Reinforcement Learning (RL)

Chapter 3:

Markov Decision Processes (MDPs)

Dynamic Programming (Policy Iteration, value Iteration and Modified Policy Iteration)

Saeed Saeedvand, Ph.D.

Contents

In this Chapter:

- ✓ Environment dynamics
- ✓ Stochastic processes with Markovian assumption
- ✓ Stochastic processes with Stationary assumption
- ✓ Policy Iteration
- ✓ Value Iteration
- ✓ Modified Policy Iteration

Aim of this chapter:

- ✓ Understand concepts of formal problem of finite Markov decision processes. Discuss associative aspect choosing different actions in different situations and understand the Dynamic Programming with Policy Iteration and value Iteration algorithms with example.

Markov Processes

What is the Markov process?

- ✓ Markov chain or Markov process is a **stochastic model describing a sequence of possible events**
- ✓ The **probability of each event** depends only on the state of the **previous event**

Markov Processes

Markov Process definition

- ✓ In the control loop as we know we have **sequence of states, actions and rewards** in certain time steps (state).

There can be **two scenarios** for the environment:

- ✓ Environment is **deterministic** (simpler to plan for future)
- ✓ As we know due to uncertainty we also have **stochastic** environment (process is dynamic for both reward and next state that we want to model)

$$(S_0, A_0, R_0), (S_1, A_1, R_1), (S_2, A_2, R_2), \ldots$$

Markov Processes

Markov Process important properties

- ✓ Underlying Process of every dynamic system has a general structure
- ✓ With sufficient history (usually short) we can have prediction in dynamic system

For example temperature predication:

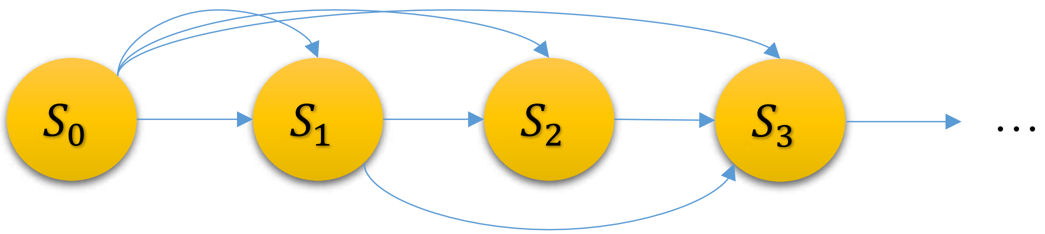
- Although there is dynamic changes but there are some underlaying rules for them
- We can use a short history to predict future
- Environment can be both deterministic or probabilistic

Markov Processes

Stochastic Process

- ✓ State S in the State-Space
- ✓ Stochastic dynamics of the environment can be represented as follows:

$$P(S_t | S_0, \dots, S_{t-1})$$



Note: This Conditional distribution can be very large (due to relations)

Markov Processes

Solution

✓ For modeling large State-Space we need to consider two assumptions:

Markov Assumption

State S' is depending on **only k previous State S** or a **finite history of states**

Stationary Process Assumption

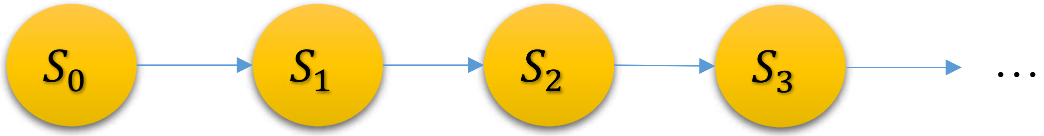
Environment's Dynamics is static during time

Markov Processes

Markov Assumption

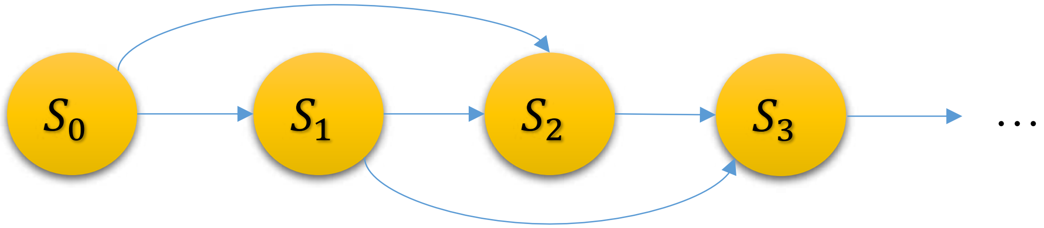
✓ State S’ is depending on **only k previous State S** or a **finite history of states**

$P(S_t|S_{t-1})$



First-order Markov Process

$P(S_t|S_{t-2},S_{t-1})$



K-order Markov Process (k=2)

Advantage

✓ Reduce the computational complexity (note we may need to add more variables in order to achieve it)

Markov Processes

Stationary Process Assumption

How to achieve stationary process?

- ✓ Adding new variables to system until dynamics of environment becomes stationary.

If we do it we can state next state can be achieved by conditional distribution:

$$\mathrm{P}(S'|S)$$

Markov Processes

Stationary Process Assumption

Example:

Considering the only right arm's end-effector:

$$\langle R_x, R_y, R_z, R_r, R_p, R_y \rangle$$

Is there any other Dynamics?

Depending on application **Velocity** here

Keep adding dynamicity into model

$$\langle R_x, R_y, R_z, R_{roll}, R_{pitch}, R_{yaw}, R_V \rangle$$

$$or$$

$$\langle R_x, R_y, R_z, R_{roll}, R_{pitch}, R_{yaw}, R_{Vx}, R_{Vy}, R_{Vz}, R_{Vroll}, R_{Vpitch}, R_{Vyaw} \rangle$$

Markov Processes

Challenge

- ✓ Adding more variables and dynamics increase the computational complexity!

Solution

- ✓ Making **tradeoff** between adding more variables with holding both Stationary and Markov assumptions

Markov Decision Processes

Idea

✓ Assuming we have Markov assumption and Stationarity the idea is:

- Predicting how actions are effecting future states

$$P(S_{t+k}|S_t)$$

- It is called calculation of state transition form S to the S'

$$P = \begin{bmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{bmatrix}$$

Markov Decision Processes

Therefore using this state transitions we have want make decisions

So we enter to **Markov Decision Process**

Markov Decision Processes (MDPs)

What is the MDPs?

- ✓ MDPs are a **classical formalization of sequential decision making** where **actions influence** not just immediate rewards, but also subsequent situations, or states, and future rewards.
- ✓ MDPs involve **delayed reward** and the need to **trade off** immediate and delayed reward.
- ✓ In MDPs we estimate the value $q_*(s, a)$ of each action a in each state s, or we estimate the value $v_*(s)$ of each state given optimal action selections.

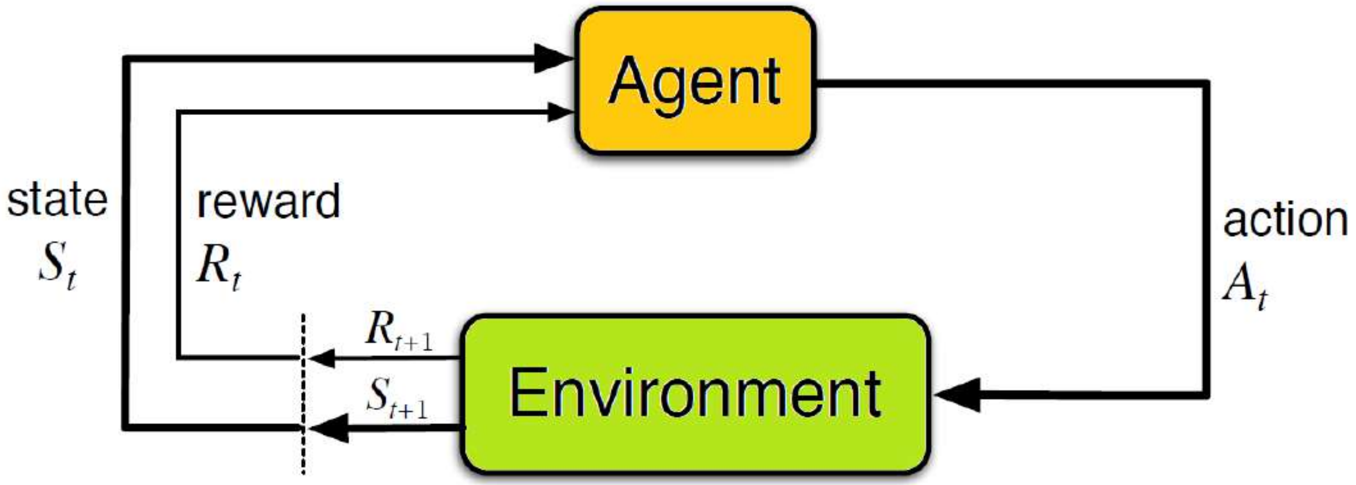
Markov Decision Processes (MDPs)

What is the MDPs?

