Game Playing with AI

Deterministic vs. Stochastic Games, Partially Observable Games, Deep Learning

Lecture by Margrit Betke

Chess

16 pieces per player:
1 king
1 queen
2 rooks
2 bishops
2 knights
8 pawns

8 x 8 board

Pawn, rook, knight, bishop, queen, king

Abstract strategy two-player game

IBM Deep Blue static evaluator:

Pawn: 1

Knight: 3, Bishop: 3.25

Rook: 5

Queen: 9

Piece count

King safety

Image Credits: Wikipedia: Bubba73, F. Camaratta

Deterministic Games

- Chess
- Go

Can be represented by a “Deterministic Game Tree”

Minimax Procedure uses “deterministic static evaluation scores”

Stochastic Games

Combine luck & skill

Luck?

Probability of outcome when throwing one die or several dice

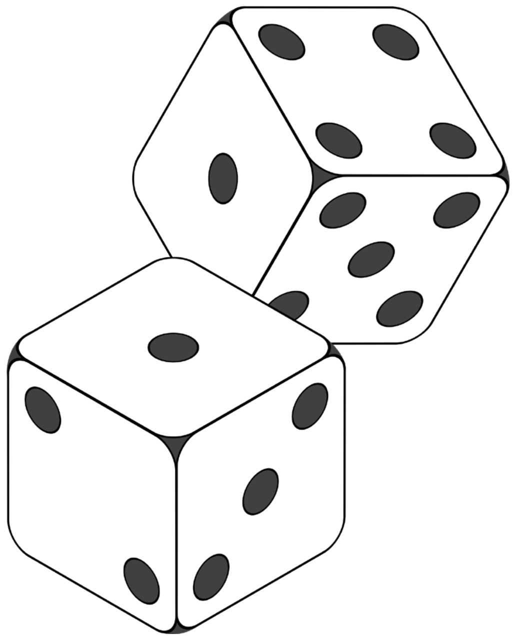


Image credit: S. Green Wikipedia

Stochastic Games

Combine luck & skill

Game tree includes	chance nodes	●
	maximizer nodes	▲
	minimizer nodes	▼

Minimax Procedure uses “expected static evaluation scores”

Chance Nodes

Example: 2 dice are used

		Die 1					
		1	2	3	4	5	6
	1						
	2						
Die 2	3						
	4						
	5						
	6						

6x6 = 36 ways to roll two dice

Chance Nodes

6 doubles: Probability = 1/36

		Die 1					
		1	2	3	4	5	6
Die 2	1	(1,1)					
	2		(2,2)				
	3			(3,3)			
	4				(4,4)		
	5					(5,5)	
	6						(6,6)

Image credit: Battle Forge Games

Chance Nodes

Non-doubles: Probability = 2/36 = 1/18

(2,4) means the same as (4,2)

Die 1

Die 2		1	2	3	4	5	6
	1	(1,1)					
	2		(2,2)		(2,4)		
	3			(3,3)			
	4		(4,2)		(4,4)		
	5					(5,5)	
	6						(6,6)

Image credit: Dice World

Chance Nodes

How many non-doubles?
Upper triangle

		Die 1					
		1	2	3	4	5	6
	1	(1,1)	(1,2)	(1,3)	…		
	2		(2,2)			…	
Die 2	3			(3,3)		…	
	4				(4,4)	(4,5)	(4,6)
	5					(5,5)	(5,6)
	6						(6,6)

Chance Nodes

How many non-doubles?
5+4+3+2+1 = 15 rolls

		Die 1					
		1	2	3	4	5	6
	1	(1,1)	(1,2)	(1,3)	…		
	2		(2,2)			…	
Die 2	3			(3,3)		…	
	4				(4,4)	(4,5)	(4,6)
	5					(5,5)	(5,6)
	6						(6,6)

Chance Nodes

Total number of distinct rolls:
6 doubles + 15 non-doubles

		Die 1					
		1	2	3	4	5	6
	1	(1,1)	(1,2)	(1,3)	…		
	2		(2,2)			…	
Die 2	3			(3,3)		…	
	4				(4,4)	(4,5)	(4,6)
	5					(5,5)	(5,6)
	6						(6,6)

Chance Nodes

Probability of event space (= all rolls) must be one.
6 doubles + 15 non-doubles: Prob(all) = 1/36 *6 + 1/18 *15 = 1

		Die 1					
		1	2	3	4	5	6
	1	(1,1)	(1,2)	(1,3)	…		
	2		(2,2)			…	
Die 2	3			(3,3)		…	
	4				(4,4)	(4,5)	(4,6)
	5					(5,5)	(5,6)
	6						(6,6)

Backgammon -- Setup

Two player board game. The board is grouped into four quadrants of six triangles each.

