Lecture 17: MDP: Value Iteration and Policy Iteration
Outline

• The Bellman equation and Bellman update

• Contraction

• Value iteration

• Policy iteration
The Bellman equations for utilities

- The relationship between the utility of a state and the utility of its neighbors

\[ U(s) = R(s) + r \cdot \max_{a \in \text{Actions}(s)} \sum_{s', P(s'|s,a)} U(S') \]

- Assuming the agent chooses the optimal action

- What is the best action in state (1,1)? in state (3,4)?

- How many Bellman equations do we have for this MDP?

- Can we solve these equations directly and efficiently?
Value iteration: Idea

- Start with estimate $U_0 = 0$,
- Keep plugging in current estimate $U_i$ to get new estimate $U_{i+1}$;
- Repeat until little or no change in estimation.
Value iteration: Algorithm

- Initialize \( U_0(S) = 0 \) for all \( S \)

- For \( I = 0, 1, 2, \ldots \)
  - For all \( S \), \( U_{i+1}(s) = R(s) + r \cdot \max_{a \in \text{Actions}(s)} \sum P(s'|s,a) U_i(S') \)
    - "Bellman update"

- If \( \max_s |U_{i+1}(s) - U_i(s)| < \epsilon \), stop and output \( U_{i+1} \).

- For all \( S \), \( \Pi^*(s) = \arg \max_a \sum_{s'} P(s'|s,a) U(S') \)
A contraction is a function of one argument. When applied to two different inputs in turn, the output values are getting “closer together”.

- A contraction has one fixed point.
- Ex. “divided by 2” is a contraction. The fixed point is 0.

Bellmen update is a contraction. Its fixed point is the vector/point of the true utilities of the states.

The estimate of utility at each iteration is getting closer to the true utility.
Policy iteration: Algorithm

- Start with any policy $\Pi_0$.
- For $i = 0, 1, 2, \ldots$
  - Evaluate: compute $U^{\Pi_i}(s)$
  - Greedify: $\Pi_{i+1}(s) = \arg\max_a \sum_{s'} P(s'|s,a)U^{\Pi_i}(S')$
  - Stop when $\Pi_{i+1} = \Pi_i$. 
Policy iteration: how to evaluate $\pi$?

- **Iterative approach** – simplified value iteration.
  - Like value iteration, except now action at state $S$ is fixed to be $\pi(S)$.
  - $U_{i+1}^\pi(s) = R(s) + \sum_{s'} P(s'|\pi(s),a) U_i^\pi(s')$

- **Direct approach**.
  - $U^\pi(s) = R(s) + \sum_{s'} P(s'|\pi(s),a) U^\pi(s')$
  - A system of linear equations, can be solved directly in $O(n^3)$.
  - Efficient for small state spaces.
Policy iteration: why does it work?

• Can prove (Policy improvement theorem)
  • $U^{\Pi_{i+1}}(s) \geq U^{\Pi_i}(s)$, with strict inequality for some $s$ unless $\Pi_i = \Pi^*$

• Means policies getting better and better $\Pi_{i+1}$
  • Will never visit same policy $\Pi$ twice
  • Will only terminate when reach $\Pi^*$

• $\#\text{iterations} \leq \#\text{policies}$
  • In practice, no case found where more than $O(n)$ iterations are needed.
  • Open question: does policy iteration converge in $O(n)$? ($n$ is the number of states in the MDP)
POMDP: Example and definition

- A robot in a grid

- MDP:
  - Initial state and states: hidden
  - Actions:
  - Transition model: \( P(s' | s, a) \)
  - Rewards: \( R(s) \)
  - Observation model: \( P(o | s) \)
Review questions: true or false

1. Value iteration is an algorithm for estimating the true utility of each state in a MDP.

2. The $n$ ($n$ is the number of states) Bellman equations for utility in a MDP can uniquely determine the true utilities of the states. These equations can be solved directly since they are exactly $n$ variables and $n$ equations.

3. Bellman update is a contraction. The fixed point is the vector/point of the true utilities of the states.

4. To evaluate a policy (compute $U^\pi(s)$), we can write $n$ Bellman equations with the actions fixed as $\Pi(s)$. A simplified value iteration algorithm can be used to solve them since they can not be solved directly.
5. In Policy iteration, a policy will not be visited twice. Each iteration will lead to a new policy that is strictly better than the last one for at least one state.

6. The number of different policies is $m^n$ (m is the average number of actions available for each state, and n is the number of states in a MDP). So policy iteration usually takes exponential time to run.

7. Policy iteration is guaranteed to terminate and find an optimal policy.

8. In POMDPs (partially observable MDPs), the agent does not know the state it is in. Instead of a transition model $P(s' | s, a)$, it has an observation model $P(o | s)$. 
• **W4 is due on Tuesday Nov. 24th**
  – Turn in hard copy in class.

• **P4 has been released and is due on Tuesday Dec. 1st**
  – Upload files to CS dropbox by midnight.