Markov Decision Process

Why do we care about MDPs

Markov Decision Processes are the defining problem of RL and any way to solve them is a method of RL

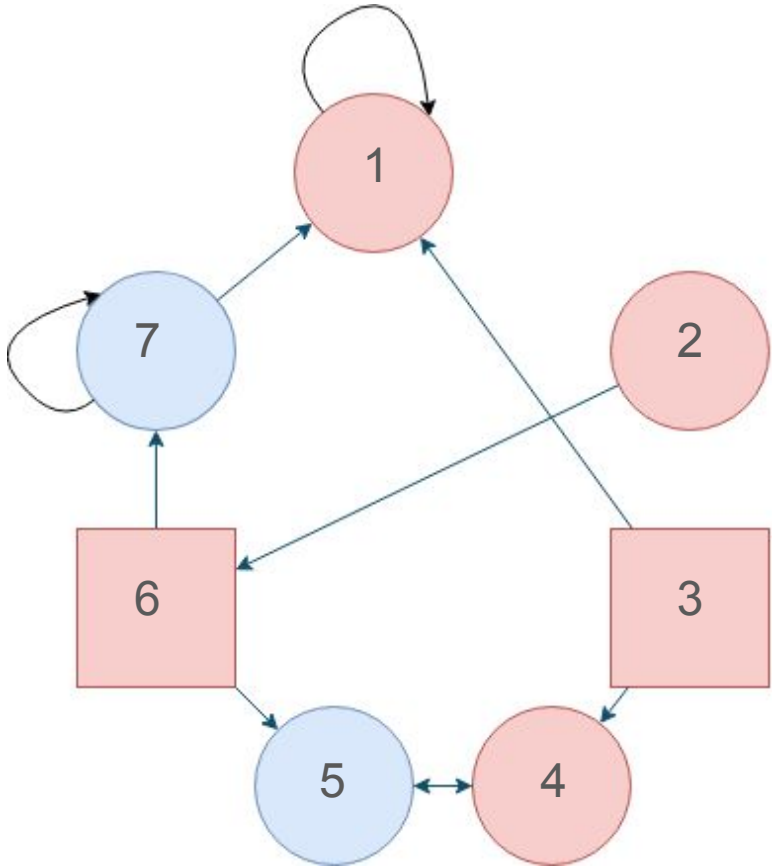
A Simple Zero-Sum Game

Players: Adam (Square) and Eve (Circle)

Objective for Eve:

Move a token to a blue vertex

Question - Given a starting vertex, does Eve have a winning strategy?



In General, we define a Game on a Graph as:

- $G = (V, E)$
- $V = V_{Adam} \uplus V_{Eve}$
- A Colouring Function: $E \rightarrow C$
- A Win Condition (Qualitative): $V_{win} \subset V$

  In this case, our 2 players are Adam and Eve. Eve wins if there exists a single strategy such that no matter what Adam does, the token reaches $v \in V_{win}$

- Objective: Reachability, Safety, Buchi, co-Buchi, etc.

In General, we define a Game on a Graph as:

- A Win Condition (Quantitative): f: Paths $\rightarrow R \cup \{\pm\infty\}$

  In this case, our 2 players are Max and Min. If the cumulative total is below a certain threshold ε Min wins, else Max wins

- Alternative Formulation: Optimization!

  We can think of these as rewards

MDP as Game of Agent vs Environment

The previous game is an example of a 2-player game: Adam and Eve.

Reinforcement Learning → agent vs the environment.

Games on Graph → 1.5 player game

Example of such a game: Blackjack

Simplified Blackjack Rules

Deck & Card Values:

- **Cards:** {1,2,3,…,10,11,12,13}
- **Draw:** With replacement (Pr(c)=1/13)
- Both player and dealer get their cards face up. The dealer starts with 1 card.

Player's Turn:

- **Hit:** Draw one card
- **Stand:** End turn; player's score is the sum total of the cards that they have drawn

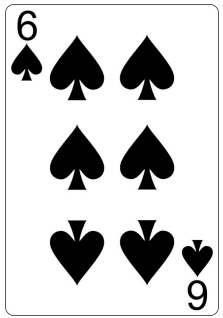
Dealer's Turn:

- Hits until d≥17
- d>21: dealer busts → **player wins**.