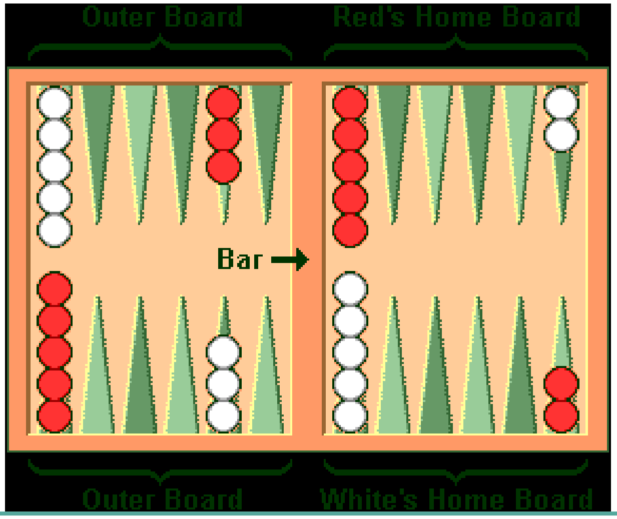
The points are numbered for either player starting in that player's home board.
The outermost point is the twenty-four point, which is also the opponent's one point.

Each player has fifteen checkers of his own color.
The initial arrangement of checkers is:
two on each player's twenty-four point, five on each player's thirteen point, three on each player's eight point, and five on each player's six point.

Both players have their own pair of dice and a dice cup used for shaking.

Red home board

Outer boards



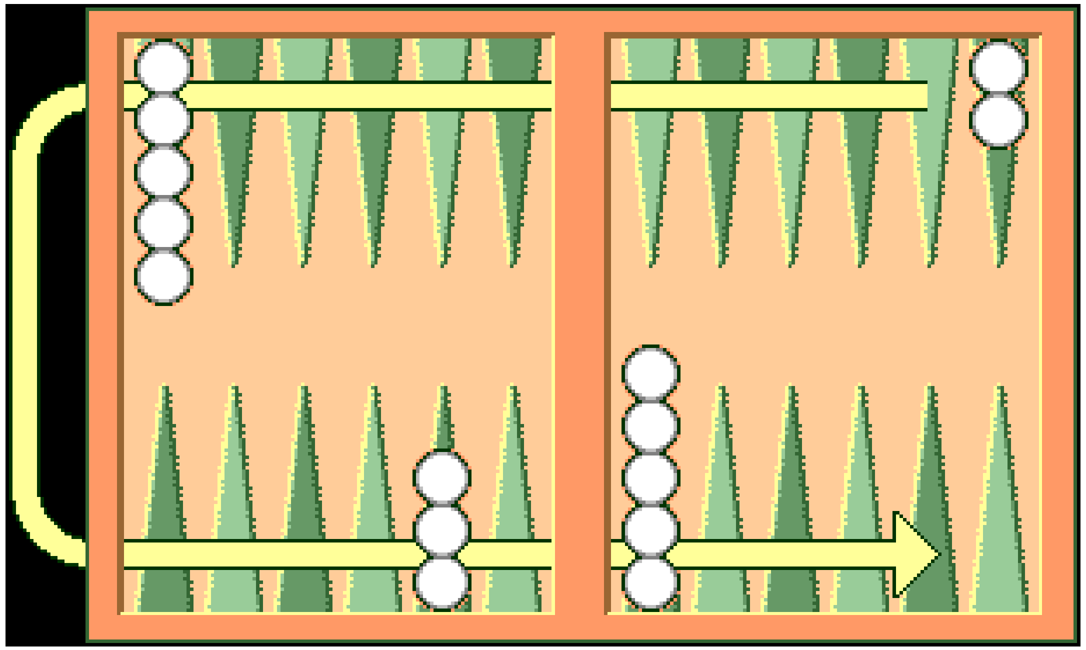
White home board

Illustrations from GNU manual

Backgammon – Goal of the Game

The goal of the game is for a player to move all of their checkers into their own home board and then bear them off.