- ✓ MDPs is **a discrete-time stochastic control process**
- ✓ MDPs provides a mathematical framework for modeling decision making in situations that outcomes are:
  - **Partly random** and **partly under the control of a decision maker**
- ✓ A **tool as a** mathematically idealized and **formal presentation** of **environment** in order to analyze and develop reinforcement learning algorithms

Markov Decision Processes (MDPs)

✓ Agent–environment interaction in a Markov decision process (similar concept)

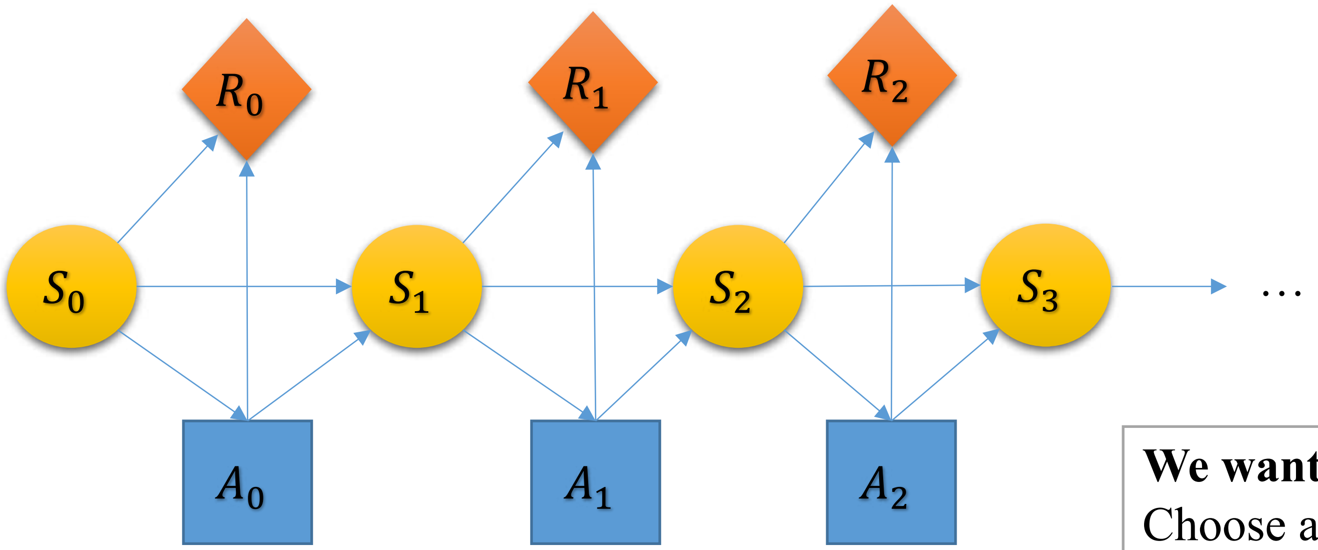


Markov Decision Processes (MDPs)

✓ Finite MDP:

MDPs are including set of state (S_t), actions (A_t), rewards (R_t)

Under Markov Assumption:



Prediction of future states and reward by actions

We want to design algorithms to:
Choose actions to maximize rewards.

Markov Decision Processes (MDPs)

Reward (remember)

- Reward is **evaluation of agent's current situation** and we define a function for it:

$$r_t = R(s_t, a_t)$$

Usually we assume it is **stationary** and is not changing during time with the same parameters of a state!

Note: We usually define a big reward for the Terminal (Success/Win)

We want to design algorithms to: Choose actions to maximize the rewards.

$$max \sum_t R(s_t, a_t)$$

Markov Decision Processes (MDPs)

Our current assumptions:

- Fully observable discreet environment
- We have a complete model and transition dynamics of the environment (no learning yet)
- Stochastic process (Uncertainty)
- Action selection is sequential and can depend on previous ones

Markov Decision Processes (MDPs)

Challenge

What if the process in the environment in **infinite**? $\sum_t R(S_t, A_t)$

We can use Discounted Factor γ

- Discount factor is to have discounted rewards as bellow:

$$\sum_t \gamma^t R(S_t, A_t)$$

✓ Where γ is between [0,1] (1 is for not discounting)

✓ Idea is that the feature rewards expected to be higher so we discount them

Markov Decision Processes (MDPs)

Challenge

What if the process in the environment in **infinite**? $\sum_t R(S_t, A_t)$

We also can use average of rewards

$$1/t \sum_t R(S_t, A_t)$$

Note: Computationally expensive, in future we can see improvements techniques

Markov Decision Processes (MDPs)

Definition of the MDPs?

- ✓ There are set of states S, actions A
- ✓ Reward model

$$R_t = R(S_t, A_t)$$

- ✓ Transition Model:

$$\mathrm{P}(S_t | S_{t-1}, a_{t-1})$$

- ✓ Discount Factor (γ)
- ✓ Horizon (h) (episode, or time steps)

Goal is to map state to action (Finding Optimal policy)

Markov Decision Processes (MDPs)

Example of the MDPs for Trade Market?

- ✓ States = Share status
- ✓ Actions = Buy, Sell, Hold
- ✓ Reward model = Profit
- ✓ Transition Model = Stochastic change of market
- ✓ Discount Factor (γ) = 0.9
- ✓ Horizon (h) (episode, or time steps) = Infinity

When to buy and when to sell to maximize the profit

What is the goal of the MDPs?

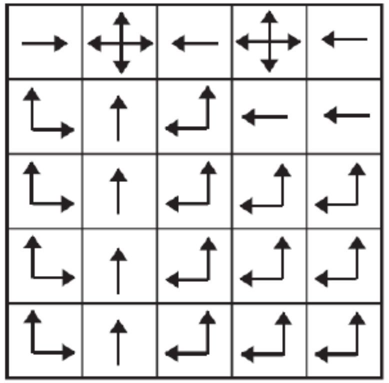
✓ The goal of **MDPs and RL** is to **find best policy π** (optimize to maximize reward)

Remember:

- Policy specifies an **action a**, that is **taken in the state s**
- More precisely, **π is a probability**, that an **action a is taken in a state s**

✓ Under Markov assumption we can present policy as:

$$a_t = \pi(s_t)$$



What is the Policy Optimization

- ✓ **Policy Optimization is** to optimizing mapping state into action

- ✓ **Therefore,** we need to **estimate value function** to evaluate current state of environment

- ✓ The value of Optimal Policy:

$$v_{\pi}^{*}(s_t)$$

- ✓ Therefore:

$$v_{\pi}^{*}(s_t) \geq v_{\pi}(s_t)$$

Dynamic Programming (Iterative Methods)

✓ **Policy Optimization algorithms**

- Policy Iteration
- Value Iteration
- …

Dynamic Programming (Policy Iteration)

Policy Iteration (based on MDPs)

- ✓ In this algorithm we optimize policy directly
- ✓ In policy iteration, we start by choosing a random or arbitrary policy.
- ✓ Then, we iteratively evaluate and improve the policy until convergence.
- ✓ Therefore policy iteration performs **two steps** until convergence:

1. **Policy evaluation**
2. **Policy improvement**

Dynamic Programming (Policy Iteration)

Policy Iteration (based on MDPs)

1 Policy evaluation step

✓ In this step **we evaluate the policy π at state s** so that we calculate the Q-value using the **Bellman equation**:

Bellman's Equation

$$V_\pi(s) = r(s) + \gamma \sum_{s' \in S} p(s', r | s, \pi(s)) V_\pi(s')$$

Dynamic Programming (Policy Iteration)

Policy Iteration (based on MDPs)

