UPPSALA
UNIVERSITET

Reinforcement Learning

Lecture 3 - Dynamic Programming

Per Mattsson

2022

Department of Information Technology

Repetition

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  Is in general a stochastic variable.

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  Is in general a stochastic variable.
- **State-value function:**

  Expected return when starting in s and following policy π,

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  Is in general a stochastic variable.
- **State-value function:**

  Expected return when starting in s and following policy π,

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right].$$

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  Is in general a stochastic variable.
- **State-value function:**

  Expected return when starting in s and following policy π,

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right].$$

- **Action-value function:**

  Expected return when starting in s, taking action a and *then* follow π,

Important concepts

- **States, actions and rewards:** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$.
- **Dynamics/model:** $p(s', r|s, a)$.
- **Policy:** $\pi(a|s)$ (For deterministic policy also $a = \pi(s)$.)
- **The return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  Is in general a stochastic variable.
- **State-value function:**

  Expected return when starting in s and following policy π,

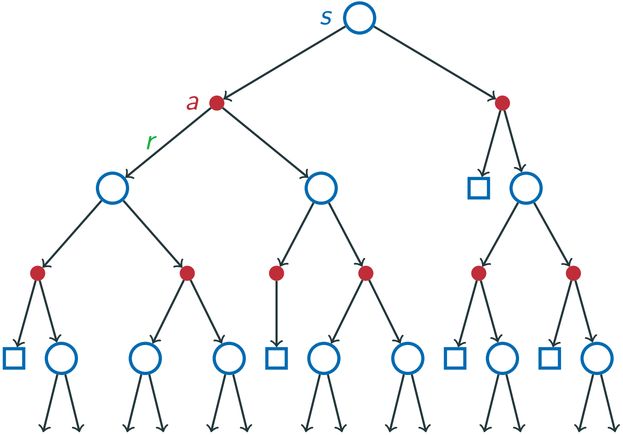
$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s\right].$$

- **Action-value function:**

  Expected return when starting in s, taking action a and *then* follow π,

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a\right]$$

Markov Decision Process Tree



- **Relations:**

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

Relations and Bellman equations

- **Relations:**

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

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

- **Relations:**

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

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

For a deterministic policy $a = \pi(s)$ we have $v_\pi(s) = q_\pi(s, \pi(s))$.

UPPSALA UNIVERSITET

- **Relations:**

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

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

  For a deterministic policy $a = \pi(s)$ we have $v_\pi(s) = q_\pi(s, \pi(s))$.

- **Bellman equation for state-values:**

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

- **Relations:**

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

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

For a deterministic policy $a = \pi(s)$ we have $v_\pi(s) = q_\pi(s, \pi(s))$.

- **Bellman equation for state-values:**

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

Optimal value function and policy

- **Optimal value functions:**

$$v_*(s) = \max_\pi v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$
$$q_*(s, a) = \max_\pi q_\pi(s, a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

Optimal value function and policy

- **Optimal value functions:**

$$v_*(s) = \max_{\pi} v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$

$$q_*(s, a) = \max_{\pi} q_\pi(s, a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

- **Bellman optimality equation:**

$$v_*(s) = \max_{a} q_*(s, a)$$

Optimal value function and policy

- **Optimal value functions:**

$$v_*(s) = \max_\pi v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$

$$q_*(s, a) = \max_\pi q_\pi(s, a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

- **Bellman optimality equation:**

$$v_*(s) = \max_a q_*(s, a) = \max_a \sum_{r,s'} p(s', r | s, a)[r + \gamma v_*(s')].$$

Optimal value function and policy

- **Optimal value functions:**

$$v_*(s) = \max_\pi v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$

$$q_*(s, a) = \max_\pi q_\pi(s, a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

- **Bellman optimality equation:**

$$v_*(s) = \max_a q_*(s, a) = \max_a \sum_{r, s'} p(s', r | s, a)[r + \gamma v_*(s')].$$

