CMPSC 448: Machine Learning

Lecture 18. Dynamic Programming for Markov Decision Processes

Rui Zhang

Fall 2021

PennState

Outline

- Introduction to Reinforcement learning
- Multi-armed Bandits
- **Markov Decision Processes (MDP)**
  - **Dynamic Programming when we know the world**
- Learning in MDP: When we don't know the world
  - Monte Carlo Methods
  - Temporal-Difference Learning (TD): SARSA and Q-Learning

Note: All of the lectures are tabular methods; we will only briefly discuss the motivation function approximation methods (e.g., DQN, policy gradient, deep reinforcement learning)

Markov Decision Process

A Markov decision process (MDP) is a Markov Reward Process with decisions.

If state, action, reward sets are finite, it is a finite MDP.

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$

$\mathcal{S}$ is a finite set of states

$\mathcal{A}$ is a finite set of actions

p is a state transition probability matrix for each action a

$$p(s, a, s') = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

r is a reward function $r(s, a) = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$

γ is a discount factor $\gamma \in [0, 1]$

To define a finite MDP, you need to give:

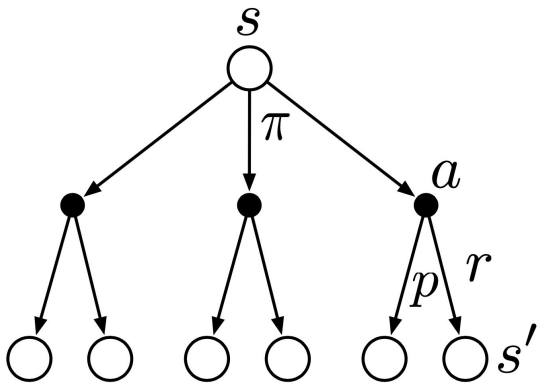
- state set, action set, discount factor
- one-step "dynamics"

Bellman Equation and Backup Diagram for $v_\pi(s)$

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big]$$

This is a set of equations (in fact, linear), one for each state. The value function for is its unique solution.

Backup diagram



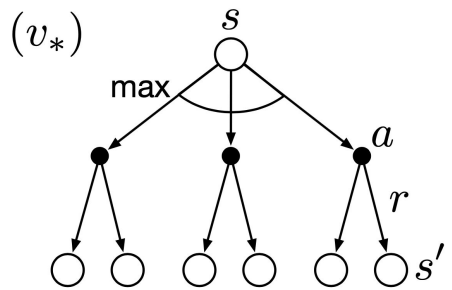
Backup diagram for v_π

Bellman Optimality Equation and Backup Diagram for v_*

The value of a state under an optimal policy must equal the expected return for the best action from that state:

$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$

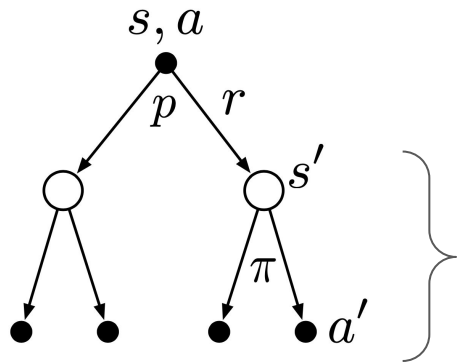
Backup diagram



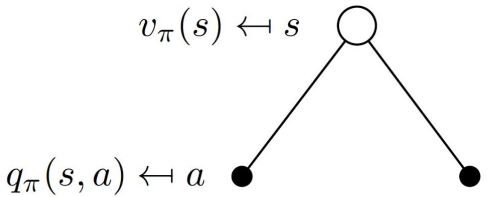
v_* is the unique solution of this system of nonlinear equations

Bellman Equation and Backup Diagram for $q_\pi(s, a)$

$$
\begin{aligned}
q_\pi(s, a) &= \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \\
&= \sum_{s', r} p(s', r | s, a)[r + \gamma v_\pi(s')] \\
&= \sum_{s', r} p(s', r | s, a)[r + \gamma \sum_{a'} \pi(a' | s') q_\pi(s', a')]
\end{aligned}
$$



q_π backup diagram

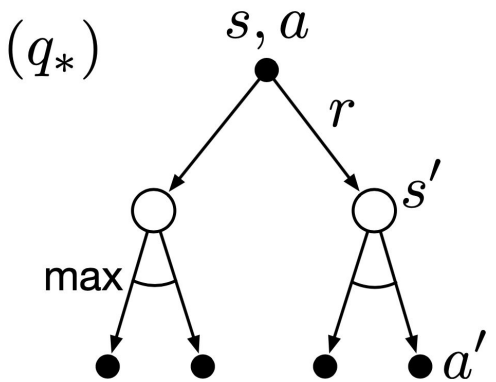


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

Bellman Optimality Equation and Backup Diagram for q_*

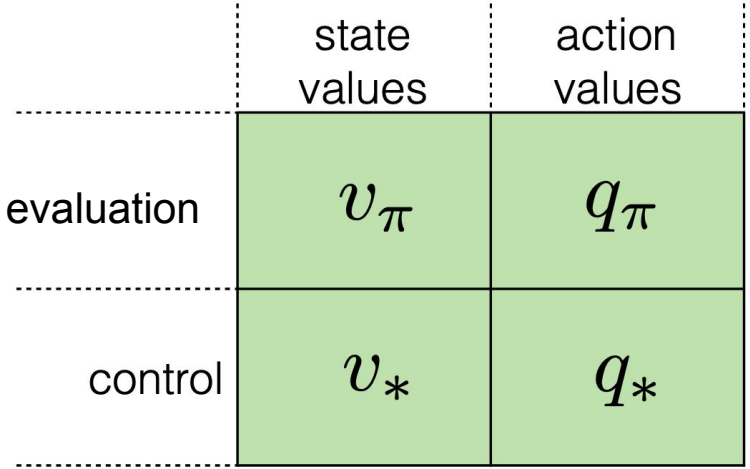
$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r|s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right].
\end{aligned}
$$

Backup diagram



q_* is the unique solution of this system of nonlinear equations

4 value functions



All theoretical objects, mathematical ideals (expected values)

Two problems in MDP

Input: a perfect model of RL as a MDP

$$\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$$

where its state, action, and reward sets are finite

Two problems:

1. **evaluation** (**prediction**): given a policy π , what is value function $v_\pi(s)$?
2. **control**: find the optimal policy π_* or optimal value functions, i.e., v_* or q_*

You can think that in order to solve problem 2, we must first know how to solve problem 1.