2 Policy improvement step

✓ In the policy improvement step:
- Searching for the action that maximizes the Q value at each step
- Update the policy by greedily by performed search

$$\pi(s) = argmax_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

Until converge to optimal ($V_\pi^*(s)$, and $\pi^*(s)$)

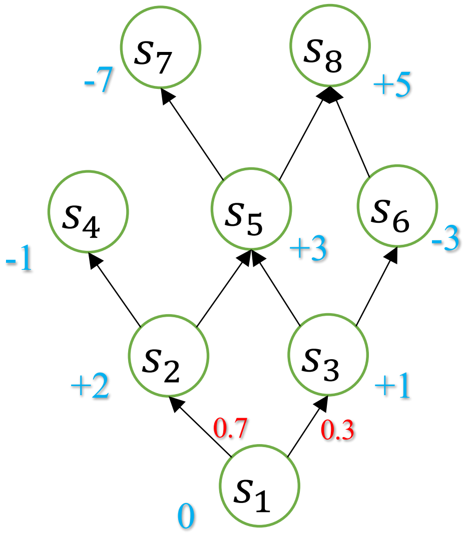
Repeat these two step until value function converge to optimal value function (replace the current action by the best action)

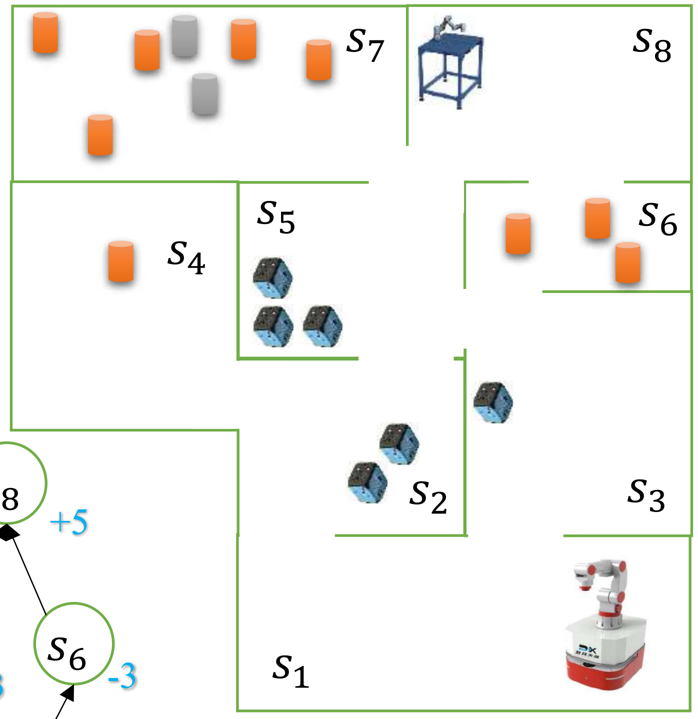
Evaluation $V_\pi(s)$ $\longrightarrow$ Improvement $\pi(s)$

Dynamic Programming (Example)

Example

- ✓ Each intersection as a state has two actions, ‘**right**’ and ‘**left**’ and only can move **upward.**
- ✓ The environment is stochastic
- ✓ We need to build a state transition probability matrix (MDPs)
- ✓ For simplicity lets assume for environment the probability of left is always 0.7, and right is 0.3



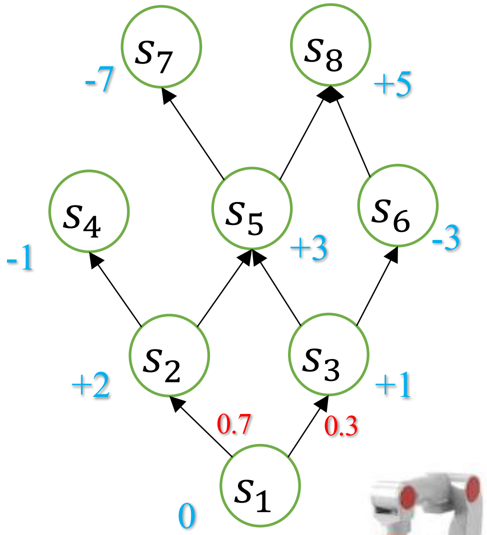


Dynamic Programming (Example)

✓ Since we have two actions we need two transition probability matrix.

For chosen action left:

State transition diagram (left):



$$T[a(left)]=\begin{array}{cc} & \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \end{array} \\ \begin{array}{c} s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} & \begin{bmatrix} 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

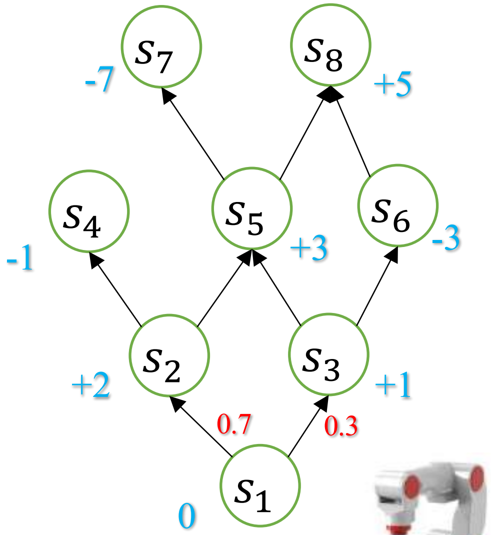
Dynamic Programming (Example)

For chosen action right:

State transition diagram (right):

$$
T[a(right)]=\begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array}
\begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \end{array}
$$

$$
\begin{bmatrix}
0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

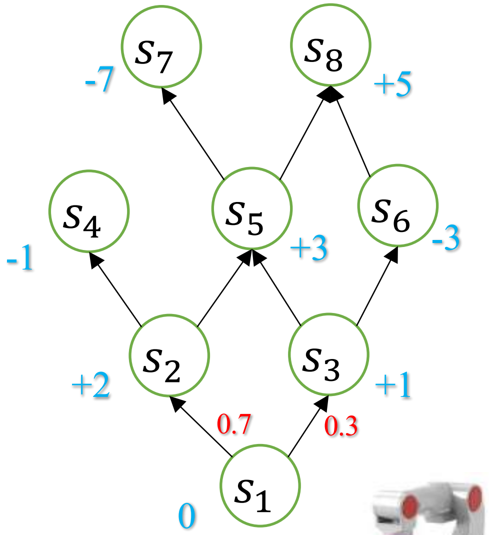


✓ Both based on the probabilities and the state transition diagram.

Dynamic Programming (Example)

Assumptions:

- Discounted factor 0.9
- All states' initial value V(s) is 0 in biggening



Initial Random Policy:

Policy (π)	S1	S2	S3	S4	S5	S6	S7	S8
	R	R	R	-	R	R	-	-

Dynamic Programming (Policy Iteration)

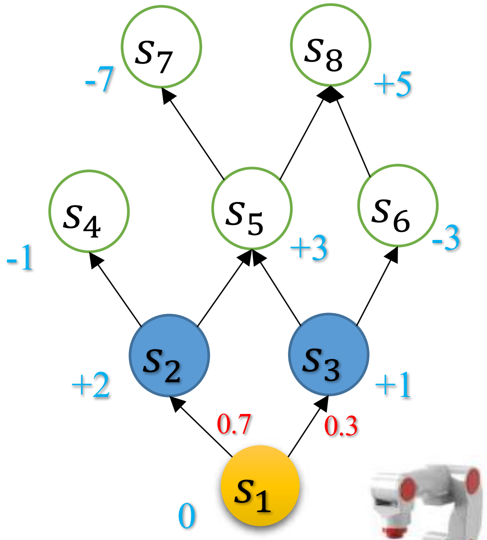
Example

Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
R	R	R	-	R	R	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	0	0	0	0	0	0	0



$$V_\pi(s) = r(s) + \gamma \sum_{s' \in S} p(s', r | s, \pi(s)) V_\pi(s')$$

1 Policy evaluation step

In the first iteration for the **Policy Evaluation (Only S1)**:

$$V_\pi(s_1) = r(s_1) + 0.9 \sum_{s' \in S} p(s', r | s_1, \pi(s_1)) V_\pi(s')$$

$$V_\pi(s_1) = 0 + 0.9 * (0.7 * 0 + 0.3 * 0) = 0$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0							