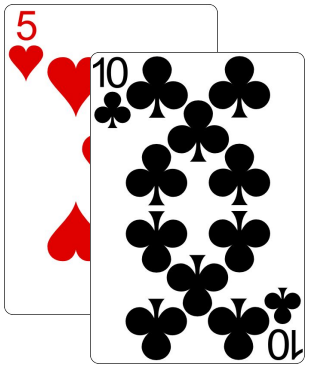
Outcome (if neither busts):

- p>d: **win**
- p<=d: **loss**

Dealer

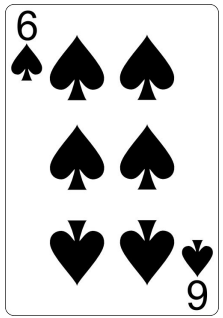


Player

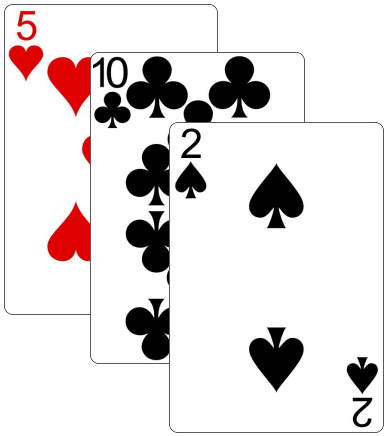


Hit or Stand?

Dealer

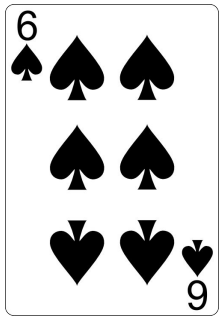


Player

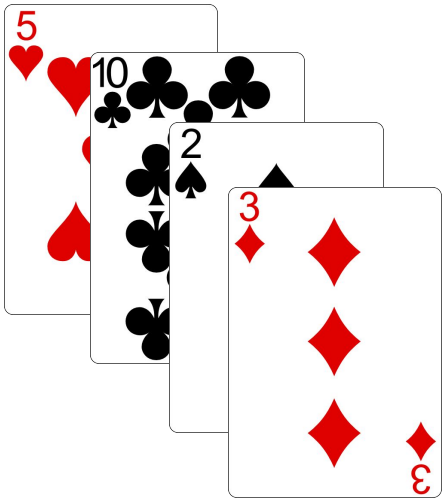


Hit or Stand?