Solution 1: Write out Bellman Equations and Solve them

Solve systems of equations

- Write Bellman Equations for all state and state-action pairs
- Solve Linear Systems of Equations for Evaluation (i.e., compute v_π and q_π)
- Solve Non-linear Systems of Equations for Control (i.e., compute v_* and q_*)

Solution 2: Dynamic Programming for MDP

Idea: use dynamic programming on bellman equation for value functions to organize and structure the search

Dynamic programming in context of MDP/RL, refers to collection of algorithms to compute optimal policies given a perfect model of the environment as a Markov Decision Process (MDP)

$$\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$$

Note that we know all the information about MDP (states, rewards, transition probabilities, etc)

Outline

We introduce two DP methods to find an optimal policy for a given MDP:

Policy Iteration

- Policy Evaluation
- Policy Improvement

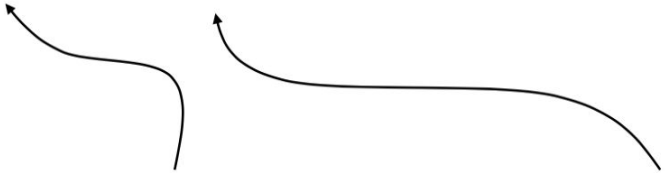
Value Iteration

- One-sweep Policy Evaluation + One-step Policy Improvement

Both methods rely on Bellman condition on the optimality of a policy (on the previous slides): The value of a state under an optimal policy must equal the expected return for the best action from that state

Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$



E: policy evaluation **I**: policy improvement

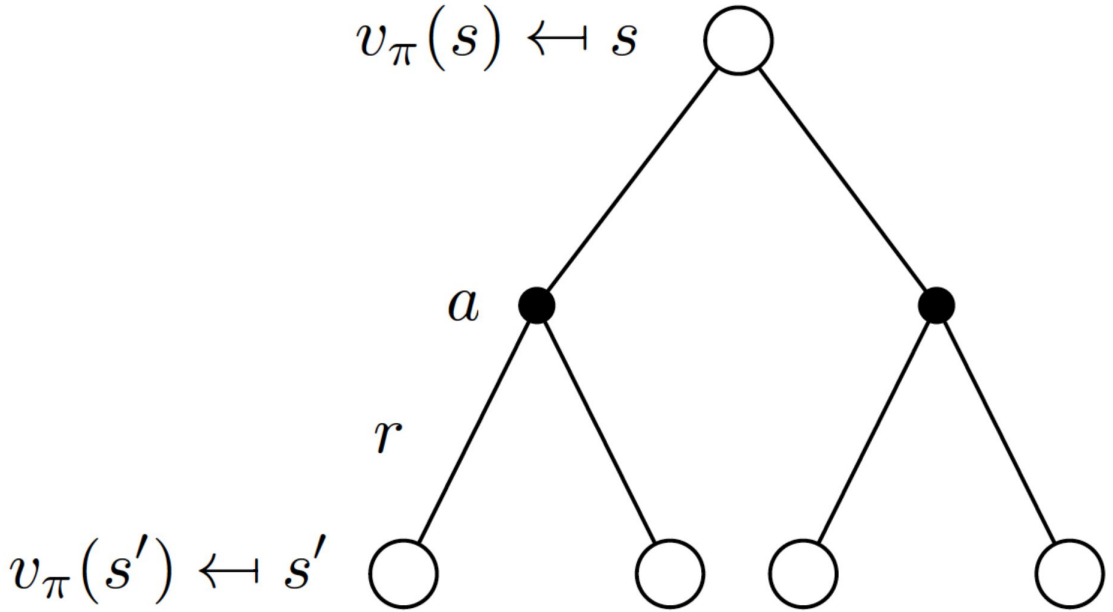
"**greedification**"

Policy Evaluation: Estimate v_π (iterative policy evaluation)

Policy Improvement: Generate $\pi' \geq \pi$ (Greedy policy improvement)

Policy Evaluation

Policy Evaluation: for a given arbitrary policy π compute the state-value function v_π



Policy Evaluation by Solving A system of Linear Equations

Recall: state-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Recall: Bellman equation for v_π

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

A system of $|\mathcal{S}|$ simultaneous equations $(v_\pi(s) \text{ for } s \in \mathcal{S})$

Note: environment dynamics $p(s', r|s, a)$ are completely known

Bellman Expectation Backup Operator

Recall **Bellman Equation** for $v_\pi(s)$

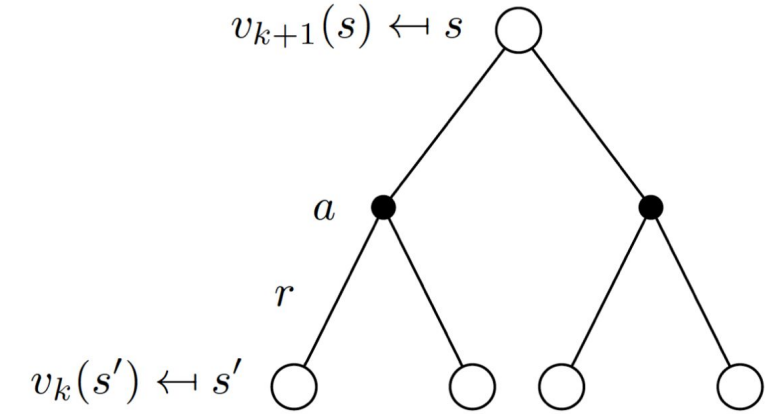
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big]$$

From this, let's define **Bellman Expectation Backup Operator** on $v_\pi(s)$

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a) \Big[r + \gamma v_k(s') \Big] \end{aligned}$$

Iterative Policy Evaluation by Bellman Expectation Backup Operator

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



$$v_0 = \mathbf{0}$$

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

$$= \sum_a \pi(a|s) \left(\sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \right)$$

value of state in next iteration

value of state in previous iteration

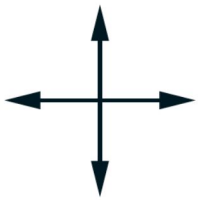
Iterative Policy Evaluation

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
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 $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Example: a small GridWorld



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

An un-discounted episodic MDP with $\gamma = 1$

Actions = {up, down, right, left}

Nonterminal states: {1, 2, . . ., 14}

One terminal state (shown twice as shaded squares)

Actions that would take agent off the grid leave state unchanged

Reward is –1 until the terminal state is reached

Iterative Policy Evaluation for the small GridWorld

$\pi =$ equiprobable random action choices

v_k for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Iterative Policy Evaluation for the small GridWorld

$\pi =$ equiprobable random action choices

v_k for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

$$= \sum_a \pi(a|s) \left(\sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \right)$$

Iterative Policy Evaluation for the small GridWorld

$\pi =$ equiprobable random action choices

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

$$= \sum_a \pi(a|s) \left(\sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \right)$$

Iterative Policy Evaluation for the small GridWorld

$\pi =$ equiprobable random action choices

actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

v_π

The final estimate is in fact v_π, which is this case gives, for each state, the negative of the expected number of steps from that state until termination

Policy Improvement

Suppose we have computed v_π for a deterministic policy π

For a given state s, $v_\pi(s)$ tells us how good it is to follow π

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

Policy Improvement

Suppose we have computed v_π for a deterministic policy π

For a given state s, $v_\pi(s)$ tells us how good it is to follow π

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

Let's take action a and thereafter follow the policy π and see what happens to the agent's reward, this is just $q_\pi(s, a)$

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

$$q_\pi(s, a) > v_\pi(s)$$

Policy Improvement Theorem

Let π and π' be any pair of deterministic policies, such that for all $s \in \mathcal{S}$, we have:

$$q_\pi\left(s, \pi'(s)\right) \geq v_\pi(s)$$

Then the policy π' must be as good as, or better than π.

The theorem can be easily generalized to stochastic policies (actions are selected by different probabilities at every states under policy, which is more realistic)

Policy Improvement Theorem - Proof

$$
\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \;\middle|\; S_t = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \;\middle|\; S_t = s\right] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \;\middle|\; S_t = s\right] \\
&= v_{\pi'}(s).
\end{aligned}
$$

Policy improvement with greedification

Do this for all states to get a new policy $\pi' \geq \pi$ that is greedy with respect to v_π

$$\begin{aligned}
\pi'(s) &= \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}$$

Policy improvement with greedification

Do this for all states to get a new policy $\pi' \geq \pi$ that is greedy with respect to v_π

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],\end{aligned}$$

Note that $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a) \leq \max_a q_\pi(s,a) = q_\pi(s,\pi'(s))$ then from policy improvement theorem, we have $\pi' \geq \pi$

Policy improvement with greedification

Do this for all states to get a new policy $\pi' \geq \pi$ that is greedy with respect to v_π

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],\end{aligned}$$

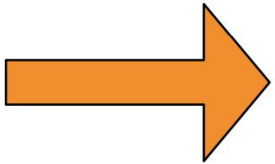
Note that $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a) \leq \max_a q_\pi(s,a) = q_\pi(s,\pi'(s))$ then from policy improvement theorem, we have $\pi' \geq \pi$

What if the policy is unchanged by this? then the policy satisfies Bellman Optimality Equation, and it must be the optimal policy!