Dynamic Programming (Policy Iteration)

Example

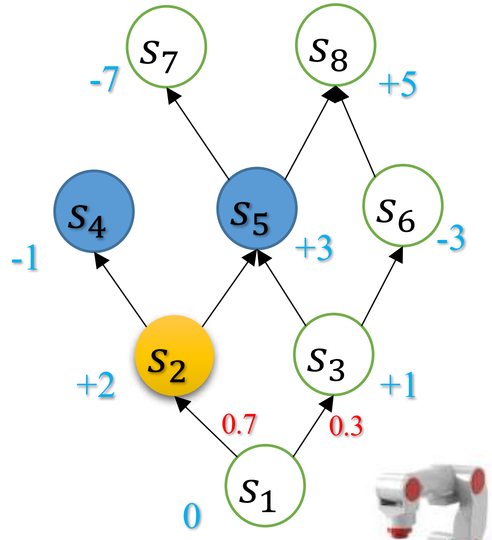
Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
R	R	R	-	R	R	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	0	0	0	0	0	0	0

$$V_\pi(s) = r(s) + \gamma \sum_{s' \in S} p(s', r | s, \pi(s)) V_\pi(s')$$



1 **Policy evaluation step**

In the first iteration for the **Policy Evaluation (Only s_2)**:

$$V_\pi(s_2) = r(s_2) + 0.9 \sum_{s' \in S} p(s', r | s_2, \pi(s_2)) V_\pi(s')$$

$$V_\pi(s_2) = 2 + 0.9 * (0.7 * 0 + 0.3 * 0) = 2$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

If we continue Policy evaluation steps for all states (only reward comes, since V_π are zero)

$$V_\pi(s) = r(s) + \gamma \sum_{s' \in S} p(s', r | s, \pi(s)) V_\pi(s')$$

Dynamic Programming (Policy Iteration)

Example

Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	R	R	-	R	R	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

$$\pi(s) = \boxed{argmax_a \gamma} \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

$$T[a(left)] = \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{c} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \end{array} \\ \begin{bmatrix} 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$T[a(right)] = \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{c} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \end{array} \\ \begin{bmatrix} 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

2 Policy improvement steps

✓ **First** iteration for the **Policy Evaluation** s_1:

For s_1 and a(left)

s_2, s_3

$$V_\pi(s_1) = 0.9 \sum_{s' \in S} p(s', r | s_1, \pi(s_1)) V_\pi(s')$$

$$V_\pi(s_1) = 0.9 * ((0.7 * 2) + (0.3 * 1))$$

$$V_\pi(s_1) = \boxed{\mathbf{1.53}}$$

For s_1 and a(right)

$$V_\pi(s_1) = 0.9 \sum_{s' \in S} p(s', r | s_1, \pi(s_1)) V_\pi(s')$$

$$V_\pi(s_1) = 0.9 * ((0.3 * 2) + (0.7 * 1))$$

$$V_\pi(s_1) = \mathbf{1.17}$$

Dynamic Programming (Policy Iteration)

Example

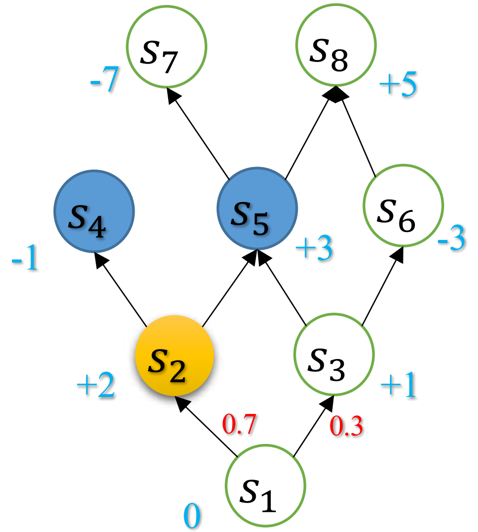
Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	**R**	R	-	R	R	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

$$\pi(s) = \boxed{argmax_a \gamma} \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$



2 Policy improvement steps

✓ **First** iteration for the **Policy Evaluation** s_2:

For s_2 and a(left)

$$V_\pi(s_2) = 0.9 \sum_{s' \in S} p(s', r | s_2, \pi(s_2)) V_\pi(s')$$

$$V_\pi(s_2) = 0.9 * ((0.7 * (-1)) + (0.3 * 3))$$

$$V_\pi(s_2) = 0.18$$

For s_2 and a(right)

$$V_\pi(s_2) = 0.9 \sum_{s' \in S} p(s', r | s_2, \pi(s_2)) V_\pi(s')$$

$$V_\pi(s_2) = 0.9 * ((0.3 * (-1)) + (0.7 * 3))$$

$$V_\pi(s_2) = \boxed{1.62}$$

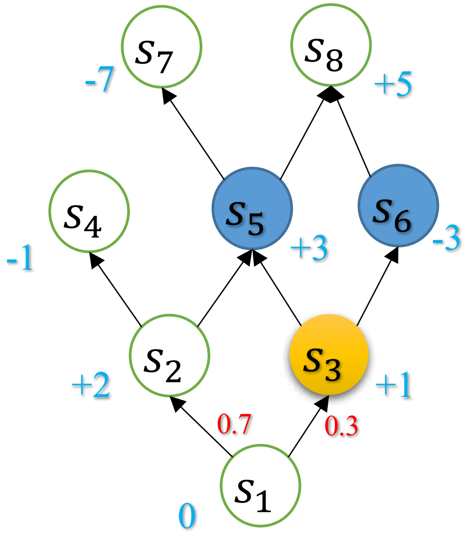
Dynamic Programming (Policy Iteration)

Example

Policy (π)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	L	R	L	-	R	R	-	-

Value: V_π	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	0	2	1	-1	3	-3	-7	5

$$\pi(s) = \boxed{argmax_a \gamma} \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$



2 Policy improvement steps

✓ **First** iteration for the **Policy Evaluation** s_3:

For s_3 and a(left)

$$V_\pi(s_3) = 0.9 \sum_{s' \in S} p(s', r | s_3, \pi(s_3)) V_\pi(s')$$

$$V_\pi(s_3) = 0.9 * ((0.7 * 3) + (0.3 * (-3)))$$

$$V_\pi(s_3) = \boxed{\mathbf{1.08}}$$

For s_3 and a(right)

$$V_\pi(s_3) = 0.9 \sum_{s' \in S} p(s', r | s_3, \pi(s_3)) V_\pi(s')$$

$$V_\pi(s_3) = 0.9 * ((0.3 * (3)) + (0.7 * (-3)))$$

$$V_\pi(s_3) = -1.89$$

Dynamic Programming (Policy Iteration)

Example

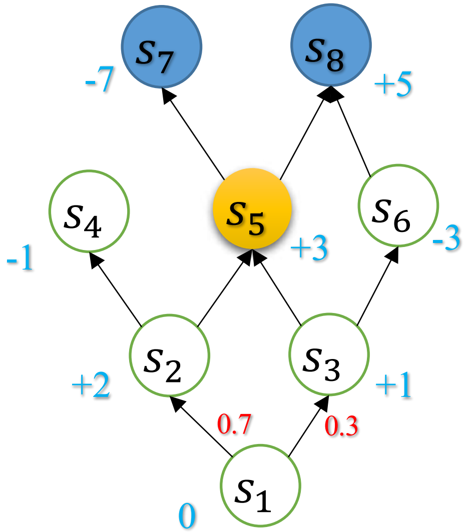
Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	R	L	-	R	R	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

$$\pi(s) = \boxed{argmax_a \gamma} \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$



2 Policy improvement steps

✓ **First** iteration for the **Policy Evaluation** s_5:

For s_5 and a(left)

$$V_\pi(s_5) = 0.9 \sum_{s' \in S} p(s', r | s_5, \pi(s_5)) V_\pi(s')$$

$$V_\pi(s_5) = 0.9 * ((0.7 \ * (-7)) + (0.3 * (5)))$$

$$V_\pi(s_5) = -3.6$$

For s_5 and a(right)

$$V_\pi(s_5) = 0.9 \sum_{s' \in S} p(s', r | s_5, \pi(s_5)) V_\pi(s')$$

$$V_\pi(s_5) = 0.9 * ((0.3 \ * (-7)) + (0.7 * (5)))$$

$$V_\pi(s_5) = \boxed{\mathbf{-0.54}}$$

Dynamic Programming (Policy Iteration)

