Math

Derivative Rules

Chain rule: $\frac{d}{dx}f(u) = \frac{d}{du}f(u) \cdot \frac{d}{dx}u(x)$
Sum/difference rule: $(f \pm g)' = f' \pm g'$
Product rule: $(f \cdot g)' = f' \cdot g + f \cdot g'$
Quotient rule: $(\frac{f}{g})' = \frac{f' \cdot g - g' \cdot f}{g^2}$

Common Derivatives

$\boxed{\frac{d}{dx}a^x = a^x \ln(a)}$ $\boxed{\frac{d}{dx}e^x = e^x}$

$\boxed{\frac{d}{dx}\log_a(x) = \frac{1}{x\ln(a)}}$ $\boxed{\frac{d}{dx}\ln(x) = \frac{1}{x}}$

$\boxed{\frac{d}{dz}\sigma(z) = \frac{d}{dz}(1+e^{-z})^{-1} = \sigma(z)(1-\sigma(z))}$

$\boxed{\frac{d}{dx}\tanh(x) = \frac{d}{dx}\frac{e^x - e^{-x}}{e^x + e^{-x}} = 1 - \tanh^2(x)}$

Markov decision processes

Find the maximum value policy by using MDPs that help us cope with randomeness and uncertainty, in order to find our way between an initial state and an end state.

Notations

Definition - the objective of a MDP is to maximize rewards.

- States S: including S_{start}
- Termination state: $\text{IsEnd}(s)$
- $\text{Actions}(s)$
- $\text{Reward}(s, a, s')$
- Transition probabilities $T(s, a, s')$ $(\forall s, a, \sum_{s' \in S} T(s, a, s') \equiv 1)$
- Discount: $0 \leq \gamma \leq 1$ (usually default 1)

Policy - π is a function that maps each state s to an action $a \in \text{Actions}(s)$

Games

Game tree - describes the possibilities of a game and models opponents & randomness. Each node is a decision point for a player; each root-to-leaf path is a possible outcome of the game. Legend: △ - maximizing node, ▽ - minimizing node, and ◯ - chance node

Two-player zero-sum game - Each state is fully observed and such that players take turns; utility of the agent is negative the utility of the opponent (so the sum of the two utilities is zero).

Players: $= \{\text{agent}, \text{opp}\}$

s_{start}: start state

Actions(s): possible actions from state s

Succ(s, a): resulting state if choose action a in state s

IsEnd(s): whether s is an end state

Utility(s): agent's utility for end state s

Player(s): player who controls the state s

Types of policies -

Stochastic policies: $\pi_p(s, a) \in [0, 1]$ probability of player p taking action a in state s.

Deterministic policies: $\pi_p(s) \in \text{Actions}(s)$ action that player p takes in state s. A (special) instance of Stochastic policies.

Game evaluation - analogous to recurrence for policy evaluation in MDPs. $V_{\text{eval}}(s) =$

$$\begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \sum_{a \in A(s)} \pi_{\text{ag}}(s,a) V_{\text{eval}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{ag} \\ \sum_{a \in A(s)} \pi_{\text{op}}(s,a) V_{\text{eval}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{op} \end{cases}$$

As the agent, we want to solve $\pi_{\text{agent}}(s, a)$: the best thing we should do.

Expectimax - $V_{\text{exptmax}}(s)$ is the max expected utility of any agent policy when playing w.r.t. a *fixed and known* π_{opp}. $V_{\text{exptmax}}(s) =$

$$\begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \max_{a \in A(s)} V_{\text{e-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{ag} \\ \sum_{a \in A(s)} \pi_{\text{op}}(s,a) V_{\text{e-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{op} \end{cases}$$

$\Rightarrow \pi_{\text{exptmax}(7)}, \pi_7$ (assuming the fixed opponent policy π_{opp} is π_7, then the best policy computed by expectimax recurrence for agent is denoted as $\pi_{\text{exptmax}(7)}$).