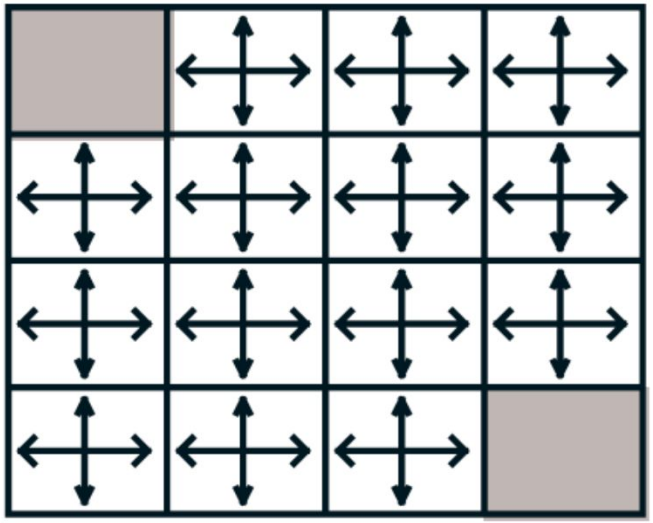
Example of Greedification

v_π

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



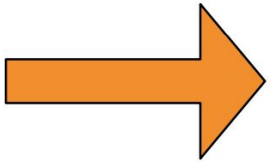
Greedy policy by greedification w.r.t. v_π



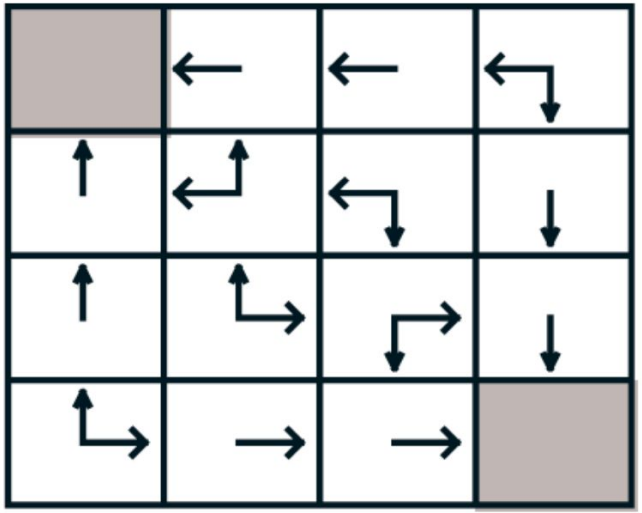
Example of Greedification

v_π

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



Greedy policy by greedification w.r.t. v_π



Policy Iteration: Iterate between Evaluation and Improvement

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

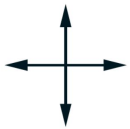
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

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

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

Iterative Policy Eval for the Small GridWorld

π = equiprobable random action choices



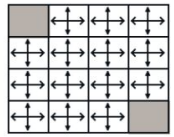
actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

$\gamma = 1$

An un-discounted episodic MDP with
Actions = {up, down, right, left}
Nonterminal states: {1, 2, . . ., 14}
One terminal state (shown twice as shaded squares)
Actions that would take agent off the grid leave state unchanged
Reward is –1 until the terminal state is reached

	V_k for the Random Policy	Greedy Policy w.r.t. V_k
$k = 0$	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	 random policy
$k = 1$	0.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 0.0	
$k = 2$	0.0 -1.7 -2.0 -2.0 -1.7 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -1.7 -2.0 -2.0 -1.7 0.0	
$k = 3$	0.0 -2.4 -2.9 -3.0 -2.4 -2.9 -3.0 -2.9 -2.9 -3.0 -2.9 -2.4 -3.0 -2.9 -2.4 0.0	
$k = 10$	0.0 -6.1 -8.4 -9.0 -6.1 -7.7 -8.4 -8.4 -8.4 -8.4 -7.7 -6.1 -9.0 -8.4 -6.1 0.0	
$k = \infty$	0.0 -14. -20. -22. -14. -18. -20. -20. -20. -20. -18. -14. -22. -20. -14. 0.0	

Iterative Policy Eval for the Small GridWorld

$\pi =$ equiprobable random action choices

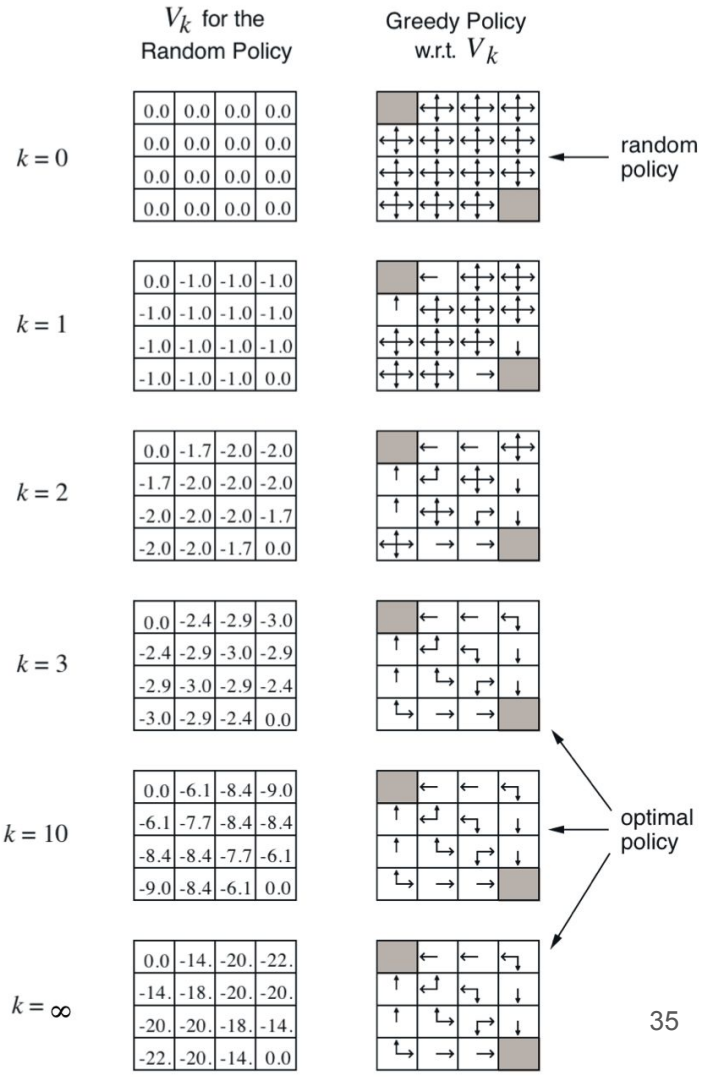
actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