Example

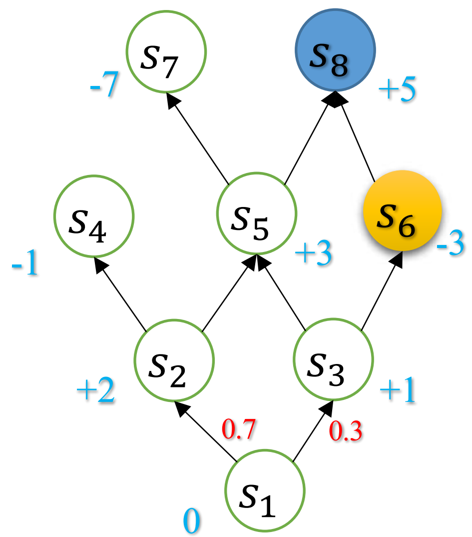
Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	R	L	-	R	L	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

$$\pi(s) = \boxed{argmax_a \gamma} \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$



2 Policy improvement steps

✓ **First** iteration for the **Policy Evaluation** s_6:

For s_6 and a(left)

$$V_\pi(s_6) = 0.9 \sum_{s' \in S} p(s', r | s_6, \pi(s_6)) V_\pi(s')$$

$$V_\pi(s_6) = 0.9 * ((1 * 5))$$

$$V_\pi(s_6) = \boxed{4.5}$$

Dynamic Programming (Policy Iteration)

Example

Policy (π)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	R	L	-	R	L	-	-

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

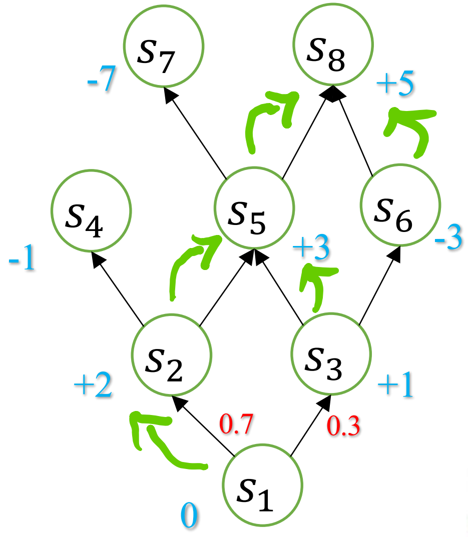
T[a(left)]=

	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s1$	0	0.7	0.3	0	0	0	0	0
$s2$	0	0	0	0.7	0.3	0	0	0
$s3$	0	0	0	0	0.7	0.3	0	0
$s4$	0	0	0	0	0	0	0	0
$s5$	0	0	0	0	0	0	0.7	0.3
$s6$	0	0	0	0	0	0	0	1
$s7$	0	0	0	0	0	0	0	0
$s8$	0	0	0	0	0	0	0	0

T[a(right)]=

	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s1$	0	0.3	0.7	0	0	0	0	0
$s2$	0	0	0	0.3	0.7	0	0	0
$s3$	0	0	0	0	0.3	0.7	0	0
$s4$	0	0	0	0	0	0	0	0
$s5$	0	0	0	0	0	0	0.3	0.7
$s6$	0	0	0	0	0	0	0	1
$s7$	0	0	0	0	0	0	0	0
$s8$	0	0	0	0	0	0	0	0

Visualize on the MDP
(only after one iteration)



Dynamic Programming (Policy Iteration)

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

2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad a \leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   $\quad$ If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Dynamic Programming (Value Iteration)

Value Iteration (based on MDPs)

- ✓ Another method to solve Bellman equation is called *value iteration* which assesses the value directly.
- ✓ Compute the **optimal state value function** by iteratively updating the estimate $V_{\pi}(s)$
- ✓ The value iteration algorithm updates the **state value function** in a **single step**.
- ✓ This is possible by calculating all possible rewards by looking ahead.
- ✓ The value iteration algorithm is also guaranteed to converge to the optimal values.

Dynamic Programming (Value Iteration)

Value Iteration (based on MDPs)

✓ Start with a random value function $V_\pi(s)$ and update it in each step:

Bellman's Equation

$$V_\pi(s) = \; r(s) + \gamma \max_a \sum_{s' \in S} p(s', r|s, a) V_\pi(s')$$

The idea is (similar to policy iteration but adding max):

- Take the **maximum** over all possible actions in the value iteration algorithm.

Dynamic Programming (Value Iteration)

Value Iteration (based on MDPs)

$$V_\pi(s) = \; r(s) + \gamma \max_a \sum_{s' \in S} p(s', r \mid s, a) V_\pi(s')$$

How?

For example for state s_1 of any arbitrary problem:

$$V_\pi(s_1) = \; r(s_1) + \gamma \max_a \left[\sum_{s' \in S} p(s', r \mid s_1, a) V_\pi(s')\right]$$

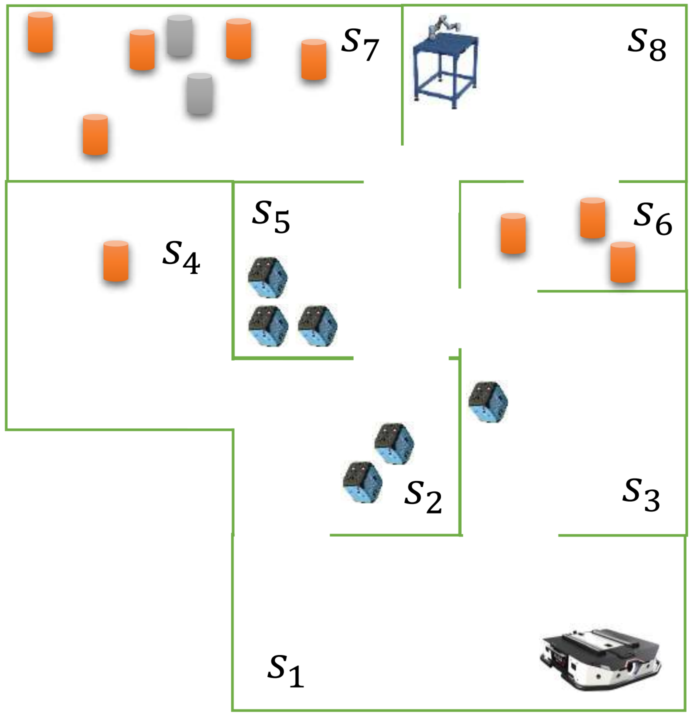
If we have two actions L and R:

$$V_\pi(s_1) = \; r(s_1) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r \mid s_1, \pi(s_1) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r \mid s_1, \pi(s_1) = R) V_\pi(s') \end{bmatrix}$$

Dynamic Programming (Value Iteration)

Value Iteration (based on MDPs)

Example

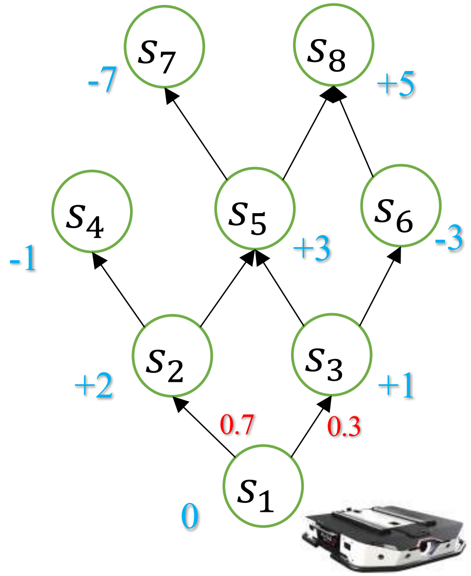


T[a(left)]=

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.7	0.3	0	0	0	0	0
s2	0	0	0	0.7	0.3	0	0	0
s3	0	0	0	0	0.7	0.3	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.7	0.3
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0

T[a(right)]=

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.3	0.7	0	0	0	0	0
s2	0	0	0	0.3	0.7	0	0	0
s3	0	0	0	0	0.3	0.7	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.3	0.7
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0



Value Iteration, Example

Update value

$$V_\pi(s) = \ r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

First iteration and s_1:

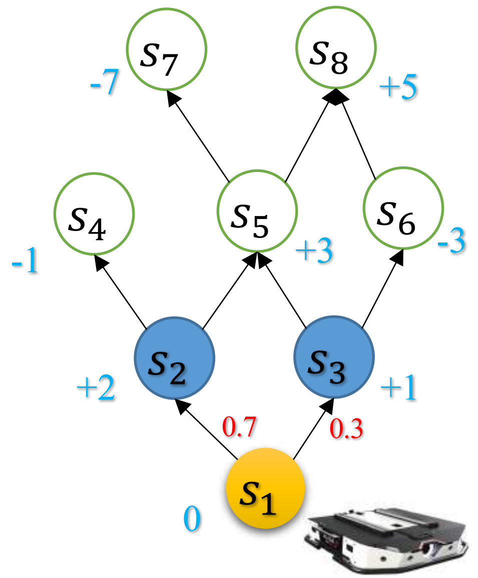
$$V_\pi(s_1) = \ r(s_1) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_1) = \ r(0) + 0.9 \ * \ \max_a \begin{bmatrix} 0.7 * 0 + 0.3 \ * 0 \\ 0.3 * 0 + 0.7 \ * 0 \end{bmatrix}$$

$$V_\pi(s_1) = 0$$

$$\text{T[a(left)]}= \begin{array}{c} \\ \left[\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ s1 & 0 & \boxed{0.7} & \boxed{0.3} & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \end{array}$$

$$\text{T[a(right)]}= \left[\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ s1 & 0 & \boxed{0.3} & \boxed{0.7} & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right]$$



Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	0	0	0	0	0	0	0

Value Iteration, Example

Update value

$$V_\pi(s) = \ r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

First iteration and s_2:

$$V_\pi(s_2) = \ r(s_2) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_2, \pi(s_2) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_2, \pi(s_2) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_2) = \ 2 + 0.9 \ * \ \max_a \begin{bmatrix} 0.7 * 0 + 0.3 \ * 0 \\ 0.3 * 0 + 0.7 \ * 0 \end{bmatrix}$$

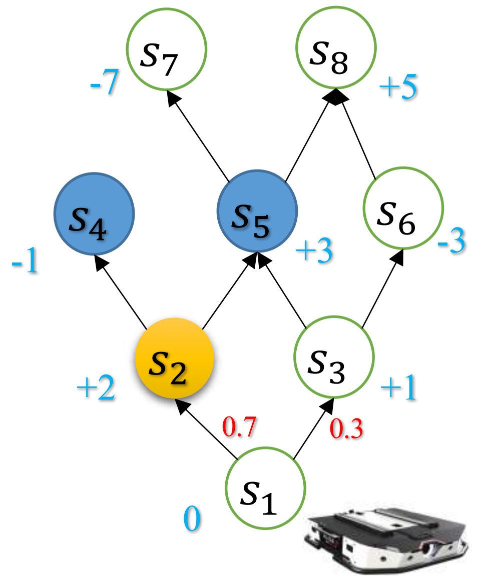
$$V_\pi(s_2) = 2$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	0	0	0	0	0	0

$$\text{T[a(left)]}= \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\text{T[a(right)]}= \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$



Value Iteration, Example

Update value

$$V_\pi(s) = \ r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

First iteration and **all *states***:

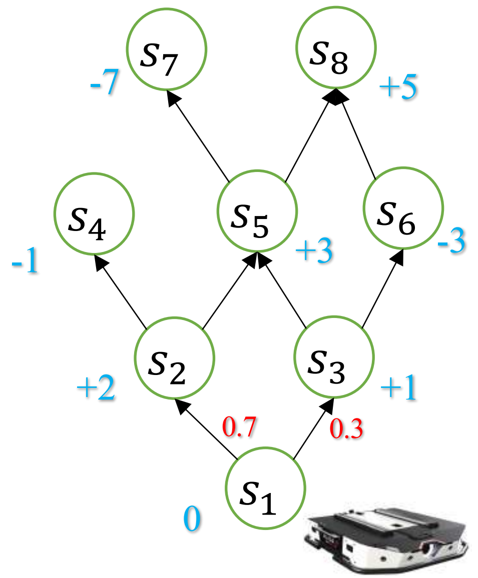
$$\text{T[a(left)]=}\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\text{T[a(right)]=}\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
0	2	1	-1	3	-3	-7	5

Continue same for all states



Value Iteration, Example

Update value

$$V_\pi(s) = \; r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

Second iteration and s_1:

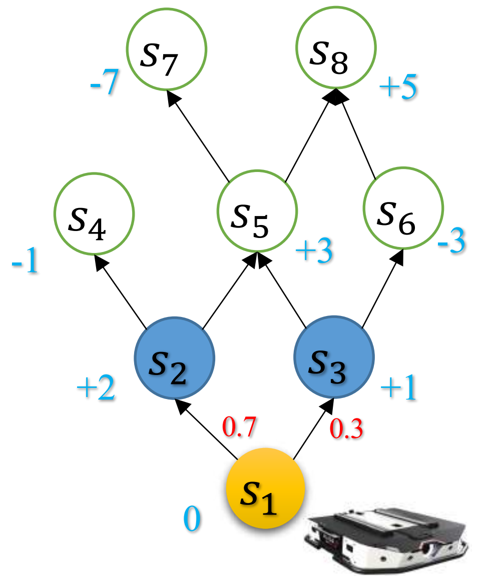
$$V_\pi(s_1) = \; r(s_1) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_1) = \; r(0) + 0.9 * \max_a \begin{bmatrix} 0.7 * \mathbf{2} + 0.3 * \mathbf{1} \\ \underbrace{0.3 * \mathbf{2}}_{s_2} + \underbrace{0.7 * \mathbf{1}}_{s_3} \end{bmatrix}$$

$$V_\pi(s_1) = 0 + 0.9 * \max_a \begin{bmatrix} 1.7 \\ 1.3 \end{bmatrix} = 1.53$$

$$\text{T[a(left)]} = \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\text{T[a(right)]} = \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$



Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	2	1	-1	3	-3	-7	5

Value Iteration, Example

Update value

$$V_\pi(s) = r(s) + \gamma \max_a \sum_{s' \in S} p(s', r \mid s, a) V_\pi(s')$$

T[a(left)]=

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.7	0.3	0	0	0	0	0
s2	0	0	0	0.7	0.3	0	0	0
s3	0	0	0	0	0.7	0.3	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.7	0.3
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0

T[a(right)]=

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.3	0.7	0	0	0	0	0
s2	0	0	0	0.3	0.7	0	0	0
s3	0	0	0	0	0.3	0.7	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.3	0.7
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0

Second iteration and s_2:

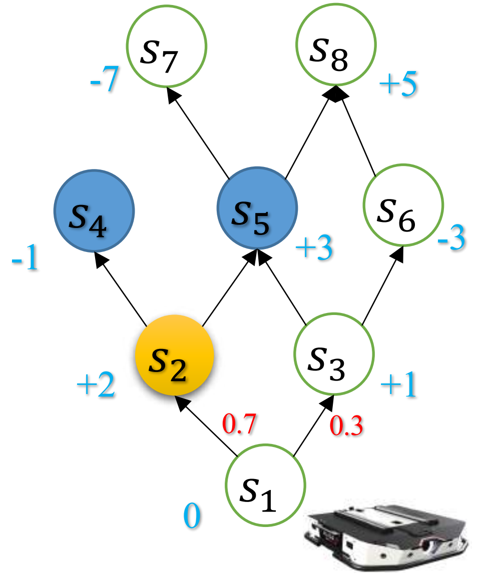
$$V_\pi(s_2) = r(s_2) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r \mid s_2, \pi(s_2) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r \mid s_2, \pi(s_2) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_2) = r(2) + 0.9 * \max_a \begin{bmatrix} 0.7 * -\mathbf{1} + 0.3 * \mathbf{3} \\ \underbrace{0.3 * -\mathbf{1}}_{s_4} + \underbrace{0.7 * \mathbf{3}}_{s_5} \end{bmatrix}$$

$$V_\pi(s_2) = 2 + 0.9 * \max_a \begin{bmatrix} 0.2 \\ 1.8 \end{bmatrix} = 3.62$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	**3.62**	1	-1	3	-3	-7	5