Dealer

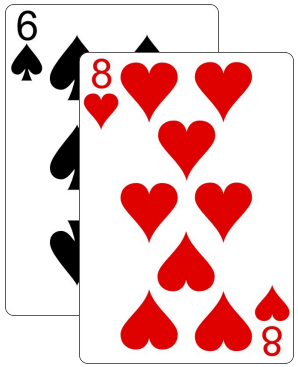


Player

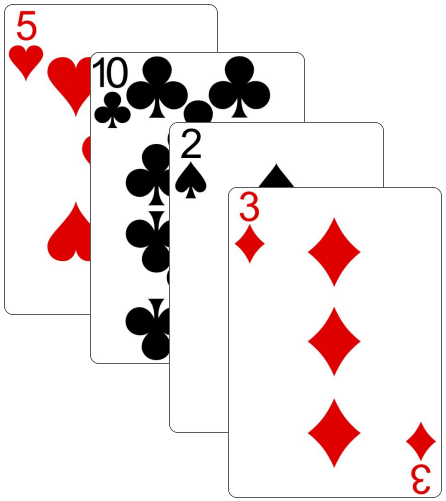


Players Total: 20

Dealer

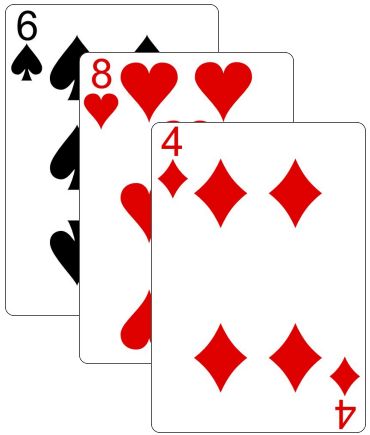


Player

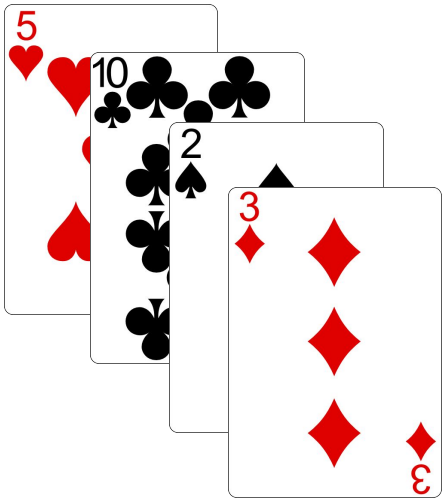


Players Total: 20

Dealer

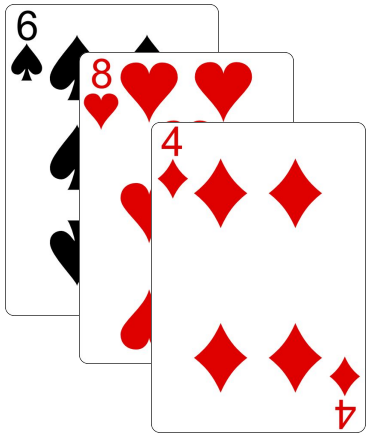


Player



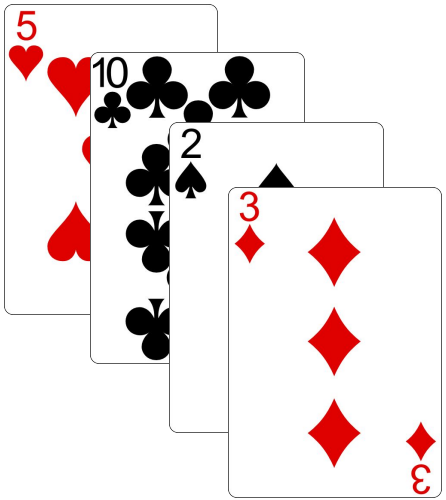
Players Total: 20

Dealer



Dealer's Total: 18

Player



Players Total: 20

Player Wins!

Modelling Blackjack as a MDP

S = {(p,d): p is the player's score, d is the dealer's score} ∪ {Win, Lose}

Action Space = {Hit, Stand}

Goal

Win: 1-sink,

Lose: 0-sink

Transition Function

Hit:

(p+x,d) where P(x = {1,2,...,13}) = 1/13

Lose if p+x>21

Stand:

Player Stops

Dealer draws until d≥17

Game Formulation of an MDP - Markov Chain

Graph - $\{V_{avg}, E\}$

V_{avg} - $\{1,2,...,n-1,n\}$

All vertices have two outgoing edges with probability ½

v_{n-1} : 0-sink, v_n: 1-sink | Both edges are self loops

Transition matrix: NxN with all entries as 0,½, or 1

Said to be stopping if from every vertex you can reach a sink

Reachability of Markov Chains

We wish to reach the 1-sink and avoid the 0-sink

$v_n = 1.$ |. $v_{n-1} = 0.$ | $v_i = ½\ v_j + ½\ v_k$

Let

$\boldsymbol{v}$ = {1,2,...n-2} : $\boldsymbol{v}$ is a column vector

Q = (n-2) x (n-2) transition matrix without 0-sink or 1-sink

b = (n-2) x 1 vector of transitioning to 1-sink from the current vertex

$\boldsymbol{v} = Q\boldsymbol{v} + b$

Properties of Markov Chains

Properties that are useful to solving Markov Chains

- $\lim_{n\to\infty} Q^n = 0$ for a stopping matrix
- $(I - Q)^{-1} = 1 + Q + Q^2 + \ldots$
- Unique solution given by $\boldsymbol{\nu} = (I - Q)^{-1}b$

Game Formulation of Markov Decision Processes

$G = \{V_{avg} \cup V_{max}, E\}$

$V_{avg} \cup V_{max} = \{1,2,...,n-1,n\}$

All vertices have two outgoing edges

v_{n-1} : 0-sink (lose), v_n: 1-sink (win) (both edges are self loops)

V_{avg}: Both vertices have probability ½ (simplification)