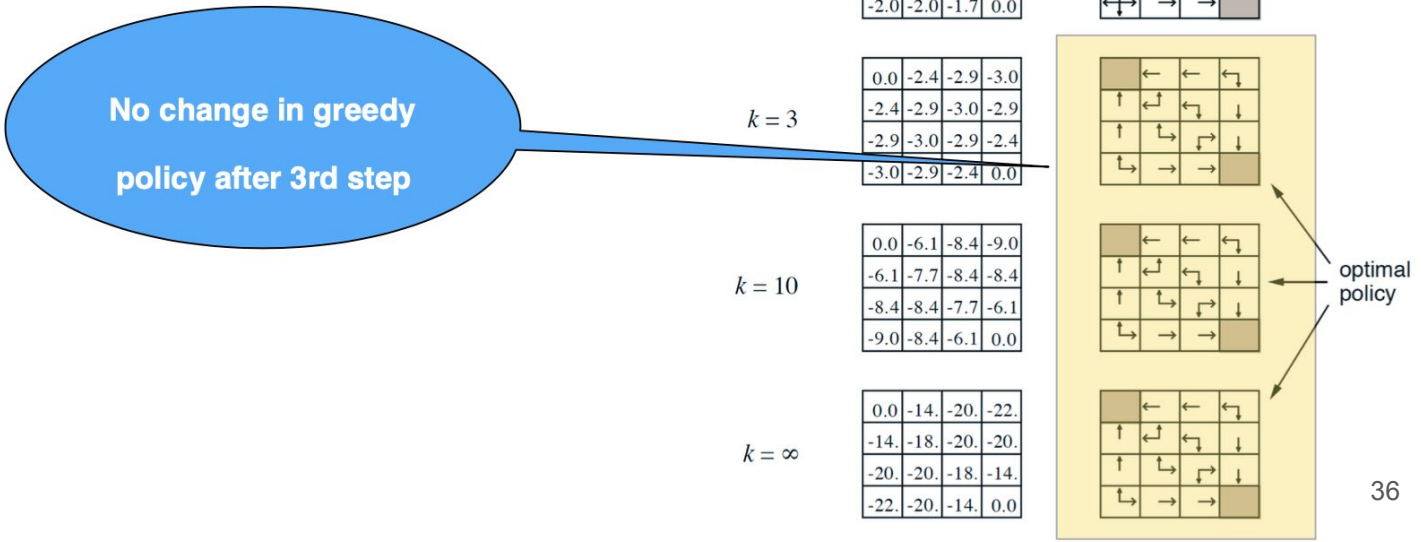
$\gamma = 1$

An un-discounted episodic MDP with
Actions = {up, down, right, left}
Nonterminal states: {1, 2, . . ., 14}
One terminal state (shown twice as shaded squares)
Actions that would take agent off the grid leave state unchanged
Reward is –1 until the terminal state is reached



From Policy Iteration to Value Iteration

Do we need to do policy evaluation until convergence before greedification or it could be truncated somehow?



From Policy Iteration to Value Iteration

Recall Policy Iteration alternates between the following two steps:

1. Policy Evaluation: Multiple Sweeps of Bellman Expectation Backup Operation until Convergence

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

$$= \sum_a \pi(a|s) \left(\sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \right)$$

2. Policy Improvement: One step of greedification

$$\begin{aligned} \pi'(s) &= \arg\max_a q_\pi(s, a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \end{aligned}$$

From Policy Iteration to Value Iteration

But, we don't need to do policy evaluation until convergence.

Instead, in Value Iteration:

1. Just One Sweep of Bellman Expectation Backup Operation

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

$$= \sum_a \pi(a|s) \left(\sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s')\right] \right)$$

2. One step of greedification

$$\begin{aligned} \pi'(s) &= \arg\max_a q_\pi(s, a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s')\Big], \end{aligned}$$

Value Iteration

Let’s interleave the evaluation and greedification

ONE sweep of evaluation is followed by ONE step of greedification

Combine these two together gives one update of value iteration as following

$$
\begin{aligned}
v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma v_k(s') \Big],
\end{aligned}
$$

We call this **Bellman Optimality Backup Operator**

In this way, we don't need to explicitly maintain a policy

Bellman Optimality Backup Operator

Bellman Optimality Equation for $v_*(s)$

$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$

Bellman Optimality Backup Operator on $v_\pi(s)$

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big], \end{aligned}$$

Value Iteration

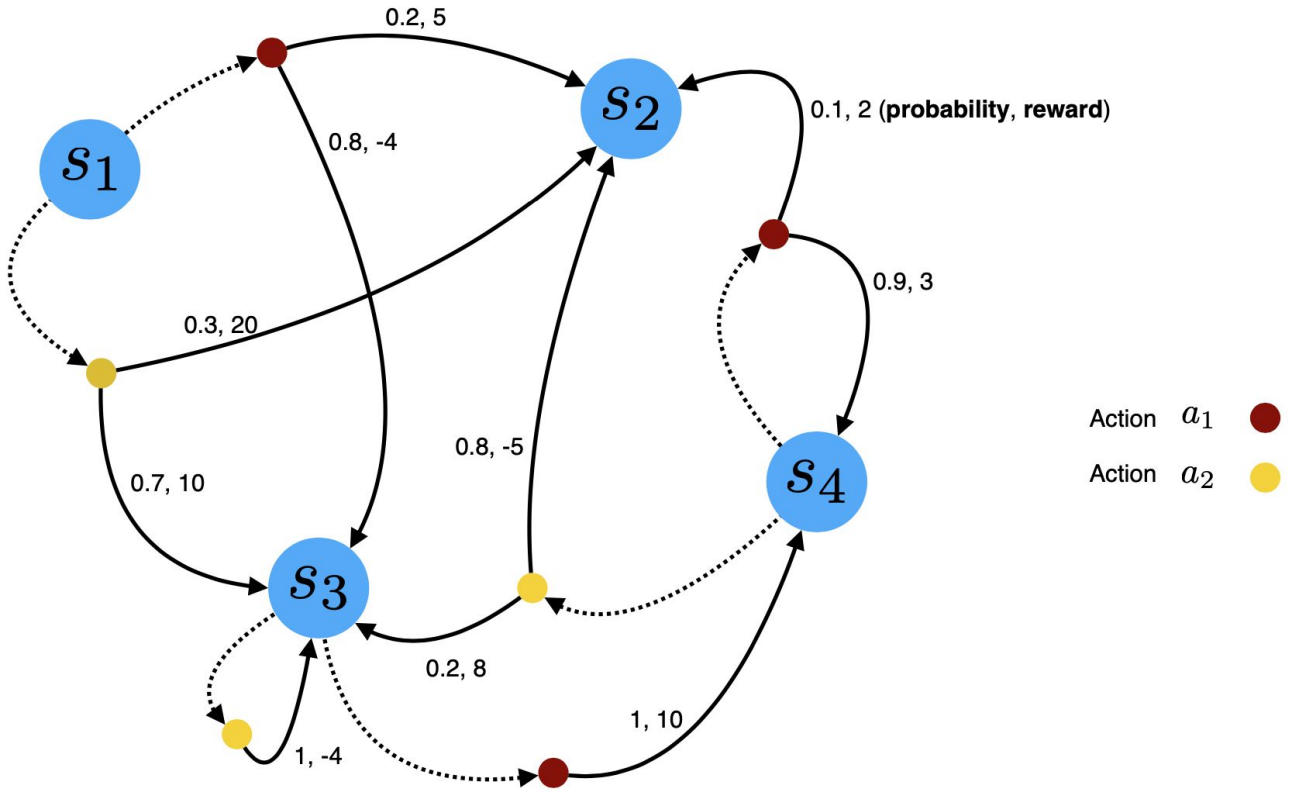
Value Iteration, for estimating $\pi \approx \pi_*$