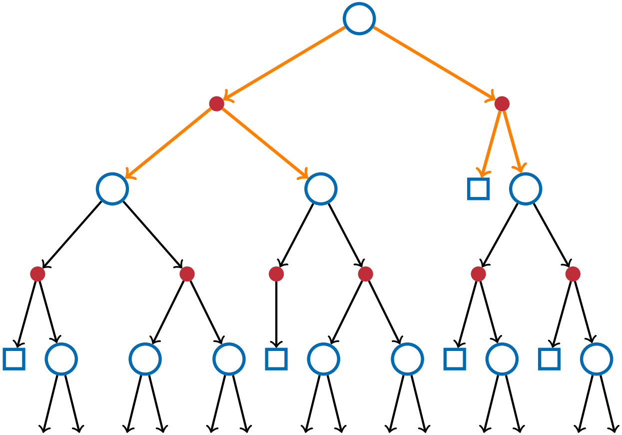
- **Optimal policy:** We get an optimal policy if we act greedily w.r.t v_*, i.e.,

$$\pi_*(s) = \arg\max_a q_*(s, a).$$

Dynamic Programming?

- Dynamic: Sequential or temporal component to the problem.
- Programming: optimizing a “program” (c.f. linear programming)
- Solving complex problems by breaking them down into subproblems.

Bellman equation: $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$



Two problems

- In this lecture we will assume that we know the dynamics $p(s', r|s, a)$.

Two problems

UPPSALA UNIVERSITET

- In this lecture we will assume that we know the dynamics $p(s', r|s, a)$.

Prediction:

- Given a policy π, predict the expected future return from each state.

Two problems

- In this lecture we will assume that we know the dynamics $p(s', r|s, a)$.

Prediction:

- Given a policy π, predict the expected future return from each state.
- That is, find the state-value function $v_\pi(s)$.

Two problems

- In this lecture we will assume that we know the dynamics $p(s', r|s, a)$.

Prediction:

- Given a policy π, predict the expected future return from each state.
- That is, find the state-value function $v_\pi(s)$.

Control:

- Given an MDP, find an optimal policy π_*.

Two problems

- In this lecture we will assume that we know the dynamics $p(s', r|s, a)$.

Prediction:

- Given a policy π, predict the expected future return from each state.
- That is, find the state-value function $v_\pi(s)$.

Control:

- Given an MDP, find an optimal policy π_*.
- If we first compute v_*, then we can use

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

$$\pi_*(s) = \arg\max_a q_*(s, a).$$

Policy evaluation (Prediction)

Policy evaluation (Prediction)

- **Problem:** Given π, compute $v_\pi(s)$ for all $s \in \mathcal{S}$.

Policy evaluation (Prediction)

- **Problem:** Given π, compute $v_\pi(s)$ for all $s \in \mathcal{S}$.
- **The Bellman Equation:**

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

Policy evaluation (Prediction)

- **Problem:** Given π, compute $v_\pi(s)$ for all $s \in \mathcal{S}$.
- **The Bellman Equation:**

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

- System of linear equations. Can in principle be solved analytically.

Policy evaluation (Prediction)

- **Problem:** Given π, compute $v_\pi(s)$ for all $s \in \mathcal{S}$.
- **The Bellman Equation:**

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

- System of linear equations. Can in principle be solved analytically.
- For large state and/or action space, more efficient with iterative solutions.

Iterative Policy Evaluation

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).

Iterative Policy Evaluation

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).
- In each iteration, use the RHS of the Bellman equation:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')], \qquad \text{for all } s \in \mathcal{S}$$

Iterative Policy Evaluation

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).
- In each iteration, use the RHS of the Bellman equation:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')], \qquad \text{for all } s \in \mathcal{S}$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then v_k solves the Bellman equation (i.e. $v_k(s) = v_\pi(s)$)

Iterative Policy Evaluation

UPPSALA UNIVERSITET

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).
- In each iteration, use the RHS of the Bellman equation:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')], \qquad \text{for all } s \in \mathcal{S}$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then v_k solves the Bellman equation (i.e. $v_k(s) = v_\pi(s)$)
- **Convergence:** It can be shown that $v_k(s) \to v_\pi(s)$ as $k \to \infty$.