Minimax - Find an optimal agent policy against an adversary by assuming the worst case: the opponent does everything to minimize the agent's utility. $V_{\text{minimax}}(s) =$

$$\begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \max_{a \in A(s)} V_{\text{m-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{ag} \\ \min_{a \in A(s)} V_{\text{m-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{op} \end{cases}$$

$\Rightarrow \pi_{\max}, \pi_{\min}$:
$\pi_{\max}(s) = \arg\max_{a \in A(s)} V_{\text{minimax}}(\text{Suc}(s,a))$
$\pi_{\min}(s) = \arg\min_{a \in A(s)} V_{\text{minimax}}(\text{Suc}(s,a))$

Minimax properties - we can play an agent policy π_{agent} against an opponent policy π_{opp}, which produces an expected utility via game evaluation, denoted as $V(\pi_{\text{agent}}, \pi_{\text{opp}})$

1. if the agent were to change its policy from $\pi_{\max}$ to any π_{agent}, then the agent wouldn't be better off (and in general, worse off).

   $\boxed{\forall \pi_{\text{agent}}, V(\pi_{\max}, \pi_{\min}) \geq V(\pi_{\text{agent}}, \pi_{\min})}$

2. if the opponent were to change its policy from $\pi_{\min}$ to any π_{opp}, then the opponent wouldn't be better off (the value of the game can only increase, which is favorable to the agent).

   $\boxed{\forall \pi_{\text{opp}}, V(\pi_{\max}, \pi_{\min}) \leq V(\pi_{\max}, \pi_{\text{opp}})}$

   From the agent's point of view, this can be interpreted as guarding against the worst case $\Rightarrow$ If $V_{\text{minimax}}(s) = 1$, the agent is guaranteed at least a value of 1 no matter what the opponent does.

3. if the opponent is known to be not adversarial, then the minimax policy might not be optimal for the agent.

   $\boxed{\text{For } \pi_7, V(\pi_{\max}, \pi_7) \leq V(\pi_{\text{exptmax}(7)}, \pi_7)}$

$$\begin{aligned} V(\pi_{\text{exptmax}(7)}, \pi_{\min}) &\leq V(\pi_{\max}, \pi_{\min}) \\ &\leq V(\pi_{\max}, \pi_{\text{opp}}) \\ &\leq V(\pi_{\text{exptmax}(7)}, \pi_7) \end{aligned}$$

Expectiminimax - **Players**: $= \{\text{agent}, \text{opp}, \text{coin}\}$: a third player representing any sort of natural randomness (metaphorically "coin") is introduced which always follows a known stochastic policy. $V_{\text{exptminmax}}(s) =$

$$\begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \max_{a \in A(s)} V_{\text{e-m-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{ag} \\ \min_{a \in A(s)} V_{\text{e-m-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{op} \\ \sum_{a \in A(s)} \pi_{\text{co}(s,a)} V_{\text{e-m-m}}(\text{Suc}(s,a)) & \text{Playr}(s) = \text{co} \end{cases}$$

Speeding up minimax

Depth-limited tree search - Stop at maximum depth $d_{\max}$. Use: at state s, call $V_{\text{minmax}}(s, d_{\max})$. Convention: decrement depth at last player's turn. $V_{\text{minmax}}(s, d) =$

$$\begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \text{Eval}(s) & d = 0 \\ \max_{a \in A(s)} V_{\text{e-m-m}}(\text{Suc}(s,a), d) & \text{Playr}(s) = \text{ag} \\ \min_{a \in A(s)} V_{\text{e-m-m}}(\text{Suc}(s,a), d-1) & \text{Playr}(s) = \text{op} \end{cases}$$

Evaluation function - a domain-specific and possibly very weak estimate of the value $V_{\text{minmax}}(s)$, analogous to $A^\star$'s FutureCost(s) but unlike $A^\star$ no guarantees on the error from approximation.

Depth-limited exhaustive search - $O(b^{2d})$ time. Still not ideal.

Optimal path - path that minimax policies take. Values of all the nodes on path are the same.

Alpha-beta pruning - a domain-general exact method optimizing the minimax algorithm by avoiding the unnecessary exploration of parts of the game tree. To do so, each player keeps track of the best value they can hope for (stored in α for the maximizing player and in β for the minimizing player). At a given step, $\beta < \alpha \Rightarrow$ the optimal path is not going to be in the current branch as the earlier player had a better option at their disposal.