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 $\Delta < \theta$

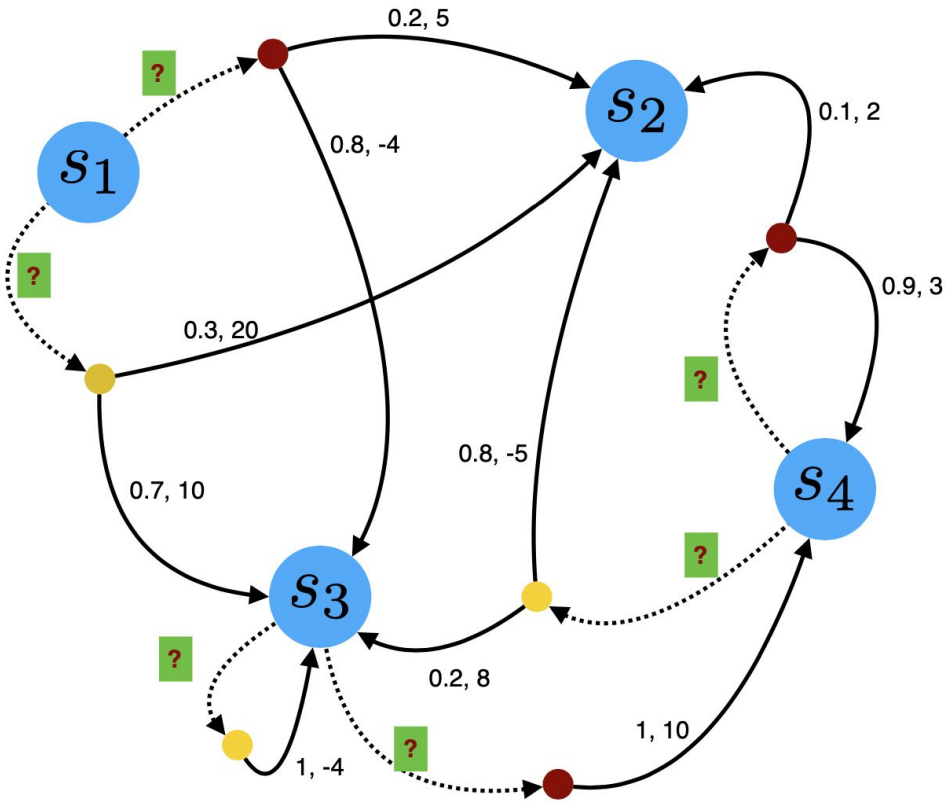
Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$

An example MDP



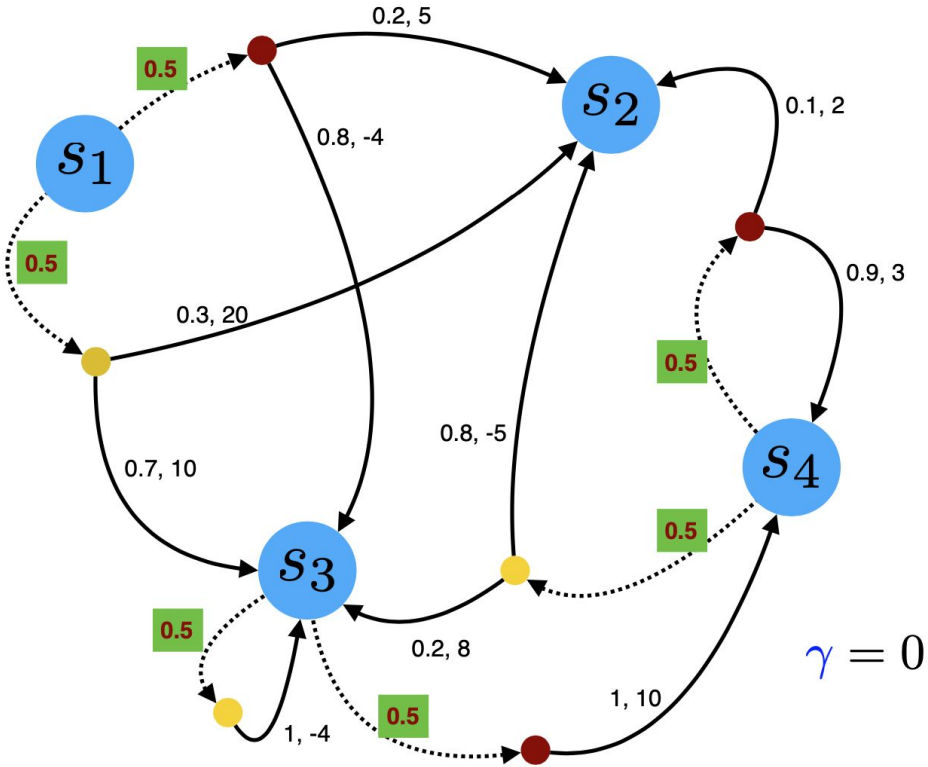
The dynamics of MDP is given: $p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$

An example MDP



Our goal is to learn a policy (mapping of states to actions or assignment of probabilities to actions at every state) such that if we follow the advices of the policy, we maximize the cumulative reward!

value iteration: initialization



$\gamma = 0.8$

$$v_{k+1}(s) \doteq \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k \left(S_{t+1} \right) | S_t = s \right]$$

$$= \sum_a \pi(a|s) \sum_{s', r} p\left(s', r | s, a\right) \left[r + \gamma v_k \left(s' \right) \right]$$

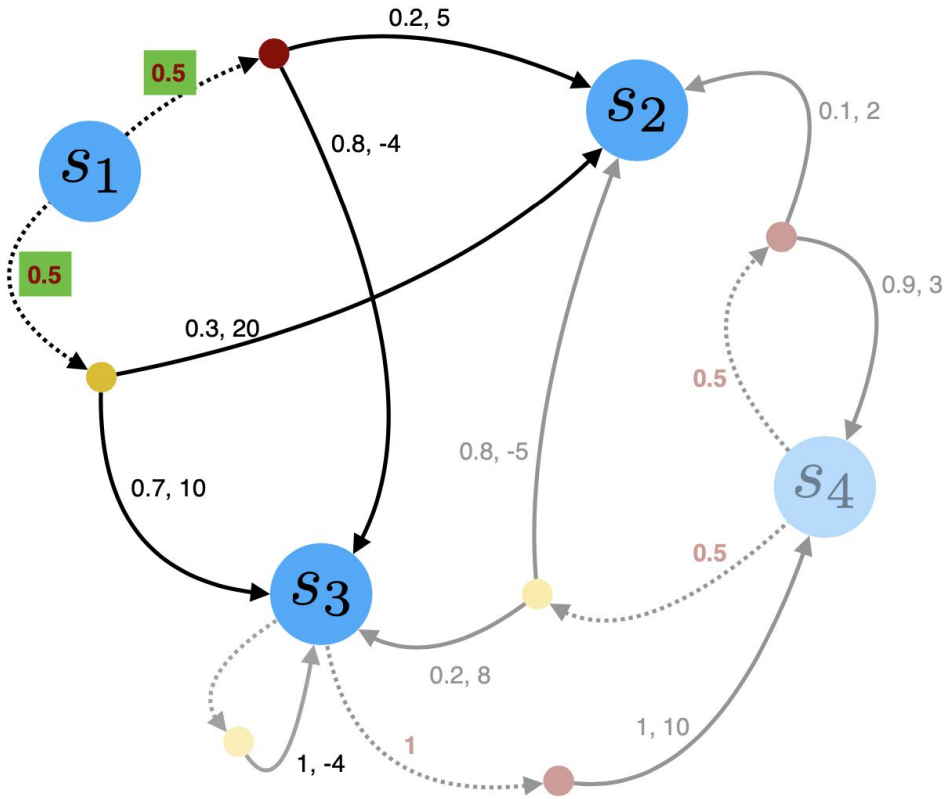
s_1	0.5	0.5
s_2	0.5	0.5
s_3	0.5	0.5
s_4	0.5	0.5