Iterative Policy Evaluation

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).
- In each iteration, use the RHS of the Bellman equation:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')], \qquad \text{for all } s \in \mathcal{S}$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then v_k solves the Bellman equation (i.e. $v_k(s) = v_\pi(s)$)
- **Convergence:** It can be shown that $v_k(s) \to v_\pi(s)$ as $k \to \infty$.
    - Contraction mapping: Let u_k and v_k be two different estimates, then

$$\|u_{k+1} - v_{k+1}\|_\infty \leq \gamma \|u_k - v_k\|_\infty.$$

Iterative Policy Evaluation

- Start from some initial guess v_0 (e.g $v_0(s) = 0$ for all s).
- In each iteration, use the RHS of the Bellman equation:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')], \qquad \text{for all } s \in \mathcal{S}$$

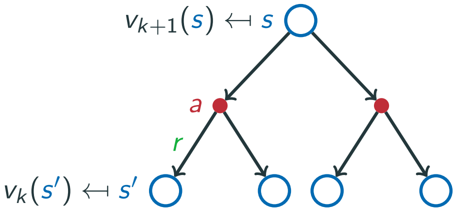
- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then v_k solves the Bellman equation (i.e. $v_k(s) = v_\pi(s)$)
- **Convergence:** It can be shown that $v_k(s) \to v_\pi(s)$ as $k \to \infty$.
  - Contraction mapping: Let u_k and v_k be two different estimates, then

$$\|u_{k+1} - v_{k+1}\|_\infty \leq \gamma \|u_k - v_k\|_\infty.$$

  - With $u_k = v_\pi$:

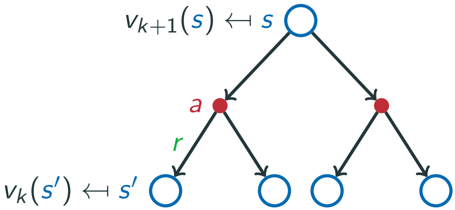
$$\|v_\pi - v_{k+1}\|_\infty \leq \gamma \|v_\pi - v_k\|_\infty.$$

Iterative policy iteration



$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Iterative policy iteration



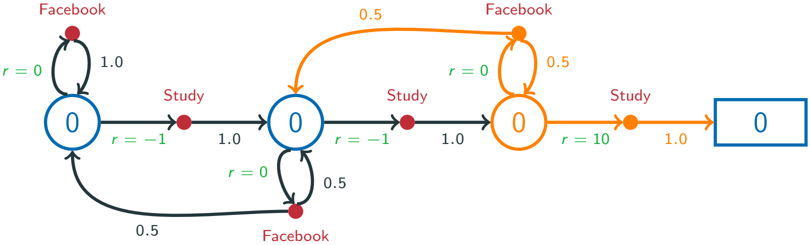
$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Bootstrapping: We use our old estimate (v_k) to compute a new estimate (v_{k+1}).

Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.

v_0

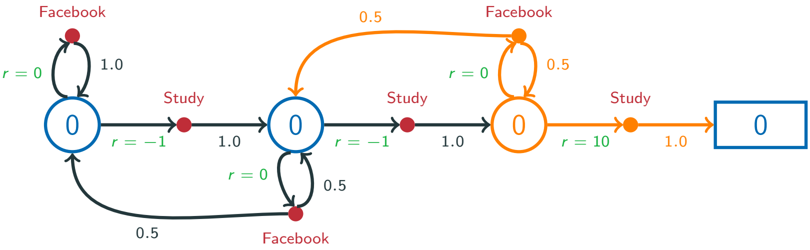


$v_1(s) =$

Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.

v_0

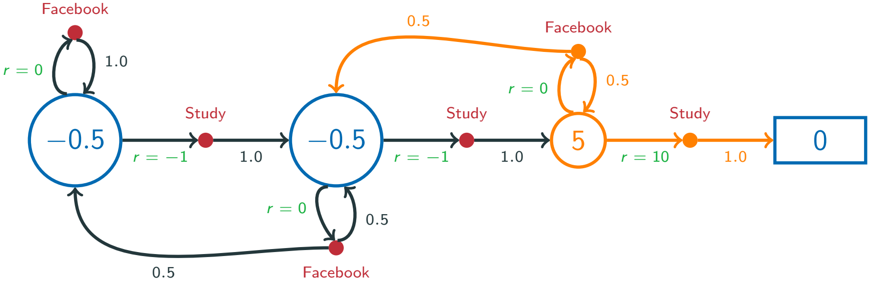


$$v_1(s) = 0.5 \underbrace{[0 + \gamma(0.5 \times 0 + 0.5 \times 0)]}_{\text{facebook}} + 0.5 \underbrace{[10 + 0\gamma]}_{\text{study}} = 5$$

Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.

v_1

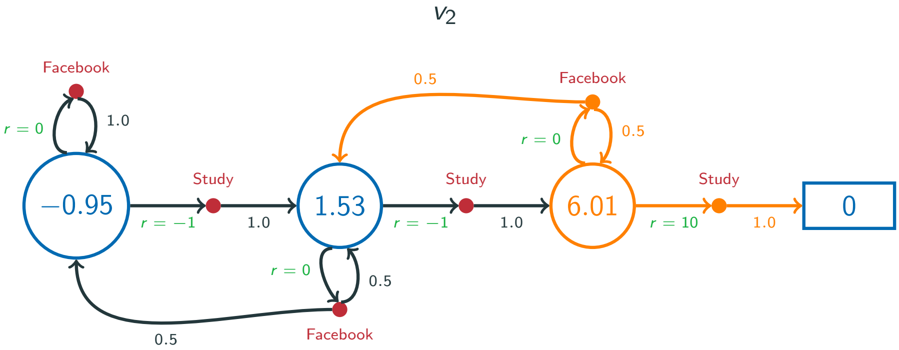


$$v_2(s) = 0.5\,\underbrace{[0 + \gamma(0.5 \times (-0.5) + 0.5 \times 5)]}_{\text{facebook}} + 0.5\,\underbrace{[10 + 0\gamma]}_{\text{study}} = 6.01$$

Example: Iterative Policy Evaluation

Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



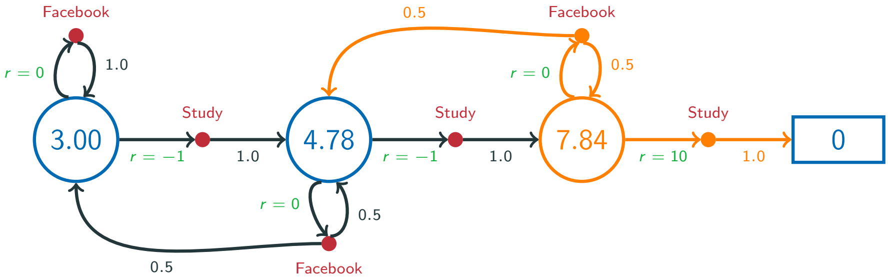
$$v_3(s) = 0.5\underbrace{\left[0 + \gamma(0.5 \times 1.53 + 0.5 \times 6.01)\right]}_{\text{facebook}} + 0.5\underbrace{\left[10 + 0\gamma\right]}_{\text{study}} = 6.70$$

Example: Iterative Policy Evaluation

Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.

$v_{30} \approx v_\pi$



$$v_{31}(s) = 0.5\underbrace{\left[0 + \gamma(0.5 \times 4.78 + 0.5 \times 7.84)\right]}_{\text{facebook}} + 0.5\underbrace{\left[10 + 0\gamma\right]}_{\text{study}} \approx 7.84$$

Discount: $\gamma = 1$

Policy: Uniform policy (all actions equally likely)

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Example: Iterative Policy Evaluation

Discount: $\gamma = 1$

Policy: Uniform policy (all actions equally likely)

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

Discount: $\gamma = 1$

Policy: Uniform policy (all actions equally likely)

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

UPPSALA UNIVERSITET

Discount: $\gamma = 1$

Policy: Uniform policy (all actions equally likely)

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

$k = 3$

0	-2.43	-2.94	-3
-2.43	-2.88	-3	-2.94
-2.94	-3	-2.88	-2.43
-3	-2.94	-2.43	0

Example: Iterative Policy Evaluation

Discount: $\gamma = 1$

Policy: Uniform policy (all actions equally likely)

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

$k = 3$

0	-2.43	-2.94	-3
-2.43	-2.88	-3	-2.94
-2.94	-3	-2.88	-2.43
-3	-2.94	-2.43	0

$k = \infty$

	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	

Implementation

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.

