

# **Multi Agent Systems**

## - Lab 3 -

### MDP Value and Policy Iteration Analysis

Slides adapted from Reinforcement Learning class by Vien Ngo,  
University of Stuttgart, 2016

## Markov decision process

- A reinforcement learning problem that satisfies the Markov property is called a Markov decision process, or MDP.
- $\text{MDP} = \{S, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{P}_0, \gamma\}$ .
  - $S$ : consists of all possible states.
  - $\mathcal{A}$ : consists of all possible actions.
  - $\mathcal{T}$ : is a transition function which defines the probability  $\mathcal{T}(s', s, a) = \text{Pr}(s' | s, a)$ .
  - $\mathcal{R}$ : is a reward function which defines the reward  $\mathcal{R}(s, a)$ .
  - $\mathcal{P}_0$ : is the probability distribution over initial states.
  - $\gamma \in [0, 1]$ : is a discount factor.

# State Value Function

The **value** (*expected discounted* return – for infinite horizon settings) of a policy  $\pi$  when started in state  $s$ :

$$V^\pi(s) = \mathbb{E}_\pi \{ r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s \}$$

where  $0 < \gamma < 1$  is the *discounting* factor

**Optimality:** policy  $\pi^*$  is optimal iff

$$\forall_s : V^{\pi^*}(s) = V^*(s) \quad \text{where } V^*(s) = \max_{\pi} V^\pi(s)$$

# Bellman Principle of Optimality

$$V^*(s) = \max_a \left[ R(a, s) + \gamma \sum_{s'} P(s' | a, s) V^*(s') \right]$$
$$\pi^*(s) = \operatorname{argmax}_a \left[ R(a, s) + \gamma \sum_{s'} P(s' | a, s) V^*(s') \right]$$

# Value Iteration

Given the Bellman equation

$$V^*(s) = \max_a \left[ R(a, s) + \gamma \sum_{s'} P(s' | a, s) V^*(s') \right]$$

→ iterate

$$\forall_s : V_{k+1}(s) = \max_a \left[ R(a, s) + \gamma \sum_{s'} P(s' | \pi(s), s) V_k(s') \right]$$

stopping criterion:

$$\max_s |V_{k+1}(s) - V_k(s)| \leq \epsilon$$

# Policy Iteration

**Value Iteration** computes  $\mathbf{V}^*$  directly

To **evaluate** a given policy  $\pi$ , one needs to compute  $V^\pi$ , that is iterate using  $\pi$  instead of  $\max_a$

$$\forall_s : V_{k+1}(s) = R(\pi(s), s) + \gamma \sum_{s'} P(s'|\pi(s), s) V_k(s') \quad \leftarrow \text{Counts as one iteration}$$

Optimal policy can then be computed in iterative way

## Policy Iteration

1) Initialise  $\pi_0$  in a given way (e.g. randomly)

2) Iterate

- Policy Evaluation: compute  $V^{\pi_k}$
- Policy Improvement:  $\pi_{k+1}(s) = \underset{a \in A}{\operatorname{argmax}} [R(a, s) + \gamma \sum_{s'} P(s'|a, s) V^{\pi_k}(s)]$

# Value Iteration variants

## Gauss-Seidel Value Iteration

- Standard VI algorithm updates all states at next iteration using **old** values at previous iteration (each iteration finishes when all states get updated).

---

### Algorithm 1 Standard Value Iteration Algorithm

---

```
1: while (!converged) do  
2:    $V_{old} = V$   
3:   for (each  $s \in \mathcal{S}$ ) do  
4:      $V(s) = \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_{old}(s')\}$  ← Counts as one iteration
```

---

- Gauss-Seidel VI updates each state using values from previous computation.

---

### Algorithm 2 Gauss-Seidel Value Iteration Algorithm

---

```
1: while (!converged) do  
2:   for (each  $s \in \mathcal{S}$ ) do  
3:      $V(s) = \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a) V(s')\}$  ← Counts as one iteration
```

---



# Value Iteration variants


## Prioritised Sweeping

- Similar to Gauss-Seidel VI, but the sequence of states in each iteration is proportional to their update magnitudes (Bellman errors).
- Define Bellman error as  $E(s; V_t) = |V_{t+1}(s) - V_t(s)|$  that is the change of  $s$ 's value after the most recent update.

---

### Algorithm 3 Prioritised Sweeping VI Algorithm

---

- 1: Initialize  $V_0(s)$  and priority values  $H_0(s)$ ,  $\forall s \in \mathcal{S}$ .
  - 2: **for**  $k = 0, 1, 2, 3, \dots$  **do**
  - 3:   pick a state to update (with the highest priority):  $s_k \in \arg \max_{s \in \mathcal{S}} H_k(s)$
  - 4:   value update:  $V_{k+1}(s_k) = \max_{a \in \mathcal{A}} [R(s_k, a_k) + \gamma \sum_{s'} P(s'|s_k, a_k) V_k(s')]$  
  - 5:   for  $s \neq s_k$ :  $V_{k+1}(s) = V_k(s)$
  - 6:   update priority values:  $\forall s \in \mathcal{S}, H_{k+1}(s) \leftarrow E(s; V_{k+1})$  (Note: the error is w.r.t the future update).
-



# OpenAI Gym Test Environments

- Three test MDP environments based on [Open AI Gym](#):
  - [Taxi-v3](#)
    - 4 locations
    - Pickup passenger at one location and drop him off at another
    - 6 actions: move NORTH, SOUTH, EAST, WEST + PICK\_UP + DROP\_OFF
    - Rewards: +20 for successful drop-off, -1 per movement, -10 for illegal pick-up or drop-off
  - [FrozenLake-v0 \(small\) and FrozenLake8x8-v0 \(larger\)](#)
    - Agent controls movement over a grid
    - Some tiles walkable, some lead agent to fall into water; agent has a start and goal tile
    - Movement of agent with uncertainty (due to slippery ice)
    - Rewards: +1 if agent finds correct path, 0 otherwise

# Tasks

- For each game consider the following values for convergence criteria
  - max\_iterations:  $5 \cdot 10^5$
  - epsilon\_threshold:  $10^{-2}$  or  $10^{-3}$
  - Discount factor: 0.9
- **Step 1:** compute  $V^*$  using the **standard Value Iteration Algorithm**
- **Step 2:** Run the two variants of Value Iteration (**Gauss-Seidel VI, Prioritized Sweeping VI**)
- **Step 3:** Run 5 instantiations (random policy init each time) for **Policy Iteration** for each game
- For Steps 2 and 3: compute averages of **number of iterations** until convergence
- **Plot convergence graph**
  - X axis: number of iterations (**NOTE!** An **iteration** is considered **an update to a state** in the value function, **i.e. an update to  $V(s)$** )
  - Y axis:  $\|V - V^*\|_2$
- **Analyze convergence speed properties**