π_0

State	Value
s_1	0
s_2	0
s_3	0
s_4	0

$v_0(s)$

value iteration: one sweep evaluation



s_1	0.5	0.5
s_2	0.5	0.5
s_3	0.5	0.5
s_4	0.5	0.5

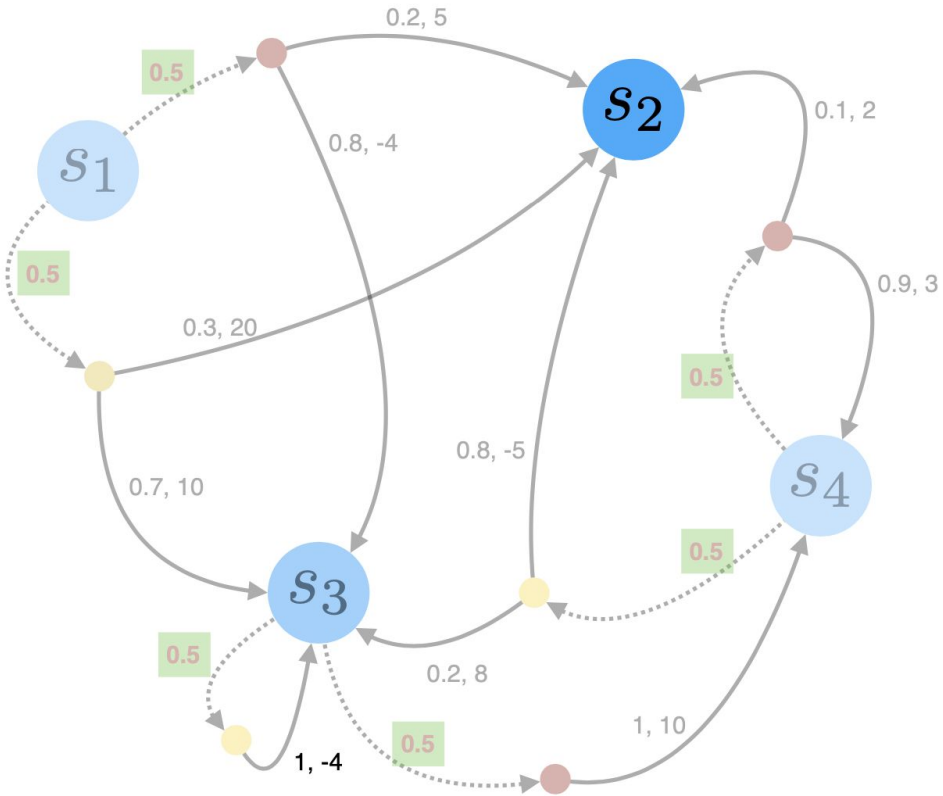
π_0

State	Value
s_1	**5.4**
s_2	0
s_3	0
s_4	0

$v_1(s)$

0.5 * (0.2 * [5 + 0] + 0.8 * [-4 + 0]) + 0.5 * (0.3 * [20 + 0] + 0.7 * [10 + 0])

value iteration: one sweep evaluation



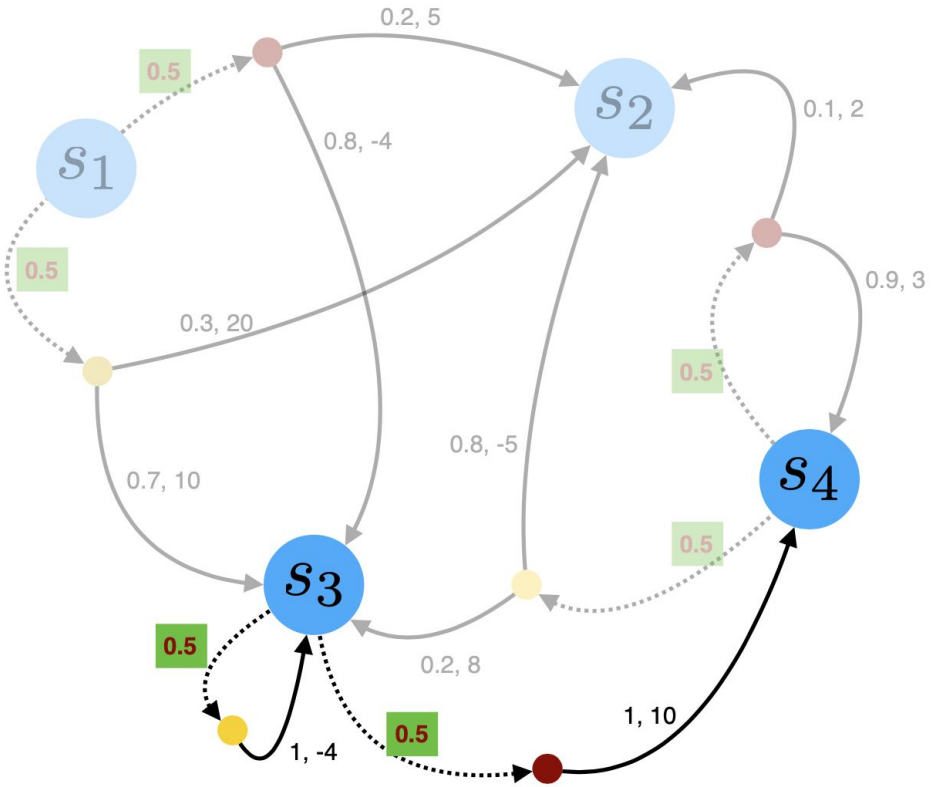
s_1	0.5	0.5
s_2	0.5	0.5
s_3	0.5	0.5
s_4	0.5	0.5

π_0

State	Value
s_1	5.4
s_2	**0**
s_3	0
s_4	0

$v_1(s)$

value iteration: one sweep evaluation



s_1	0.5	0.5
s_2	0.5	0.5
s_3	0.5	0.5
s_4	0.5	0.5

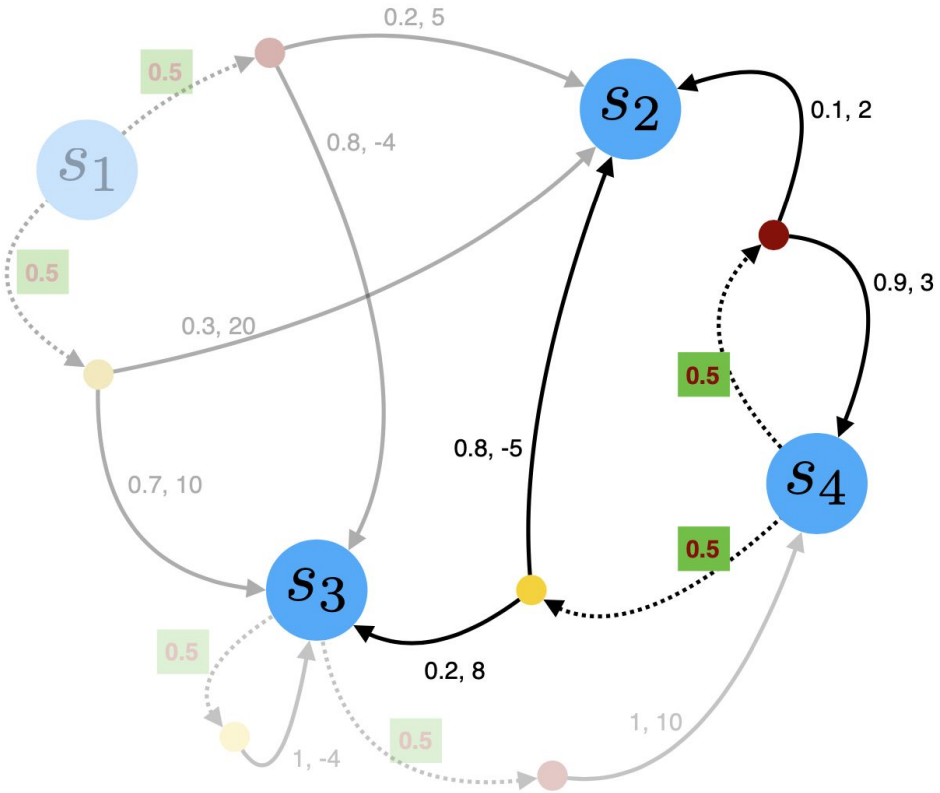
π_0

State	Value
s_1	5.4
s_2	0
s_3	**3**
s_4	0

$v_1(s)$

0.5 * (1 * [-4 + 0]) + 0.5 * (1 * [10 + 0])