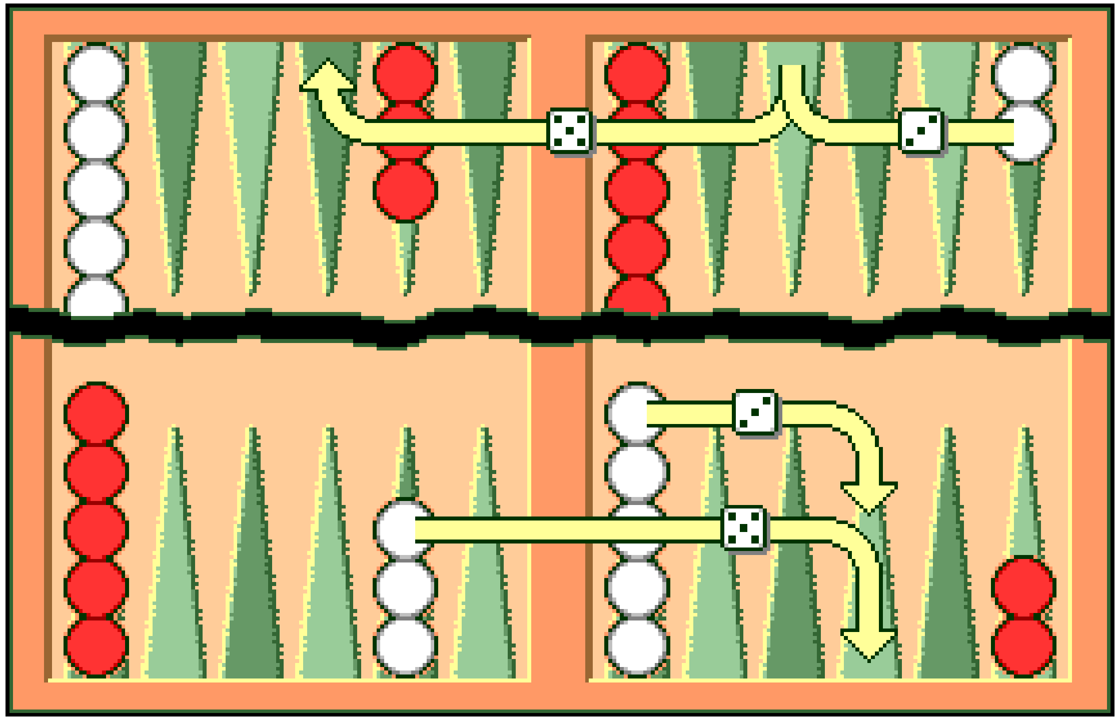
The first player to bear off all their checkers wins the game.



Direction of movement of White’s checkers. Red’s checkers move in the opposite direction

