CS 540 Introduction to Artificial Intelligence

Review for Search, Game and RL

Yudong Chen
University of Wisconsin-Madison

Dec 9, 2021
Announcements

• Please fill out course evaluation survey

• Homework:
  – HW10 due next Tuesday (before last class)

• Final exam: Dec 20, 2:45-4:45pm, online

• Class roadmap:
  – Today: Demonstration for RL; Review on search, games, RL
  – Next Tuesday: Ethics and Trust in AI
Demonstration: GridWorld

Transition is deterministic (robot moves exactly as told)

Game does not terminate

- Reaching B’s: pay -1 and game continues
- Reaching G: get +1 and robot teleports to initial state A

Discount factor = 0.9
What do optimal value/policy look like?

Let’s guess:

• Optimal route?
• $V^*(B8)$ vs $V^*(B9)$? $V^*(B8) < V^*(B9)$
• $V^*(A) = 0.22$. Then $V^*(G) =$?
• $V^*(B3) \approx 0$
• If reward($B3$) changes to $-0.5$, should we go through it?

$$V^*(G) = 1 + 0.9 \times V^*(A) = 1 + 0.9 \times 0.22 = 1.198 \times 1.2$$
What do optimal value/policy look like?

**Truth:**

- Optimal route (see left)
- $V^*(B8) = -0.28 < -0.21 = V^*(B9)$
- $V^*(A) = 0.22$. Then $V^*(G) = 1.20$
- $V^*(B3) = 0.08$ (close to 0)
- If reward($B3$) changes to $-0.5$, we should go through $B3$. 
Visualization of Q Learning and $\varepsilon$-Greedy

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

– “Reinit agent” resets the board
– “Toggle TD Learning” starts or stops the agent running
Review: Outline

• Search
  – Uninformed vs Informed
  – Optimization

• Games
  – Game theory basics, dominant strategy, equilibrium
  – Minimax search

• Reinforcement Learning
  – MDPs, value iteration, Q-learning
Uninformed vs Informed Search

Uninformed search (all of what we saw). Know:
• Path cost $g(s)$ from start to node $s$
• Successors.

Informed search. Know:
• All uninformed search properties, plus
• Heuristic $h(s)$ from $s$ to goal (recall game heuristic)
Uninformed Search: Iterative Deepening DFS

Repeated limited DFS

• Search like BFS, fringe like DFS

• **Properties:**
  – Complete
  – Optimal (if edge cost 1)
  – Time $O(b^d)$
  – Space $O(bd)$

• **Preferred algorithm** for uninformed search
# Performance of Search Algorithms on Trees

<table>
<thead>
<tr>
<th></th>
<th>Complete</th>
<th>optimal</th>
<th>time</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth-first search</td>
<td>Y</td>
<td>Y, if ¹</td>
<td>O(b^d)</td>
<td>O(b^d)</td>
</tr>
<tr>
<td>Uniform-cost search ²</td>
<td>Y</td>
<td>Y</td>
<td>O(b^{C*/e})</td>
<td>O(b^{C*/e})</td>
</tr>
<tr>
<td>Depth-first search</td>
<td>N</td>
<td>N</td>
<td>O(b^m)</td>
<td>O(bm)</td>
</tr>
<tr>
<td>Iterative deepening</td>
<td>Y</td>
<td>Y, if ¹</td>
<td>O(b^d)</td>
<td>O(bd)</td>
</tr>
</tbody>
</table>

1. edge cost constant, or positive non-decreasing in depth
2. edge costs \( \geq \varepsilon > 0 \). \( C^* \) is the best goal path cost.

---

b: branching factor (assume finite)  
d: goal depth  
m: graph depth
Informed Search: A* Search

A*: Expand best $g(s) + h(s)$, with one requirement

• Demand that $h(s) \leq h^*(s)$

• If heuristic has this property, “admissible”
  – Optimistic! Never over-estimates

• Still need $h(s) \geq 0$
  – Negative heuristics can lead to strange behavior
Search vs. Optimization

Before: wanted a path from start state to goal state
• Uninformed search, informed search

New setting: optimization
• States $s$ have values $f(s)$
• Want: $s$ with optimal value $f(s)$ (i.e., optimize over states)
• Challenging setting: too many states for previous search approaches, but maybe not a continuous function for SGD.
Hill Climbing Algorithm

Pseudocode:

1. Pick initial state $s$
2. Pick $t$ in \texttt{neighbors}(s) with the largest $f(t)$
3. if $f(t) \leq f(s)$ THEN stop, return $s$
4. $s \leftarrow t$. goto 2.