value iteration: one sweep evaluation



s_1	0.5	0.5
s_2	0.5	0.5
s_3	0.5	0.5
s_4	0.5	0.5

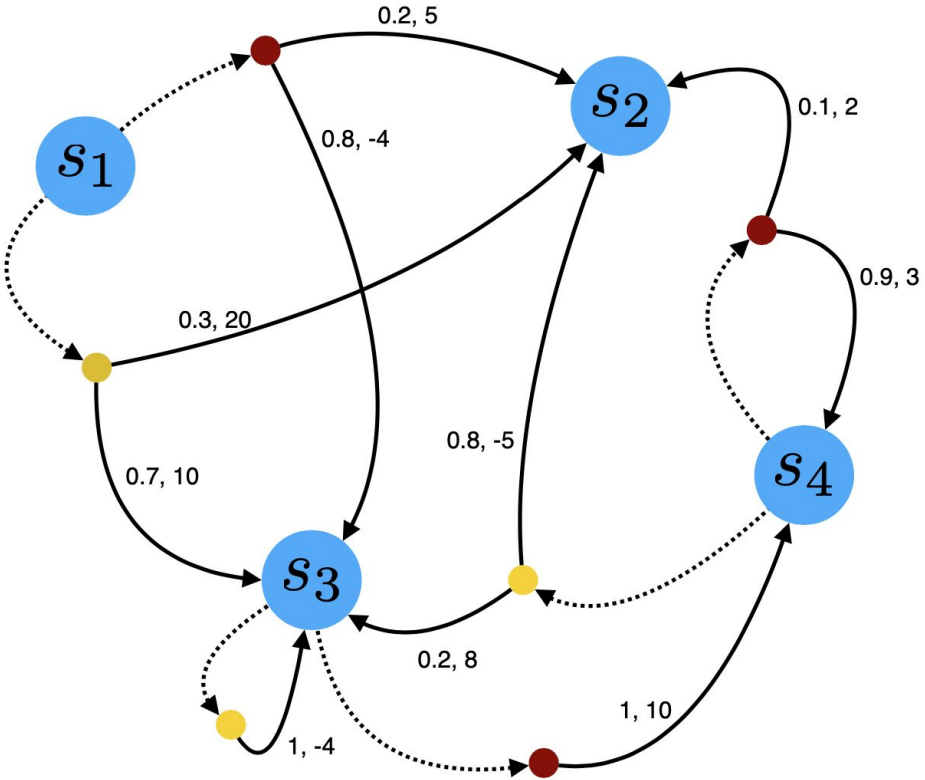
π_0

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

0.5 * (0.1 * [2 + 0] + 0.9 * [3 + 0]) + 0.5 * (0.2 * [8 + 0] + 0.8 * [-5 + 0])