Value Iteration, Example

Update value

$$V_\pi(s) = r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

Second iteration and s_3:

$$V_\pi(s_3) = r(s_3) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_3, \pi(s_3) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_3, \pi(s_3) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_3) = r(1) + 0.9 * \max_a \begin{bmatrix} 0.7 * \mathbf{3} + 0.3 * -\mathbf{3} \\ \underbrace{0.3 * \mathbf{3}}_{s_5} + \underbrace{0.7 * -\mathbf{3}}_{s_6} \end{bmatrix}$$

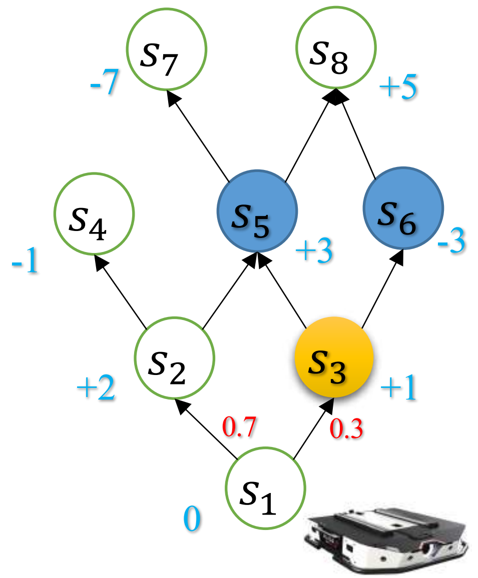
$$V_\pi(s_3) = 1 + 0.9 * \max_a \begin{bmatrix} 1.2 \\ -1.2 \end{bmatrix} = 2.08$$

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	3.62	**2.08**	-1	3	-3	-7	5

$$\text{T[a(left)]=} \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boxed{0.7} & \boxed{0.3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\text{T[a(right)]=} \begin{array}{c} \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \\ s8 \end{array} \begin{array}{cccccccc} s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boxed{0.3} & \boxed{0.7} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$



Value Iteration, Example

Update value

$$V_\pi(s) = \; r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

$$\text{T[a(left)]=}\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0} \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\text{T[a(right)]=}\begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0} \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

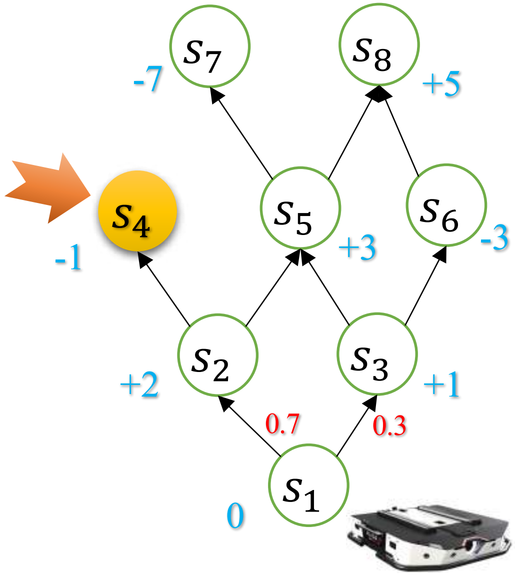
Second iteration and $\boldsymbol{s_4}$:

$$V_\pi(s_4) = \; r(s_4) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_4, \pi(s_4) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_4, \pi(s_4) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_4) = \; -1 + 0.9 * \max_a \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$V_\pi(s_4) = -1$$

We don't need to run terminal states



Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	3.62	2.08	**-1**	3	-3	-7	5

Value Iteration, Example

Update value

$$V_\pi(s) = \; r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

T[a(left)]=

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.7	0.3	0	0	0	0	0
s2	0	0	0	0.7	0.3	0	0	0
s3	0	0	0	0	0.7	0.3	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.7	0.3
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0

T[a(right)]=

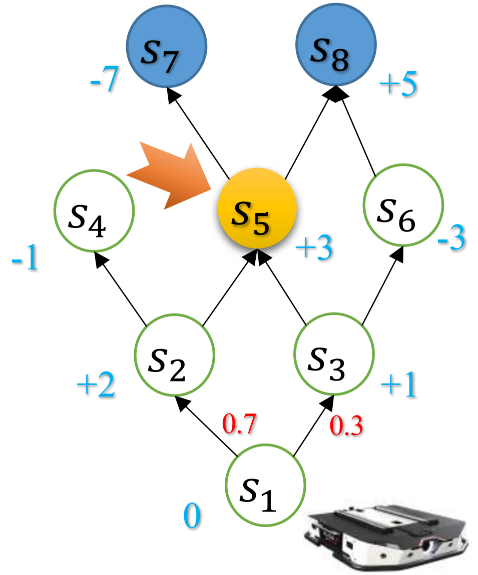
	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	0.3	0.7	0	0	0	0	0
s2	0	0	0	0.3	0.7	0	0	0
s3	0	0	0	0	0.3	0.7	0	0
s4	0	0	0	0	0	0	0	0
s5	0	0	0	0	0	0	0.3	0.7
s6	0	0	0	0	0	0	0	1
s7	0	0	0	0	0	0	0	0
s8	0	0	0	0	0	0	0	0

Second iteration and s_5:

$$V_\pi(s_5) = \; r(s_5) + \gamma \max_a \begin{bmatrix} \sum_{s' \in S} p(s', r | s_5, \pi(s_5) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_5, \pi(s_5) = R) V_\pi(s') \end{bmatrix}$$

$$V_\pi(s_5) = \; +3 + 0.9 \; * \; \max_a \begin{bmatrix} 0.7 * -\mathbf{7} + 0.3 \; * 5 \\ 0.3 * \underbrace{-\mathbf{7}}_{s_7} + \underbrace{0.7 \; * \mathbf{5}}_{s_8} \end{bmatrix}$$

$$V_\pi(s_5) = +3 + 0.9 \; * \; \max_a \begin{bmatrix} -3.4 \\ 1.4 \end{bmatrix} = 4.26$$



Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	3.62	2.08	-1	**4.26**	-3	-7	5

Value Iteration, Example

Update value

$$T[a(left)] = \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$T[a(right)] = \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

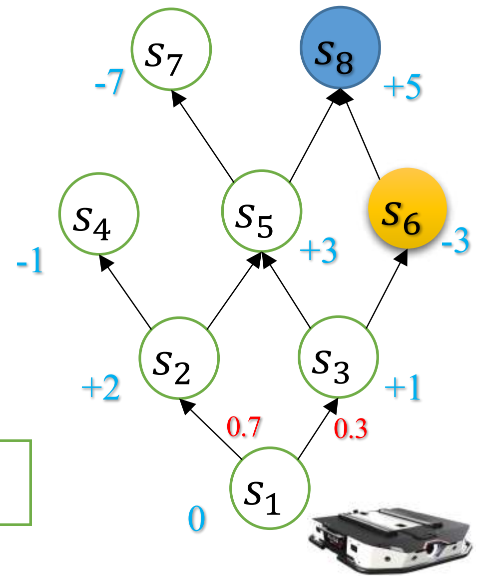
$$V_\pi(s) = r(s) + \gamma \max_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

Second iteration and s_6:

$$V_\pi(s_6) = r(s_6) + \gamma \max_a \left[\sum_{s' \in S} p(s', r | s_6, \pi(s_6) = L) V_\pi(s') \right]$$

$$V_\pi(s_6) = -3 + 0.9 * \max_a [1 * \underbrace{5}_{s_8}]$$

$$V_\pi(s_6) = 1.5$$



Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	3.62	2.08	-1	4.26	**1.5**	-7	5

Updates will not change them since they are terminal state

Value Iteration, Example

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1.53	3.62	2.08	-1	4.26	**1.5**	-7	5

✓ Convergence

- ✓ Continue same with next iteration until converge!