Order matters:

- Worst ordering: $O(b^{2d})$ time
- Best ordering: $O(b^{2 \cdot 0.5d})$ time
- Random ordering: $O(b^{2 \cdot 0.75d})$ time when $b = 2$

In practice, can use Eval(s):

- on a max node, order successorts by decreasing Eval(s')
- on a min node, order successorts by increasing Eval(s')

Temporal difference (TD) learning - picks a piece of experience (s, a, r, s') and updates $\mathbf{w}$. Used when we don't know the transitions / rewards. The value is based on exploration policy.

Evaluation function could be hand-crafted but also learned from data:
Eval$(s) = V(s; \mathbf{w}) = \mathbf{w} \cdot \phi(s)$ (linear).

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left[\underbrace{\hat{V}_\pi(s;\mathbf{w})}_{\text{prediction}} - \underbrace{(r + \gamma \hat{V}_\pi(s';\mathbf{w}))}_{\text{target}} \right] \nabla_{\mathbf{w}} \hat{V}_\pi(s;\mathbf{w})$$

For linear functions: $V(s;\mathbf{w}) = \mathbf{w} \cdot \phi(s)$; $\nabla_{\mathbf{w}} V(s;\mathbf{w}) = \phi(s)$.

$\boxed{\mathbf{w} \leftarrow \mathbf{w} - \eta [\mathbf{w} \cdot \phi(s) - (r + \gamma \mathbf{w} \cdot \phi(s'))] \phi(s)}$

Feature selection: how good my "board" is.

TD learning vs Q-learning - **Q-learning** operates on $\hat{Q}_{\text{opt}}(s, a; \mathbf{w})$, off-policy (value based on estimate of optimal policy), and doesn't need to know MDP transitions $T(s, a, s')$. **TD learning**: operates on $\hat{V}_\pi(s; \mathbf{w})$, on-policy (value is based on exploration policy), and **needs to know rules of the game Succ**(s, a).

Simultaneous games

On the contrary of turn-based games, no ordering on the player's moves in simultaneous games.

Single-move simultaneous game - **Players** $= \{A, B\}$ with given possible actions. $V(a, b)$: **A's utility** if A chooses action a and B chooses action b. **Payoff matrix**: $V \in |\text{Actions}|^2$.

Pure strategy - just a single action: $\boxed{a \in \text{Actions}}$. **Mixed strategy** - a probability distribution over actions: $\boxed{\forall a \in \text{Actions}, 0 \leq \pi(a) \leq 1}$. Examples:

- Fixed, always show "1": $\pi = [1, 0]$
- Fixed, always show "2": $\pi = [0, 1]$
- Mixed, uniformly random: $\pi = [\frac{1}{2}, \frac{1}{2}]$

Game evaluation - the value of the game if player A follows π_A and player B follows π_B:

$\boxed{V(\pi_A, \pi_B) = \sum_{a,b} \pi_A(a) \pi_B(b) V(a, b)}$

von Neumann minimax theorem - for every simultaneous two-player zero-sum game with a finite number of actions:

$\boxed{\max_{\pi_A} \min_{\pi_B} V(\pi_A, \pi_B) = \min_{\pi_B} \max_{\pi_A} V(\pi_A, \pi_B)}$

where π_A, π_B range over **mixed strategies**.

Non-zero games

Payoff matrix - utility for player p: $V_p(\pi_A, \pi_B)$.

Nash equilibrium - $(\pi_A^\star, \pi_B^\star)$ such that no player has an incentive to change their strategy:

$\boxed{\forall \pi_A, V_A(\pi_A^\star, \pi_B^\star) \geq V_A(\pi_A, \pi_B^\star)}$ and

$\boxed{\forall \pi_B, V_B(\pi_A^\star, \pi_B^\star) \geq V_B(\pi_A^\star, \pi_B)}$. **Nash's existence theorem** - in any finite-player game with finite number of actions, there exists **at least one** Nash equilibrium.