DYNAMIC PROGRAMMING
REINFORCEMENT LEARNING

COGS 202 : Week 7 Presentation
OUTLINE

- Recap (State Value and Action Value functions)
- Computations in MDP
- Dynamic Programming (DP)
- Policy Evaluation
- Policy Improvement
- Policy Iteration
- Value Iteration
- Asynchronous DP
- Generalized Policy Iteration
- Efficiency of DP
COMPONENTS OF A MDP PROBLEM

- Agent, task, environment
- States, actions, rewards
- Policy \( \pi(s, a) \) probability of doing \( a \) in \( s \)
- State Value \( V^\pi(s) \) number – Value of a state
- Action Value \( Q^\pi(s, a) \) number – Value of a state-action pair
- Model \( P^a_{ss'} \) probability of going from \( s \rightarrow s' \) taking action \( a \)
- Reward function \( R^a_{ss'} \) from doing \( a \) in \( s \) and reaching \( s' \)
- Return \( R \rightarrow \) sum of future rewards
**Value functions**

- **State value function:** $V^\pi(s)$
  - Expected return when starting in $s$ and following $\pi$

- **State-action value function:** $Q^\pi(s,a)$
  - Expected return when starting in $s$, performing $a$, and following $\pi$

- Useful for finding the optimal policy
  - Can estimate from experience
  - Pick the best action using $Q^\pi(s,a)$

**Bellman equation**

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} \left[ r^a_{ss'} + \gamma V^\pi(s') \right] = \sum_a \pi(s, a) Q^\pi(s, a)$$
Optimal Value Functions

- There’s a set of optimal policies
  - \( V^\pi \) defines partial ordering on policies
  - They share the same optimal value function
    \[
    V^*(s) = \max_\pi V^\pi(s)
    \]

- Bellman optimality equation
  \[
  V^*(s) = \max_a \sum_{s'} P_{ss'}^a \left[ r_{ss'}^a + \gamma V^*(s') \right]
  \]
  - System of \( n \) non-linear equations
  - Solve for \( V^*(s) \)
  - Easy to extract the optimal policy

- Having \( Q^*(s,a) \) makes it even simpler
  \[
  \pi^*(s) = \arg \max_a Q^*(s,a)
  \]
**Key Computations**

- How to compute $V^\pi(s)$ given a fixed policy $\pi$?
- How to compute $\pi'$ such that $V^{\pi'} \geq V^\pi$?
- How to compute $\pi^*$?
- How to compute $V^*(s)$ directly?
Solutions

Dynamic Programming
- Classical solution methods for MDPs
- Used to compute value functions, and hence, optimal policies using Bellman equations
- Efficiency and utility

Assumptions
- Finite MDP – State and Action sets are finite
**Policy Evaluation**

**Policy Evaluation**: for a given policy \( \pi \), compute the state-value function \( V^\pi \)

Recall: State value function for policy \( \pi \):

\[
V^\pi(s) = E_\pi \left\{ R_t \mid s_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}
\]

Bellman equation for \( V^* \):

\[
V^*(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[ R^a_{ss'} + \gamma V^*(s') \right]
\]

A system of \( |S| \) simultaneous linear equations
Iterative Methods

\[ V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_k \rightarrow V_{k+1} \rightarrow \cdots \rightarrow V^\pi \]

A “sweep”

A sweep consists of applying a backup operation to each state.

A full policy evaluation backup:

\[
V_{k+1}(s) \leftarrow \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V_k(s') \right]
\]
**Iterative Policy Evaluation**

Input $\pi$, the policy to be evaluated

Initialize $V(s) = 0$, for all $s \in S^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in S$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(s, a) \sum_{s'} P_{ss'}^{a} \left[ R_{ss'}^{a} + \gamma V(s') \right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output $V \approx V^\pi$
POLICY IMPROVEMENT

Suppose we have computed $V^*$ for a deterministic policy $\pi$.

For a given state $s$, would it be better to do an action $a \neq \pi(s)$?

The value of doing $a$ in state $s$ is:

$$Q^\pi(s, a) = E_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P^a_{ss'} \left[ R^a_{ss'} + \gamma V^\pi(s') \right]$$

It is better to switch to action $a$ for state $s$ if and only if

$$Q^\pi(s, a) > V^\pi(s)$$
THE POLICY IMPROVEMENT THEOREM

Let $\pi$ and $\pi'$ be two policies such that for all $s \in S$:

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s)$$

Then $\pi'$ is a better policy than $\pi$, i.e. for all $s \in S$:

$$V^{\pi'}(s) \geq V^\pi(s)$$
PROOF SKETCH

\[ V^\pi (s) \leq Q^\pi (s, \pi' (s)) \]

\[ = E_{\pi'} \{ r_{t+1} + \gamma V^\pi (s_{t+1}) \mid s_t = s \} \]

\[ \leq E_{\pi'} \{ r_{t+1} + \gamma Q^\pi (s_{t+1}, \pi' (s_{t+1})) \mid s_t = s \} \]

\[ = E_{\pi'} \{ r_{t+1} + \gamma E_{\pi'} \{ r_{t+2} + \gamma V^\pi (s_{t+2}) \} \mid s_t = s \} \]

\[ = E_{\pi'} \{ r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi (s_{t+2}) \mid s_t = s \} \]

\[ \leq E_{\pi'} \{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi (s_{t+3}) \mid s_t = s \} \]

\[ \vdots \]

\[ \leq E_{\pi'} \{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots \mid s_t = s \} \]

\[ = V^{\pi'} (s) . \]
Do this for all states to get a new policy $\pi'$ that is **greedy** with respect to $V^\pi$:

$$\pi'(s) = \arg\max_a Q^\pi (s, a)$$

$$= \arg\max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi (s')]$$

Then $V^{\pi'} \geq V^\pi$
What if $V^{\pi'} = V^\pi$ ?

i.e., for all $s \in S$, $V^{\pi'}(s) = \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]$

But this is the Bellman Optimality Equation.

So $V^{\pi'} = V^*$ and both $\pi$ and $\pi'$ are optimal policies.
**Policy Iteration**

\[ \pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \cdots \pi^* \rightarrow V^* \rightarrow \pi^* \]

- **Policy Evaluation**
- **Policy Improvement**
- "Greedification"
POLICY ITERATION

1. Initialization
   \[ V(s) \in \mathbb{R} \text{ and } \pi(s) \in \mathcal{A}(s) \text{ arbitrarily for all } s \in \mathcal{S} \]

2. Policy Evaluation
   Repeat
   \[ \Delta \leftarrow 0 \]
   For each \( s \in \mathcal{S} \):
   \[ v \leftarrow V(s) \]
   \[ V(s) \leftarrow \sum_{s'} \mathcal{P}^{\pi(s)}_{ss'} \left[ \mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right] \]
   \[ \Delta \leftarrow \max(\Delta, |v - V(s)|) \]
   until \( \Delta < \theta \) (a small positive number)

3. Policy Improvement
   \[ policy-stable \leftarrow true \]
   For each \( s \in \mathcal{S} \):
   \[ b \leftarrow \pi(s) \]
   \[ \pi(s) \leftarrow \arg \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[ \mathcal{R}^a_{ss'} + \gamma V(s') \right] \]
   If \( b \neq \pi(s) \), then \( policy-stable \leftarrow false \)
   If \( policy-stable \), then stop; else go to 2
VALUE ITERATION

- **Drawback to policy iteration** is that each iteration involves a policy evaluation, which itself may require multiple sweeps.
- **Convergence of** $V^\pi$ **occurs only in the limit** so that we in principle have to wait until convergence.
- As seen, the **optimal policy is often obtained long before** $V^\pi$ **has converged**.
- **Policy evaluation step** can be truncated in several ways without losing the convergence guarantees of policy iteration.
- **Value iteration** is to stop policy evaluation after just one sweep.
**Value Iteration**

Recall the **full policy evaluation backup**:

\[
V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V_k(s') \right]
\]

Here is the **full value iteration backup**:

\[
V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V_k(s') \right]
\]

Combination of policy improvement and truncated policy evaluation.
**Value Iteration Cont.**

Initialize $V$ arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$\nu \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}_{ss'}^a \left[ \mathcal{R}_{ss'}^a + \gamma V(s') \right]$

$\Delta \leftarrow \max(\Delta, |\nu - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi$, such that

$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^a \left[ \mathcal{R}_{ss'}^a + \gamma V(s') \right]$
ASYNCHRONOUS DP

- All the DP methods described so far require exhaustive sweeps of the entire state set.
- Asynchronous DP does not use sweeps. Instead it works like this:
  - Repeat until convergence criterion is met: Pick a state at random and apply the appropriate backup
- Still needs lots of computation, but does not get locked into hopelessly long sweeps
- Can you select states to backup intelligently? YES: an agent’s experience can act as a guide.
**Generalized Policy Iteration**

Generalized Policy Iteration (GPI): any interaction of policy evaluation and policy improvement, independent of their granularity.

A geometric metaphor for convergence of GPI:
EFFICIENCY OF DP

- To find an optimal policy is polynomial in the number of states.
- BUT, the number of states is often astronomical, e.g., often growing exponentially with the number of state variables (what Bellman called “the curse of dimensionality”).
- In practice, classical DP can be applied to problems with a few millions of states.
- Asynchronous DP can be applied to larger problems, and appropriate for parallel computation.
- It is surprisingly easy to come up with MDPs for which DP methods are not practical.
SUMMARY

- Policy evaluation: backups without a max
- Policy improvement: form a greedy policy, if only locally
- Policy iteration: alternate the above two processes
- Value iteration: backups with a max
- Generalized Policy Iteration (GPI)
- Asynchronous DP: a way to avoid exhaustive sweeps