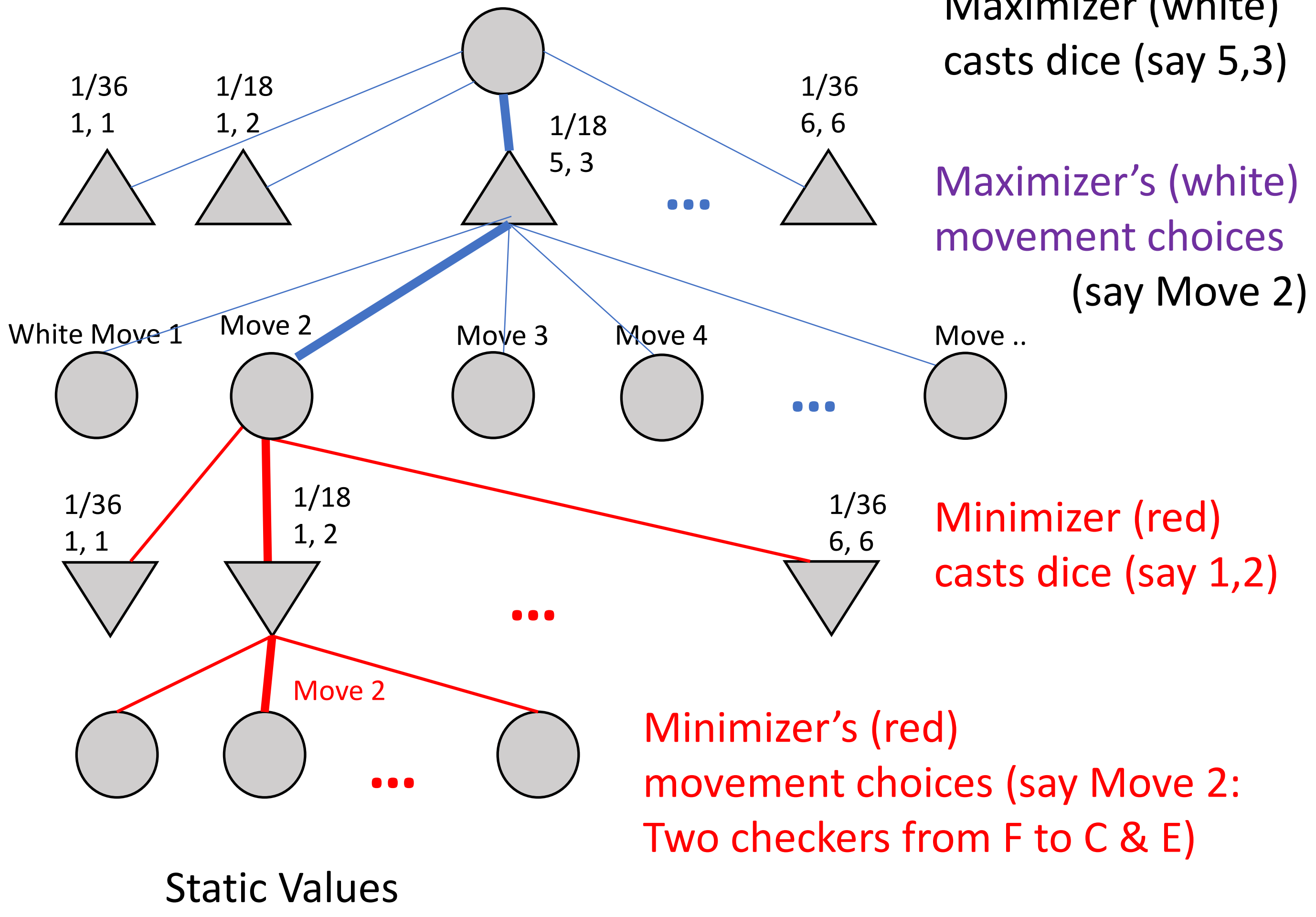
Backgammon Movements

- Start: The player throwing the higher number moves his/her checkers according to the numbers showing on both dice.
- A checker may be moved only to a triangle that is unoccupied or is occupied by one or more of the player's own checkers (or that is occupied by exactly one opposing checker).
- The numbers on the two dice constitute separate moves.