Implementation

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \to v_\pi$ as $k \to \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.

Implementation

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \rightarrow v_\pi$ as $k \rightarrow \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
    1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)

Implementation

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \to v_\pi$ as $k \to \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
  1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)
  2. For all $s \in \mathcal{S}$:

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

UPPSALA UNIVERSITET

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \to v_\pi$ as $k \to \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
  1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)
  2. For all $s \in \mathcal{S}$:
  $$v_{\text{new}}(s) = \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_{\text{old}}(s')]$$
  3. If $|v_{\text{old}}(s) - v_{\text{new}}(s)| < \text{tol}$ for all s, output v_{new} and stop.

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \to v_\pi$ as $k \to \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
  1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)
  2. For all $s \in \mathcal{S}$:

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

  3. If $|v_{\text{old}}(s) - v_{\text{new}}(s)| < \text{tol}$ for all s, output v_{new} and stop.
  4. Otherwise, let $v_{\text{old}} \leftarrow v_{\text{new}}$ and go back to step 2.

UPPSALA UNIVERSITET

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \to v_\pi$ as $k \to \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
  1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)
  2. For all $s \in \mathcal{S}$:

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

  3. If $|v_{\text{old}}(s) - v_{\text{new}}(s)| < \text{tol}$ for all s, output v_{new} and stop.
  4. Otherwise, let $v_{\text{old}} \leftarrow v_{\text{new}}$ and go back to step 2.
- This algorithm do *synchronous* updates.

- For finite $\mathcal{S}$ we can represent $v(s)$ as an array with one element for each sate in $\mathcal{S}$.
- $v_k \rightarrow v_\pi$ as $k \rightarrow \infty$. But in practice we stop when the difference between v_{k+1} and v_k is small enough.
- The algorithm is then
  1. Initialize v_{old} (e.g. $v_{\text{old}}(s) = 0$ for all s)
  2. For all $s \in \mathcal{S}$:

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

  3. If $|v_{\text{old}}(s) - v_{\text{new}}(s)| < \text{tol}$ for all s, output v_{new} and stop.
  4. Otherwise, let $v_{\text{old}} \leftarrow v_{\text{new}}$ and go back to step 2.
- This algorithm do *synchronous* updates.
- Asynchronous updates also converge to $v_\pi(s)$, as long as we keep updating all states.

In place updates

1. Start with initial $v(s)$ (e.g. $v(s) = 0$).
2. For all $s \in \mathcal{S}$:

$$v(s) \leftarrow \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v(s')]$$

3. If changes in v are small enough we are done, otherwise go to step 2.

In place updates

UPPSALA UNIVERSITET

1. Start with initial $v(s)$ (e.g. $v(s) = 0$).
2. For all $s \in \mathcal{S}$:

$$v(s) \leftarrow \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v(s')]$$

3. If changes in v are small enough we are done, otherwise go to step 2.

- Easier to implement (only needs one array $v(s)$).

In place updates

1. Start with initial $v(s)$ (e.g. $v(s) = 0$).
2. For all $s \in \mathcal{S}$:

$$v(s) \leftarrow \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v(s')]$$

3. If changes in v are small enough we are done, otherwise go to step 2.

- Easier to implement (only needs one array $v(s)$).
- Now the updates depends on which order we sweep through the states.