Bellman Factor

- ✓ In practice, we stop once the value function changes by only a small amount of change, e.g. **Bellman Factor**, e.g. $\delta = 0.01$

After convergence value for our example:

Value: V_π^*

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
3.88	**4.41**	**4.09**	**-1**	**4.26**	**1.5**	**-7**	**5**

Value Iteration, Example

Convergence notes:

- ✓State space and action space should be finite
- ✓Reward values should have an upper and lower bound
- ✓Environment should be episodic (*if continuous* the discount factor should be less than 1)

Value Iteration, Example

Value Iteration – Second Step

- ✓ Second step is **getting optimal action** to construct the **optimal policy**.
- ✓ **Only** **one iteration** to return **the maximum action sequence:**

$$\pi_t^* = argmax_a r(s) + \gamma \sum_{s' \in S} p(s'|s,a) V_\pi(s')$$

It determines witch action gives us the highest value

Value Iteration, Example

Policy Update

Value: V_π	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	3.88	4.41	4.09	-1	4.26	1.5	-7	5

$$\pi_t^* = argmax_a r(s) + \gamma \sum_{s' \in S} p(s'|s,a) V_\pi(s')$$

$$T[a(left)] = \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0.7} & \boxed{0.3} \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$T[a(right)] = \begin{array}{c|cccccccc} & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\ \hline s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s5 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0.3} & \boxed{0.7} \\ s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

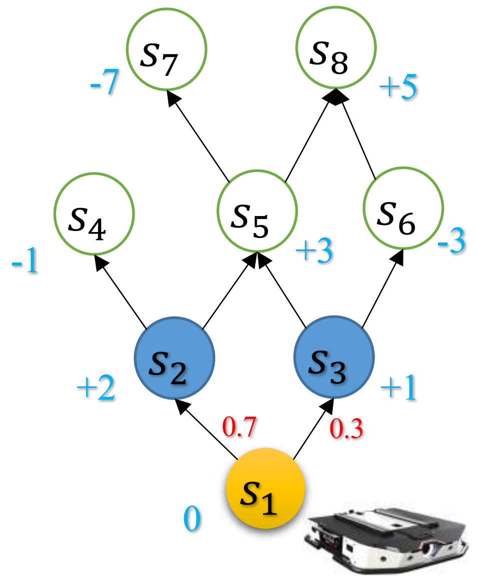
Second iteration and s_1:

$$\pi_t^*(s_1) = \text{argmax}_a r(s) + \gamma \begin{bmatrix} \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = L) V_\pi(s') \\ \sum_{s' \in S} p(s', r | s_1, \pi(s_1) = R) V_\pi(s') \end{bmatrix}$$

$$\pi_t^*(s_1) = \text{argmax}_a r(s) + 0.9 * \begin{bmatrix} 0.7 * \mathbf{4.41} + 0.3 * \mathbf{4.09} \\ 0.3 * \underbrace{\mathbf{4.41}}_{s_2} + 0.7 * \underbrace{\mathbf{4.09}}_{s_3} \end{bmatrix}$$

$$\pi_t^*(s_1) = \text{argmax}_a 0 + 0.9 * \begin{bmatrix} 4.314 \\ 4.186 \end{bmatrix} = \text{argmax}_a (\overset{L}{\boxed{\mathbf{3.88}}}, \overset{R}{3.76})$$

Policy (π_t^*)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	L	R	R	-	R	R	-	-



Value Iteration, Example

Example

$$
\text{T[a(left)]}=\begin{array}{c|cccccccc}
 & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\
\hline
s1 & 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 \\
s2 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 & 0 \\
s3 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0 & 0 \\
s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s5 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0.7} & \boxed{0.3} \\
s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

$$
\text{T[a(right)]}=\begin{array}{c|cccccccc}
 & s1 & s2 & s3 & s4 & s5 & s6 & s7 & s8 \\
\hline
s1 & 0 & 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 \\
s2 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\
s3 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\
s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s5 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{0.3} & \boxed{0.7} \\
s6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
s7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

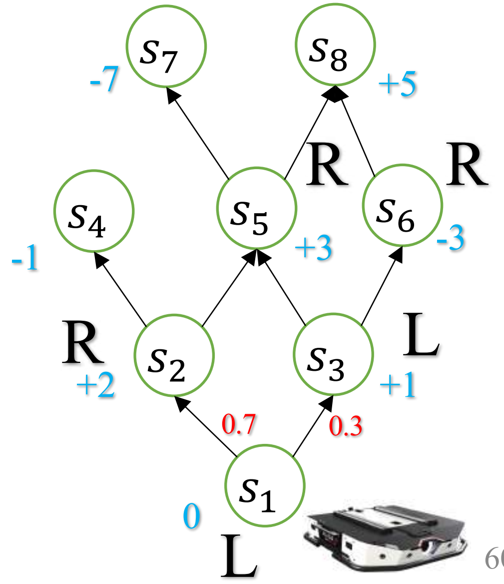
Policy update's final result:

Value: V_π

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
3.88	4.41	4.09	-1	4.26	1.5	-7	5

Policy (π_t^*)

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
L	R	L	-	R	L	-	-



Dynamic Programming (Value Iteration)

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

Dynamic Programming (Value Iteration)

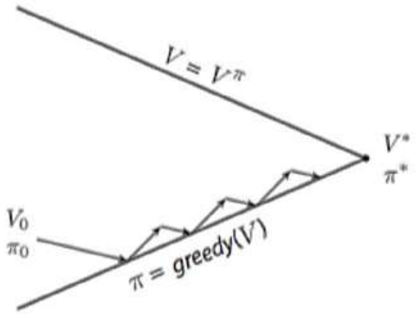
Value Iteration VS **Policy Iteration**

Value Iteration

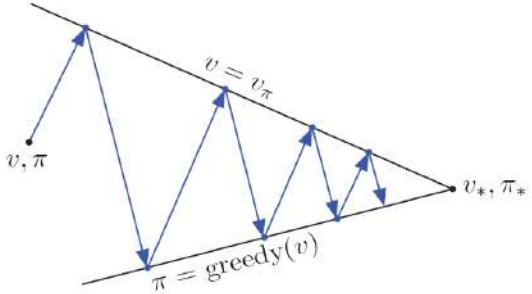
- ✓ The value iteration algorithm converges with **more iterations**
- ✓ Slower
- ✓ **Starts with a random value function** at the beginning
- ✓ Simpler algorithm $O(|A||S|^2)$

Policy Iteration

- ✓ The policy iteration algorithm converges with **lower iterations**
- ✓ Faster
- ✓ **Starts with a random policy** at the beginning
- ✓ More Complex $O(|A||S|^2 + |S|^3)$



Both algorithms are guaranteed to converge to an optimal policy



Dynamic Programming (Modified Policy Iteration)

Modified Policy Iteration

- ✓ In order to use benefits of both value iteration and policy iteration algorithms we have **modified policy iteration algorithm**

- ✓ The idea is to improve the policy update loop part with k times (limited number of times) update.

Dynamic Programming (Policy Iteration)

Modified Policy Iteration

1. Repeat K times Policy evaluation step

$$V_\pi(s) = r(s) + \gamma \sum_{s' \in S} p(s', r | s, \pi(s)) V_\pi(s')$$

2. Policy improvement step

$$\pi(s) = argmax_a \sum_{s' \in S} p(s', r | s, a) V_\pi(s')$$

Until converge to optimal ($V_\pi^*(s)$, and $\pi^*(s)$)

Repeat K times

Evaluation $V_\pi(s)$ $\longrightarrow$ Improvement $\pi(s)$

Dynamic Programming (Value Iteration)

Value Iteration

- ✓ Simpler algorithm $O(|A||S|^2)$

Policy Iteration

- ✓ Per-iteration costs $O(|A||S|^2 + |S|^3)$

Modified Policy Iteration

- ✓ Per-iteration costs $O(|A||S|^2 + k|S|^2)$

✓ Modified Policy Iteration usually performs **faster** than both value iteration and Policy iteration algorithms

Assignment

Extend the given example code (Value Iteration) to Implement and compare algorithms:
Value Iteration, Policy Iteration, and Modified Policy Iteration algorithms.

A) Compare (plot convergence speed results) of the example problem in the slides

B) Extend the example in the slides to add more actions (robot can execute move backward action too) and run the same algorithms to plot convergence.

Dynamic Programming

Finite vs Infinite episodes or Horizon

- ✓ In **infinite horizon** we can have stationary optimal policy because always end is not clear

- ✓ In **finite horizon** the optimal policy can be non-stationary optimal policy since as close as we get to end the actions that are selected can be different

Challenge: For infinite horizon in value iteration we cannot calculate infinite steps