What could happen? \textbf{Local optima!}
Hill Climbing: Local Optima

Note the local optima. How do we handle them?
Simulated Annealing

A more sophisticated optimization approach.

• **Idea:** move quickly at first, then slow down

• **Pseudocode:**

```
Pseudocode:
Pick initial state x
For k = 0 through k_{max}:
    Reduce temperature T
    Pick a random neighbour, y ← neighbor(x)
    If f(y) ≥ f(x), then x ← y
    Else, with prob. P(f(x), f(y), T) then x ← y
Output: the final state x
```
Simulated Annealing: Picking Probability

How do we pick probability $P$?

- Decrease with gap $|f(x) - f(y)|$
- Decrease with time $k$

- Temperature $T$ cools over time
  - High temperature, accept any $y$
  - Low temperature, behaves like hill-climbing
  - Still, $|f(x) - f(y)|$ plays a role: if big, replacement probability low.

$$P(x, y, T) = \exp\left(-\frac{|f(x) - f(y)|}{T}\right)$$
Genetic Algorithms

Goal of genetic algorithms: optimize using principles inspired by mechanism for evolution

- E.g., analogous to **natural selection, cross-over, and mutation**
Games setup: **multiple** agents

- Now: interactions between agents
- Agents want to maximize utility
- **Strategic** decision making.
Modeling Games: Properties

Let’s work through **properties** of games

- **Number** of agents/players
- State & action spaces: **discrete** or **continuous**
- **Finite** or **infinite**
- **Deterministic** or **random**
- **Sum**: zero or positive or negative
- **Sequential** or **simultaneous**
Simultaneous Games: Normal Form

- \( n \) players \( \{1,2,\ldots,n\} \)
- Player \( i \) strategy \( a_i \) from \( A_i \).
  - Strategy of all players: \( a = (a_1, a_2, \ldots, a_n) \)
- Player \( i \) gets rewards \( u_i(a) \) for any outcome
  - Note: reward depends on other players!

<table>
<thead>
<tr>
<th>Player 2</th>
<th>Stay silent</th>
<th>Betray</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stay silent</td>
<td>-1, -1</td>
<td>-3, 0</td>
</tr>
<tr>
<td>Betray</td>
<td>0, -3</td>
<td>-2, -2</td>
</tr>
</tbody>
</table>
Dominant Strategies and Equilibria

- \( a_i \) is **dominant** if \( a_i \) better than \( a_i' \) regardless of what other players do
  \[
  u_i(a_i, a_{-i}) \geq u_i(a_i', a_{-i}) \forall a_i' \neq a_i \text{ and } \forall a_{-i}
  \]

- \( a^* = (a_1^*, ..., a_n^*) \) is an **equilibrium** if all the players do not have an incentive to **unilaterally deviate**
  \[
  u_i(a_i^*, a_{-i}^*) \geq u_i(a_i, a_{-i}^*) \quad \forall a_i \in A_i
  \]

- A **mixed strategy** \( x^* = (x_1^*, ..., x_n^*) \) is a **Nash equilibrium** if
  \[
  u_i(x_i^*, x_{-i}^*) \geq u_i(x_i, x_{-i}^*) \quad \forall x_i \in \Delta A_i, \forall i \in \{1, \ldots, n\}
  \]
Dominant Strategies and Equilibria

- Dominant strategies $\implies$ (Pure) Equilibrium $\implies$ NE
  - Not the other way around

- NE always exists. Not necessarily for the other two
Sequential Games

More complex games with multiple moves

• Instead of normal form, **extensive form**
• Represent with a **tree**
• Perform search over the tree

• Can still look for Nash equilibrium
  – Or, other criteria like **maximin / minimax**

Wiki
Minimax Value and Strategies

Let’s stick to zero-sum two-player games

• Write down all the pure strategies (e.g., the big tree) and select the $s^*$ and $t^*$

$$s^* = \arg \max_{s \in S} \min_{t \in T} u(s, t) \quad t^* = \arg \min_{t \in T} \max_{s \in S} u(s, t)$$

• Can implement this as depth-first search: \textbf{minimax algorithm}
Minimax Search with $\alpha$-$\beta$ pruning

**Max-Value** function:

```plaintext
function Max-Value (s, $\alpha$, $\beta$)
inputs:
s: current state in game, Max about to play
$\alpha$: best score (highest) for Max along path to s
$\beta$: best score (lowest) for Min along path to s
output: $\min(\beta, \text{best-score (for Max) available from s})$
    if (s is a terminal state)
        then return (terminal value of s)
    else for each $s'$ in Succ(s)
        $\alpha := \max(\alpha, \text{Min-value}(s', \alpha, \beta))$
        if ($\alpha \geq \beta$) then return $\beta$ /* alpha pruning */
    return $\alpha$
```