V_{max}: Certain edges to $V_{avg} \cup V_{max}$, but no probabilities

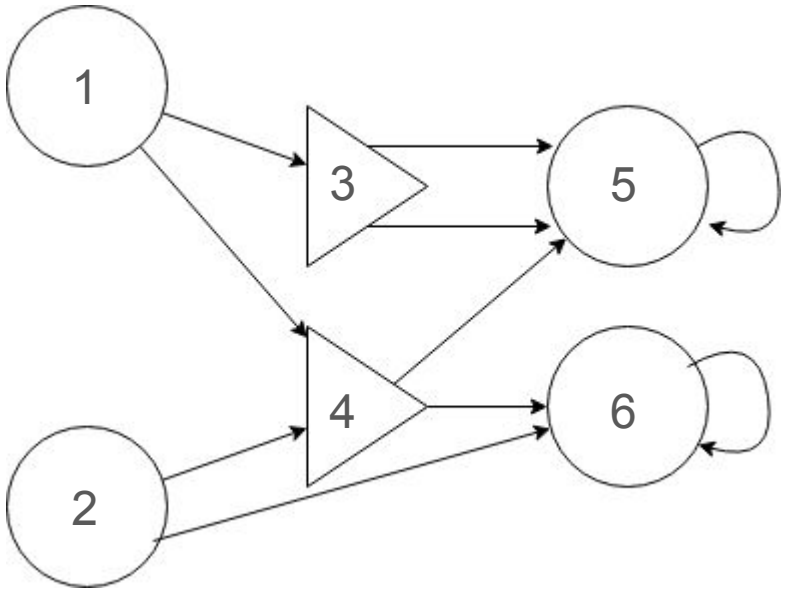
What is a strategy

Vertices: V_{max}(Circles) and V_{max} (Triangles)

Objective for Eve:

Avoid v_5, Reach v_6

Question - What should be the agent's strategy from v_1 and v_2.



Games on Graphs Terminology

Sequential: Players and the environment alternate turns while making decisions.

Fully Observable: The current state of the system is always known with certainty.

Stochastic: Outcomes of actions involve randomness, meaning the same action taken in the same state can lead to different results.

Markovian (RL Term): The decision-making process depends only on the current state and not on the sequence of past states.

Positionally Determined (Game Theory Term): The optimal strategy from a given state does not depend on the history of previous states.

A Comparison of Terminology

Games	**Reinforcement Learning**
Players	Agents
Moves	Actions
Strategy	Policy
Quantitative (Winning) Objective	Reward Function
Fog of War	Partial Observability (POMDP)
Game Balancing	Reward Shaping
Cheating	Adversarial Attack

Markov Decision Process to Markov Chain

Each strategy σ for G gives a Markov Chain G_σ. An MDP is said to be stopping if for every σ, the Markov Chain G_σ is stopping. Corresponding to G_σ is the vector denoting reachability probabilities defined in the previous slide

$$\overline{v} := \begin{bmatrix} \max_\sigma v_\sigma(1) \\ \max_\sigma v_\sigma(2) \\ \vdots \\ \max_\sigma v_\sigma(n) \end{bmatrix}$$

$$0 \leq w_i \leq 1 \quad \text{for every } i \tag{1.1}$$

$$w_n = 1$$

$$w_{n-1} = 0$$

$$\text{for each } i \leq n-2 \text{ in } V_{max} \qquad w_i = \max(w_j, w_k) \quad \text{where } j \text{ and } k \text{ are children of } i$$

$$\text{for each } i \leq n-2 \text{ in } V_{avg} \qquad w_i = \frac{1}{2}w_j + \frac{1}{2}w_k \quad \text{where } j \text{ and } k \text{ are children of } i$$

Policy Iteration Algorithm

Algorithm 1.1: Strategy improvement algorithm for MDPs, also known as policy iteration

```
1 algorithm strategy − improvement(G)
2   σ ← an arbitrary positional strategy
3   v_σ ← probabilities to reach 1−sink in G_σ
4   repeat
5       pick a node i ∈ V_max s.t. v_σ(i) < max(v_σ(j), v_σ(k)) where j,k are its children
6       σ'(i) ← arg max{v_σ(j), v_σ(k)}
7       σ ← σ'
8       v_σ ← probabilities to reach 1−sink in G_σ
9   until σ is optimal
```

Successive Approximation Algorithm

Algorithm 1.2: Value iteration algorithm, also known as Successive approximation algorithm

```
1  algorithm value−iteration(G)
2    let ū be the vector assigning 1 to 1−sink and 0 to everything else
3    repeat
4      define ū′ as follows:
5        u′(i) = max(u(j), u(k)), if i is a max node and j,k are its children
6        u′(i) = ½u(j) + ½u(k), if i is an average node and j,k are its children
7        u′(n − 1) = 0
8        u′(n) = 1
9      let u ← u′
10   until ū satisfies (1.1)
```