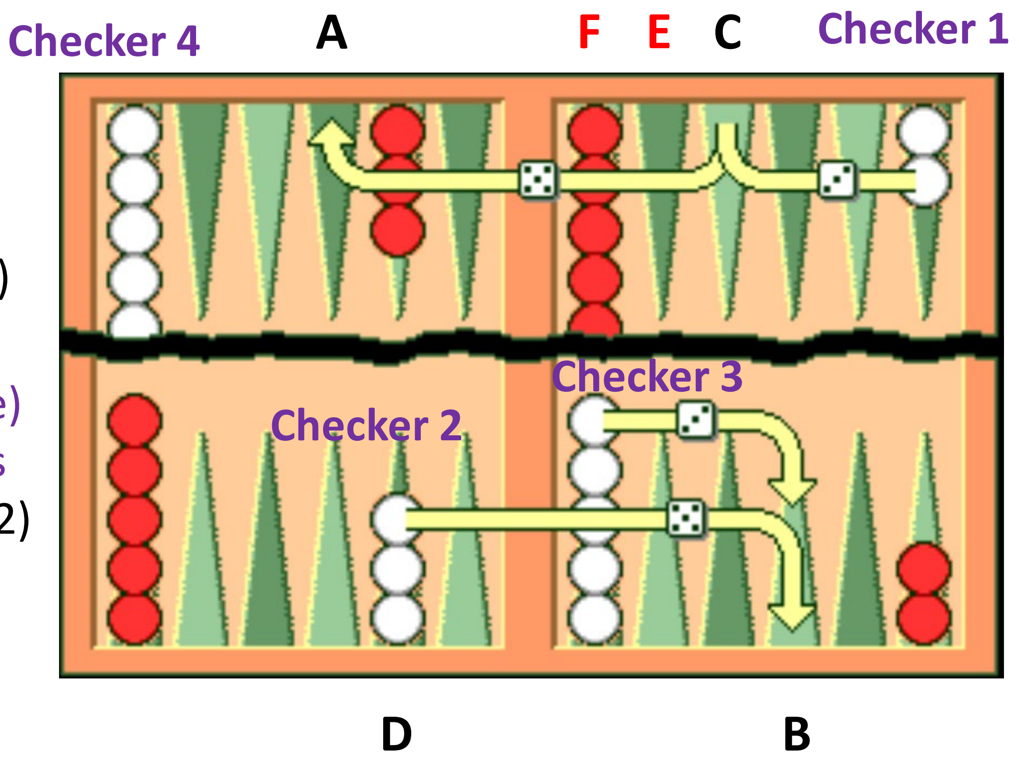


White opens the game with 5,3:
Two (of many) movement choices

Backgammon -- Game Tree



1/36
1, 1
1/18
1, 2
1/18
5, 3
1/36
6, 6
...
White Move 1
Move 2
Move 3
Move 4
...
Move ..
1/36
1, 1
1/18
1, 2
1/36
6, 6
...
Move 2
...
Static Values
Maximizer (white) casts dice (say 5,3)
Maximizer's (white) movement choices (say Move 2)
Minimizer (red) casts dice (say 1,2)
Minimizer's (red) movement choices (say Move 2: Two checkers from F to C & E)



White's Movement Choices:

Move 1: Move checker 1 to **A**

Move 2: Move checker 2 to **B**, and checker 3 to **B**

Move 3: Move checker 1 to **C**, and checker 3 to **B**

Move 4: Move checker 1 to **C**, and checker 4 to **D**

Move 5: …

Game Playing Algorithm for Stochastic Games

Minimax score for deterministic games

-> Expected minimax score(node) for chance games

= static value(node) if node at horizon

$= \max_{\text{actions}}$ {expected minimax score(node, action)} if maximizer choice at node

$= \min_{\text{actions}}$ {expected minimax score(node, action)} if minimizer choice at node

$= \sum_{r}$ Prob(r)·expected minimx score(node, r) if node is chance node

r represents possible dice roll (= chance event), Prob(r) = likelihood of event

Game Playing Algorithm for Stochastic Games

b = branching factor, n = # possible dice-roll outcomes, m = depth of tree:

Expected minimax score (node) for chance games $O(b^m n^m)$

$$
\begin{array}{ll}
= \text{static value(node)} & \text{if node at horizon} \\
= \max\limits_{\text{actions}} \{\text{expected minimax score(node, action)}\} & \text{if maximizer choice} \\
= \min\limits_{\text{actions}} \{\text{expected minimax score(node, action)}\} & \text{if minimizer choice} \\
= \sum\limits_{r} \text{Prob}(r) \cdot \text{expected minimx score(node, r)} & \text{if node is chance node}
\end{array}
$$

Game Playing Algorithm for Stochastic Games

- b = branching factor, n = # possible dice-roll outcomes, m = depth of tree: $O(b^m n^m)$
- Game tree size explodes quickly => Small look-ahead depth used
- Can use alpha-beta pruning if lower & upper bounds on scores known
- Or: Use "Roll out" technique = Monte Carlo Simulation

Analysis Technique “Roll out” = “Monte Carlo Simulation”

Produce static scores of possible board positions by preprocessing =

“Monte Carlo Simulation”

by playing thousands of games against itself

Resulting win percentage = Good approximation of value of a board position (static score)

TD-GAMMON

- Gerry Tesauro, IBM, created TD-GAMMON (1992)
- Competitive with top human players
- Look-ahead depth of 2 or 3
- Static evaluator used neural nets
- 1st “real” application of Reinforcement Learning
- Learning: > 1 million training games
- TD = “temporal difference learning” = Monte Carlo sampling & pruning

GNU Backgammon

More details on rules and open source code for GNU backgammon at https://www.gnu.org/software/gnubg/manual/gnubg.pdf

GNU Backgammon is a world class opponent and rates at around 2100 on FIBS, the First Internet Backgammon Server (at its best, it is in the top 5 of over 6000 rated players there).

GNU Backgammon can be played on numerous other on-line backgammon servers.

