearts, Queen of Clubs, 7 of spades

Hand Strength:
Two pair

Action Taken:
None Yet

Example (Cont.) after Hard AI Action

Strategy Table for Two Pair

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for Two Pair

Fold	Raise	Check	Call
0	0	0	0

Hand:
Ace of Spades, Queen of Hearts

Flop: Ace of Hearts, Queen of Clubs, 7 of spades

Hand Strength:
Two pair

Action Taken:
Fold

Example (Cont.)

Determined Utilities:

Raise: 100

Fold: 0

Call: 30

Check: 30

Hard AI has folded!

Hand:
Ace of Spades, Queen of Hearts

Flop: Ace of Hearts, Queen of Clubs, 7 of spades

Hand Strength:
Two Pair

Action Taken:
Fold

Example (Cont.) before Strategy table update

Strategy Table for Two Pair

Fold	Raise	Check	Call
.25	.25	.25	.25

Regret Table for Two Pair

Fold	Raise	Check	Call
0	100	30	30

Hand:
Ace of Spades, Queen of Hearts

Flop: Ace of Hearts, Queen of Clubs, 7 of spades

Hand Strength:
Two pair

Action Taken:
Fold

Example (Cont.) before river

Strategy Table for Two Pair

Fold	Raise	Check	Call
0	.625	.1875	.1875

Regret Table for Two Pair

Fold	Raise	Check	Call
0	100	30	30

How do we determine these utilities since we don’t have complete information?

Hand:
Ace of Spades, Queen of Hearts

Flop: Ace of Hearts, Queen of Clubs, 7 of spades

Hand Strength:
Two pair

Action Taken:
Fold

Determining Utility

- As you now know, utility is the measure of player preference for different outcomes.
  - In Poker, our utility would normally be correlated with how much money was won or lost
- Difficult to measure with imperfect information.
  - We measure our utility after each round so we don’t know what we are truly gaining/losing
- How we calculate utility:
  - In preflop, take our base likelihood of winning
    - 1/Number of players
  - In later rounds we update the likelihood from previous rounds.
  - For each opponent move, add or subtract from that likelihood based on:
    - Opponent actions
    - Round
    - Own hand strength
  - The net change in our likelihood of winning is the utility of our action that round

Measuring Opponent Actions

- How do we know what raises/lowers our chances?
  - We don’t
  - Need to make educated guesses
    - Many caveats
- The Process
  - An opponent folding will always increase win likelihood
  - An opponent raising is more likely to lower our likelihood the weaker our hand is
    - An opponent repeatedly raising is very likely to lower our odds
  - An opponent checking increases win likelihood
  - An opponent calling can change the win likelihood in either direction depending on other factors.

Implementation Challenges

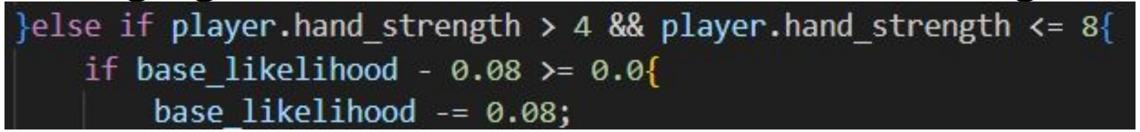
- How do we measure utility like a real player?
  - Consider many factors as possible:
    - Game Phase
    - Opponent Actions
    - Hand Strength
- How do we appropriately balance these factors?
  - A work in progress
  - Small changes can yield very different behaviors
- How would our win likelihood be different if we chose another action?
  - We don’t know how opponents would react in different scenarios
  - Instead look at change in win likelihood if we act differently but all else remains the same.

Implementation Challenges (cont.)

- How do our actions change win likelihood?
  - If we raise:
    - And our opponent raises after
      - Decrease win likelihood by 10%
  - If we call:
    - And our opponent raises after
      - Decrease win likelihood by 7%
  - Logic is that our opponent is very confident if they raise after we already did
  - Similar logic is present throughout our utility function
    - Every combination of actions yields different outcome
    - This way we can discern which choice is best

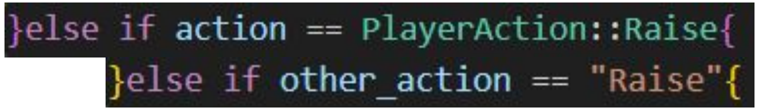
Peek Under the Hood

- Changing win likelihood based on hand strength:



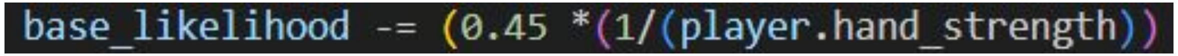
```
}else if player.hand_strength > 4 && player.hand_strength <= 8{
    if base_likelihood - 0.08 >= 0.0{
        base_likelihood -= 0.08;
```

- Check our own action, then opponent action(s):



```
}else if action == PlayerAction::Raise{
      }else if other_action == "Raise"{
```

- Change win likelihood based on prior factors + hand strength



```
base_likelihood -= (0.45 *(1/(player.hand_strength))
```

- This is the core logic throughout the utility function

Works Cited

[1] http://modelai.gettysburg.edu/2013/cfr/cfr.pdf

[2] https://www.nytimes.com/2022/01/18/magazine/ai-technology-poker.html

[3] https://www.researchgate.net/figure/Game-tree-for-one-round-of-Rock-Paper-and-Scisors-In-the-first-column-are-the_fig1_309204883

[4] https://www.pokernews.com/news/2017/10/artificial-intelligence-poker-history-implications-29117.htm