In place updates

1. Start with initial $v(s)$ (e.g. $v(s) = 0$).
2. For all $s \in \mathcal{S}$:

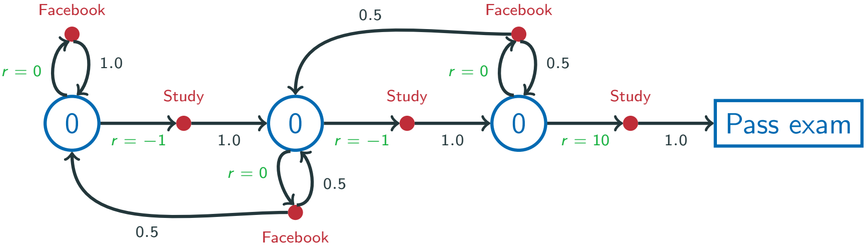
$$v(s) \leftarrow \sum_{a} \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v(s')]$$

3. If changes in v are small enough we are done, otherwise go to step 2.

- Easier to implement (only needs one array $v(s)$).
- Now the updates depends on which order we sweep through the states.
- Also converges to $v_\pi(s)$, often faster even.

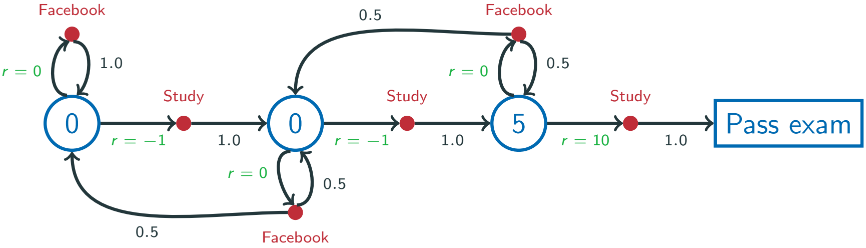
Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



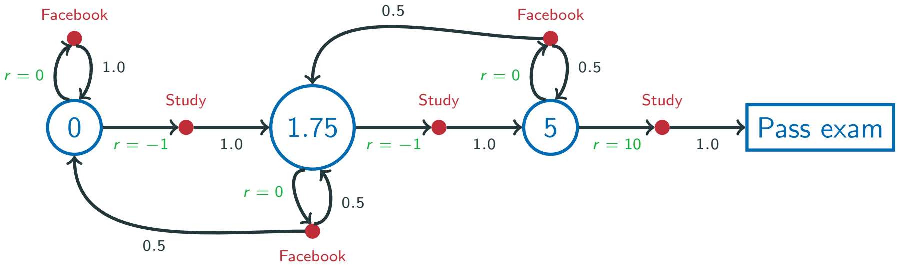
Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



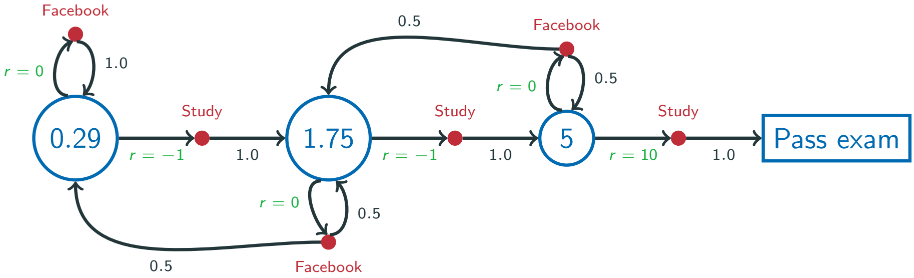
Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



Discount: $\gamma = 0.9$.

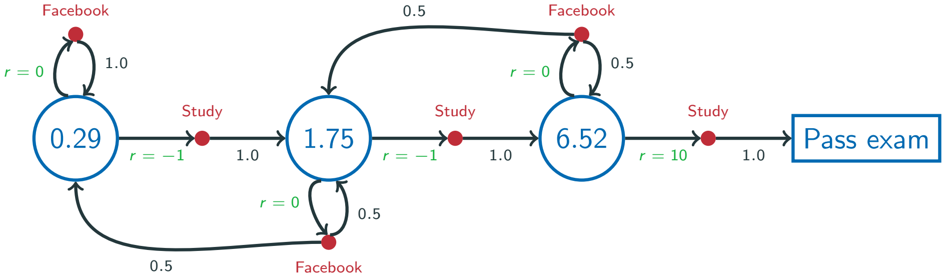
Policy: $\pi(a|s) = 0.5$ for all a and s.