Famous recent AI Game Playing Events

- Go
- StarCraft

Go

- Deterministic game but highly challenging for AI because of large branching factor
- 19x19 board
- Monte Carlo rollouts used
- In 2012: AI systems could only play at master level on reduced 9x9 board
- In 2016: Google's AlphaGo
  - Defeated Korean grandmaster Lee Sedol
  - 4 wins, one loss, drama in game 4
  - Last game: AlphaGo made an early mistake (because human made an unlikely genius move), had to dig out of its hole, game balanced on knife's edge

Image credit: Lee Jin-man, AP

Chess versus Go

- Board:

  Chess: 8 x 8 (64 squares), Go: 19x19 (361 intersection points)

- Number of first moves:
  - Chess: 20 white moves x 20 black moves = 400 possible first moves
  - Go: 361 white x 360 black = 129,960 (due to symmetry, 32,490 first moves)
- Number of possible games (i.e., leaf nodes in game tree):
  - Chess: 10^{120}
  - Go: 10^{174}
  - Number of atoms in the universe: 10^{80}

https://herculeschess.com/how-many-chess-games-are-possible/
https://www.quora.com/How-does-the-complexity-of-Go-compare-with-Chess

StarCraft

- Stochastic game with large branching factor, many local optima in search space
- Action space: ~10^{26}
- Moves per game: 1,000’s
- In 2015: Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE) 3 AI bots beaten by human
- In 2017: Sejong University competition: All AI bots beaten by humans
- In 2019: AlphaStar Demonstration (by DeepMind Technologies, owned by Alphabet/Google):   Won 10 games against humans, lost one. Uses “Deep Reinforcement Learning” and “Imitation Learning”

Partially Observable Games

Deterministic

Lack of access to choices made by opponent

Example: Battleships: Players can only see their own pieces
http://en.battleship-game.org

Stochastic

Missing information is generated randomly

Examples: Card games: bridge, whist, hearts, various forms of pokers

Card Games

Algorithm

- Consider all possible deals of the invisible cards
- Solve each as if it were a fully observable game
- Choose the move that has the best outcome averaged over all the deals

Card Games

Algorithm

- Consider all possible deals of the invisible cards
- Solve each as if it were a fully observable game
- Choose the move that has the best outcome averaged over all the deals
- P(s) = Probability that deal s occurs
- Next move = $\underset{\text{actions}}{\operatorname{argmax}} \left\{ \sum_{s} P(s)\ \text{MinimaxScore}(s, \text{action}) \right\}$

Card Games

But: Often large number of deals

- E.g., Bridge: 2 hands, 13 cards each -> 10,400,600 deals
- Use Monte Carlo = repeated random sampling
- N samples of $s_1, \ldots, s_N$ deals:

$$\text{Next move} = \underset{\text{actions}}{\text{argmax}} \left\{ 1/N \sum_{s_i = 1}^{N} P(s_i)\ \text{MinimaxScore}(s_i, \text{action}) \right\}$$

Solving Poker: Counterfactual Regret (CFR) Minimization

- Information sets are sets of states among which the acting player cannot distinguish, i.e., states where the opponent was dealt different private cards
- When every player is playing with a best response strategy to each of the other player's strategies, the combination of strategies is called a Nash equilibrium. No player can expect to improve a play by changing strategy unilaterally.
- Regret is the loss in utility an algorithm suffers for not having selected the single best deterministic strategy, which can only be known in hindsight (thus counterfactual).
- A regret-minimizing algorithm is one that guarantees that its regret grows sublinearly over time, and so eventually achieves the same utility as the best deterministic strategy.
- The key insight of CFR is that instead of storing and minimizing regret for the exponential number of deterministic strategies, CFR stores and minimizes a modified regret for each information set and subsequent action, which can be used to form an upper bound on the regret for any deterministic strategy.
- An approximate Nash equilibrium is retrieved by averaging each player's strategies over all of the iterations, and the approximation improves as the number of iterations increases.
- *Deep CFR* uses deep neural networks to approximate the behavior of CFR: Brown et al., ICML 2019

Learning Outcomes

Being able to

- explain the difference between a deterministic, stochastic, and partially observable game
- sketch or traverse a sample the game tree for the three types of games
- know how to compute an expected minimax score
- define Monte-Carlo rollouts
- explain Counterfactual regret minimization (at a high level)
- discuss some of the history of AI tackling Backgammon, Go, StarCraft, and Poker