**Min-Value** function:

```plaintext
function Min-Value(s, $\alpha$, $\beta$)
output: $\max(\alpha, \text{best-score (for Min) available from s})$
    if (s is a terminal state)
        then return (terminal value of s)
    else for each $s'$ in Succ(s)
        $\beta := \min(\beta, \text{Max-value}(s', \alpha, \beta))$
        if ($\alpha \geq \beta$) then return $\alpha$ /* beta pruning */
    return $\beta$
```

Starting from the root:
Max-Value(root, -\infty, +\infty)
Minimax Search with Heuristics

• Long games are yield huge computation
• To deal with this: limit $d$ for the search depth
• **Q:** What to do at depth $d$, but no termination yet?
  – **A:** Use a heuristic evaluation function $e(x)$

```plaintext
function MINIMAX(x, d) returns an estimate of $x$’s utility value
    inputs: x, current state in game
            d, an upper bound on the search depth
    if $x$ is a terminal state then return Max’s payoff at $x$
    else if $d = 0$ then return $e(x)$
    else if it is Max’s move at $x$ then
        return $\max\{\text{MINIMAX}(y, d-1) : y$ is a child of $x\}$
    else return $\min\{\text{MINIMAX}(y, d-1) : y$ is a child of $x\}$
```

Credit: Dana Nau
Reinforcement Learning

Basic setup:

• Set of states, $S$
• Set of actions $A$
• Information: at time $t$, observe state $s_t \in S$. Get reward $r_t$
• Agent makes choice $a_t \in A$. State changes to $s_{t+1}$, continue

Goal: find a map from **states to actions** maximize rewards.

A “policy”
Markov Decision Process (MDP)

The formal mathematical model:

- **State set** $S$. Initial state $s_0$. **Action set** $A$
- **State transition model**: $P(s_{t+1} | s_t, a_t)$
  - Markov assumption: transition probability only depends on $s_t$ and $a_t$, and not previous actions or states.
- **Reward function**: $r(s_t)$
- **Policy**: $\pi(s) : S' \rightarrow A$ action to take at a particular state.

$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$
Value function

The value function for policy $\pi$ at state $s_0$ is the expected utility over all possible state sequences from $s_0$ produced by following that policy:

$$V^\pi(s_0) = \sum_{\text{sequences starting from } s_0} P(\text{sequence})U(\text{sequence})$$

where the utility of a sequence is its corresponding discounted cumulative reward:

$$U(s_0, s_1 \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots = \sum_{t \geq 0} \gamma^t r(s_t)$$

$\gamma \in (0,1)$: discount factor.
Bellman Equation

• Set $V^*(s)$ to be value function for optimal policy.
• $V^*(s)$ satisfies the Bellman Equation: for all $s$,

$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s')$$

- Current state reward
- Discounted expected future rewards
Value Iteration

**Q:** How do we find $V^*(s)$?

- Know: reward $r(s)$, transition probability $P(s' | s, a)$
- Also know $V^*(s)$ satisfies Bellman equation

**A:** Start with $V_0(s) = 0$, $\forall s$. Then for all $s$, update

$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s' | s, a) V_i(s')$$
Now that $V^*(s_0)$ is known, what $a$ should we take?

- What’s the expected utility of an action $a$ in state $s$?
  
  $$\sum_{s'} P(s'|s, a)V^*(s')$$

- So, to get the optimal policy, compute
  
  $$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s, a)V^*(s')$$
Q-Learning

What if we don’t know transition probability $P(s' | s, a)$?

- **Q-learning**: get an action-value function $Q(s, a)$ that tells us the value of doing $a$ in state $s$

- How do we get $Q(s, a)$? Similar iterative procedure:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Learning rate
SARSA

An alternative:

- Just use the next action, no max over actions:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r(s_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \]

- Called state–action–reward–state–action (SARSA)

Learning rate
Epsilon-Greedy Policy

Need to balance **exploitation** and **exploration**

- With some $0<\epsilon<1$ probability, take a random action at each state, or else the action with highest $Q(s,a)$ value.

\[
a = \begin{cases} 
\arg\max\limits_{a \in A} Q(s, a) & \text{uniform}(0, 1) > \epsilon \\
\text{random } a \in A & \text{otherwise}
\end{cases}
\]

- Can be used in Q Learning and SARSA
Acknowledgements: Based on slides from Fred Sala, Yin Li, Jerry Zhu, Svetlana Lazebnik, Yingyu Liang, David Page, Mark Craven, Pieter Abbeel, Dan Klein