UPPSALA UNIVERSITET

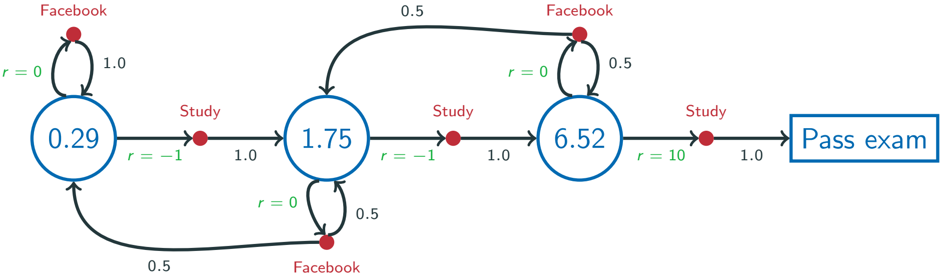
Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



If we continue like this, we will converge to $v_\pi(s)$

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	-1	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	-1	-1.25	0
0	0	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	-1	-1.25	-1.31
0	0	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	-1	-1.25	-1.31
-1	0	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

$v(s)$

0	-1	-1.25	-1.31
-1	-1.5	0	0
0	0	0	0
0	0	0	0

Example: In-place updates

Discount: $\gamma = 1$

Policy: Uniform.

One sweep over all states

0	-1	-1.25	-1.31
-1	-1.5	-1.69	-1.75
-1.25	-1.69	-1.84	-1.90
-1.31	-1.75	-1.90	0

Policy improvement

Policy improvement

- Given a policy π, we have seen how to evaluate $v_\pi(s)$.

Policy improvement

- Given a policy π, we have seen how to evaluate $v_\pi(s)$.
- Can we now find a better policy? That is, a policy π' such that

$$v_{\pi'}(s) \geq v_\pi(s), \quad \text{for all } s \in \mathcal{S}?$$

Policy improvement

- Given a policy π, we have seen how to evaluate $v_\pi(s)$.
- Can we now find a better policy? That is, a policy π' such that

$$v_{\pi'}(s) \geq v_\pi(s), \quad \text{for all } s \in \mathcal{S}?$$

- The value of taking action a in state s and then following policy π afterwards is

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

Policy improvement

- Given a policy π, we have seen how to evaluate $v_\pi(s)$.
- Can we now find a better policy? That is, a policy π' such that

$$v_{\pi'}(s) \geq v_\pi(s), \quad \text{for all } s \in \mathcal{S}?$$

- The value of taking action a in state s and then following policy π afterwards is

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

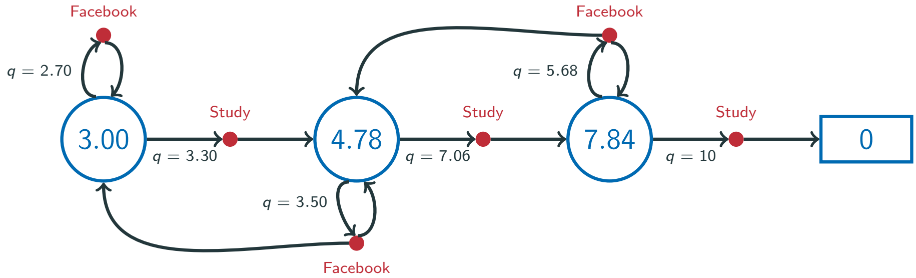
- Let us try to act **greedily** with respect to the action values, i.e.

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

Example: Greedy

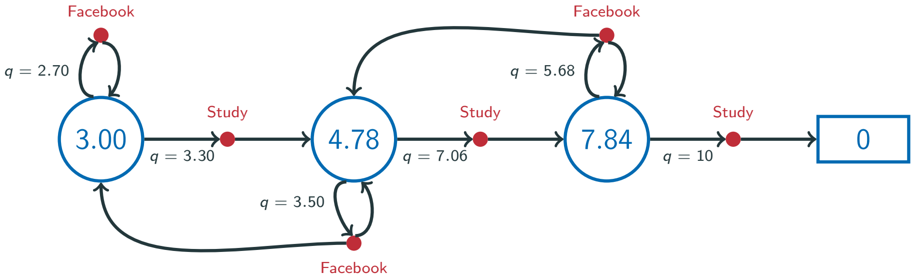
Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



Discount: $\gamma = 0.9$.