value iteration: one sweep greedification



$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

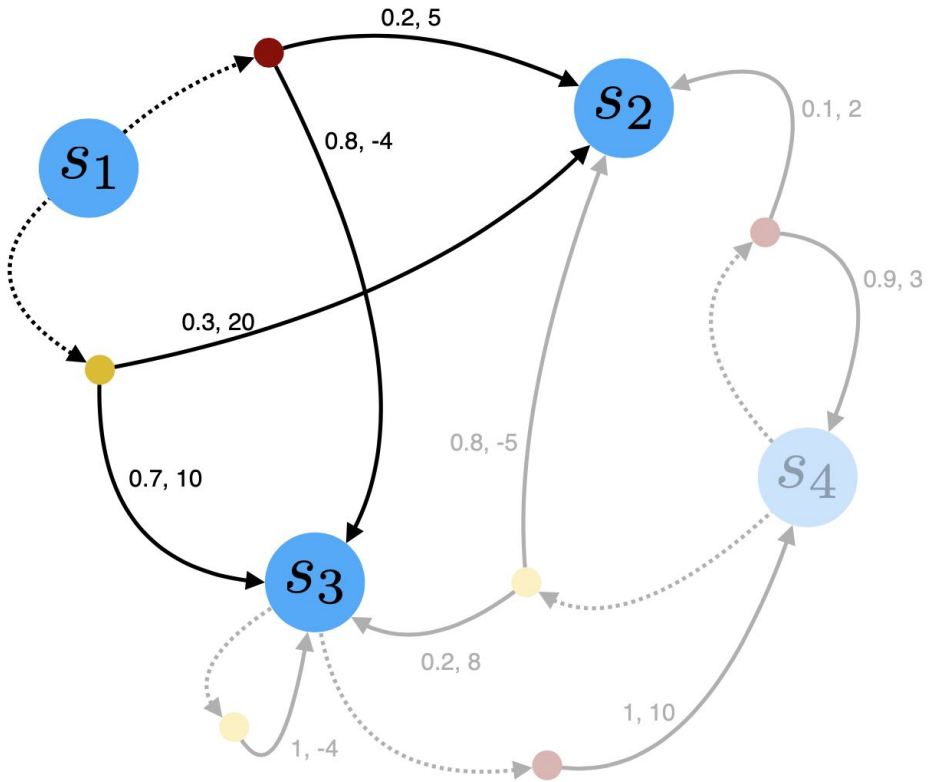
s_1	
s_2	
s_3	
s_4	

π_1

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

value iteration: one sweep greedification



0.2 * (5 + 0.8 * 0) + 0.8 * (-4 + 0.8 * 3) = -0.28

0.3 * (20 + 0.8 * 0) + 0.7 * (10 + 0.8 * 3) = 14.68

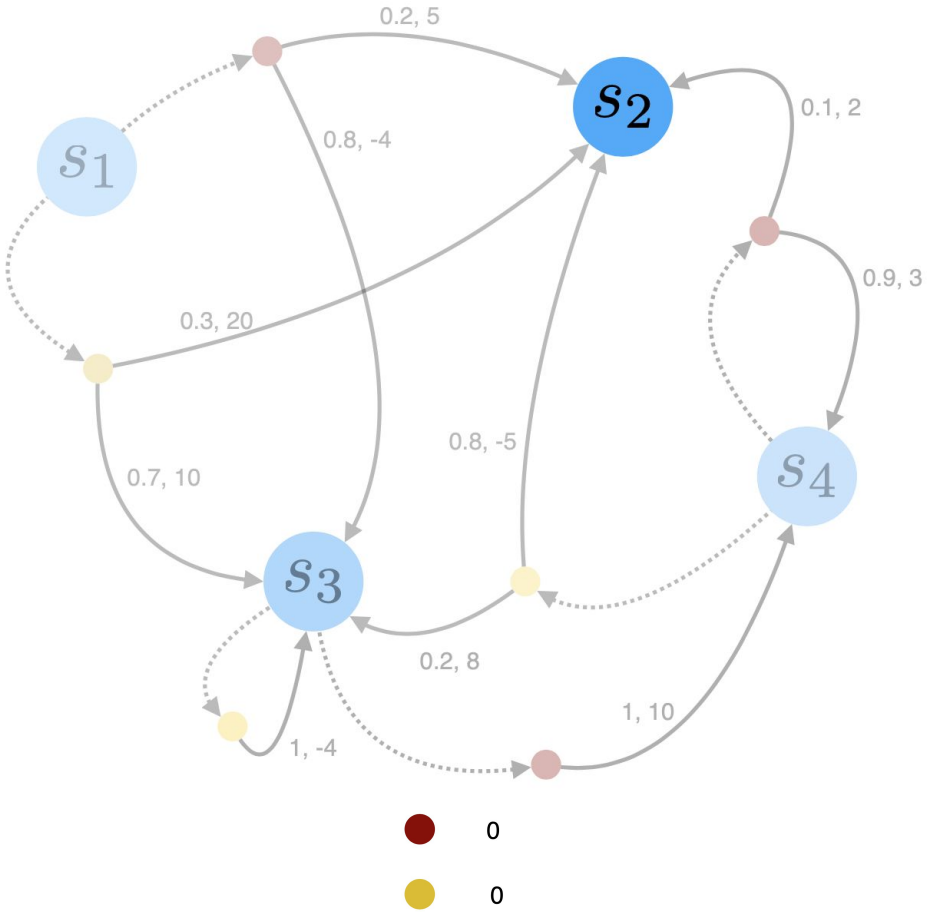
s_1	0	1
s_2		
s_3		
s_4		

π_1

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

value iteration: one sweep greedification



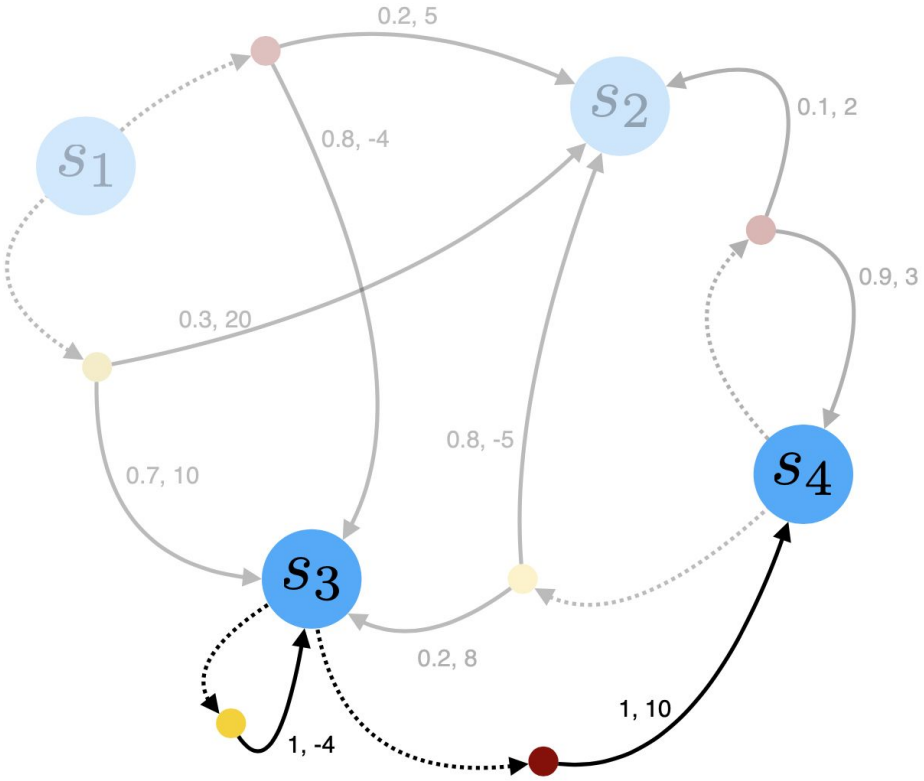
s_1	0	1
s_2	0.5	0.5
s_3		
s_4		

π_1

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

value iteration: one sweep greedification



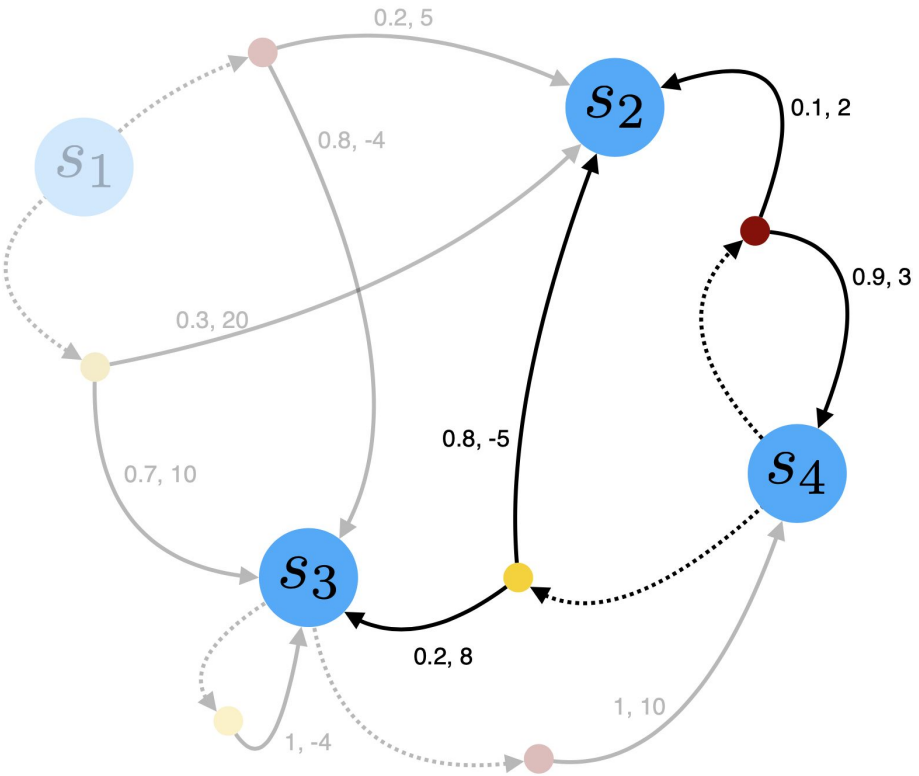
s_1	0	1
s_2	0.5	0.5
s_3	1	0
s_4		

π_1

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

value iteration: one sweep greedification



s_1	0	1
s_2	0.5	0.5
s_3	1	0
s_4	1	0

π_1

State	Value
s_1	5.4
s_2	0
s_3	3
s_4	**0.25**

$v_1(s)$

0.1 * (2 + 0.8 * 0) + 0.9 * (3 + 0.8 * 0.25) = 3.08

0.8 * (-5+ 0.8 * 0) + 0.2 * (8 + 0.8 * 3) = -1.92

Summary

Policy Evaluation: Bellman expectation backup operators (without a max)

Policy Improvement: form a greedy policy, if only locally

Policy Iteration: alternate the above two processes

Value Iteration: Bellman optimality backup operators (with a max)

DP is used when we know how the world works. Biggest limitation of DP is that it requires a probability model (as opposed to a generative or simulation model).

DP uses Full Backups (to be contrasted later with sample backups)

Next Lecture: MC and TD when we don't know how the world works