Policy: $\pi(a|s) = 0.5$ for all a and s.



The greedy policy with respect to v_π is

$$\pi'(s) = \arg\max_a q_\pi(s, a) = \text{study}.$$

The Policy Improvement Theorem

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)

The Policy Improvement Theorem

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

- Hence

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a)$$

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

- Hence

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) \geq q_\pi(s, \pi(s))$$

The Policy Improvement Theorem

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

- Hence

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s).$$

The Policy Improvement Theorem

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_{a} q_\pi(s, a).$$

- Hence

$$q_\pi(s, \pi'(s)) = \max_{a} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s).$$

The Policy Improvement Theorem

If $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in \mathcal{S}$, then

$$v_{\pi'}(s) \geq v_\pi(s) \quad \text{for all } s \in \mathcal{S}$$

The Policy Improvement Theorem

- Consider deterministic policy $a = \pi(s)$ (but result holds for stochastic $\pi(a|s)$ too)
- The greedy policy w.r.t $v_\pi(s)$ is

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

- Hence

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s).$$

The Policy Improvement Theorem

If $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in \mathcal{S}$, then

$$v_{\pi'}(s) \geq v_\pi(s) \quad \text{for all } s \in \mathcal{S}$$

- So $\pi'(s) = \arg\max_a q_\pi(s, a)$ is as good as, or better, than $\pi(s)$.

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

$$v_\pi(s) \leq q_\pi(s, \pi'(s))$$

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

$$v_\pi(s) \leq q_\pi(s, \pi'(s)) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)]$$

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

$$
\begin{aligned}
v_\pi(s) \leq q_\pi(s, \pi'(s)) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s]
\end{aligned}
$$

UPPSALA UNIVERSITET

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

$$
\begin{aligned}
v_\pi(s) \leq q_\pi(s, \pi'(s)) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}] | S_t = s]
\end{aligned}
$$

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

$$\begin{aligned}
v_\pi(s) \leq q_\pi(s, \pi'(s)) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s]
\end{aligned}$$

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Proof of the policy improvement theorem

- Consider two deterministic policies such that $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then

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

Is it really an improvement?

- **Policy improvement:**

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

will be *at least* as good as π in all states.

Is it really an improvement?

- **Policy improvement:**

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

  will be *at least* as good as π in all states.

- **No improvement?** What if $v_\pi(s) = v_{\pi'}(s)$ for all s?

  Then from previous slide we see that

$$q_\pi(s,\pi'(s)) = \max_{a} q_\pi(s,a) = q_\pi(s,\pi(s)) = v_\pi(s)$$

Is it really an improvement?

- **Policy improvement:**

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

  will be *at least* as good as π in all states.

- **No improvement?** What if $v_\pi(s) = v_{\pi'}(s)$ for all s?

  Then from previous slide we see that

$$q_\pi(s, \pi'(s)) = \max_{a} q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

  so

$$v_\pi(s) = \max_{a} q_\pi(s, a) \quad \text{for all } s.$$

Is it really an improvement?

- **Policy improvement:**

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

  will be *at least* as good as π in all states.

- **No improvement?** What if $v_\pi(s) = v_{\pi'}(s)$ for all s?

  Then from previous slide we see that

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

  so

$$v_\pi(s) = \max_a q_\pi(s, a) \quad \text{for all } s.$$

- This is the *Bellman optimality equation*! So π and π' are optimal policies!

Is it really an improvement?

- **Policy improvement:**

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

  will be *at least* as good as π in all states.

- **No improvement?** What if $v_\pi(s) = v_{\pi'}(s)$ for all s?

  Then from previous slide we see that

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

  so

$$v_\pi(s) = \max_a q_\pi(s, a) \quad \text{for all } s.$$

- This is the *Bellman optimality equation*! So π and π' are optimal policies!
- **Conclusion:** π' will be strictly better than π, unless π is already optimal!

Policy Iteration

Start with an initial policy π.

Policy Iteration

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.

UPPSALA UNIVERSITET

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.
- **3.** If we have a strict improvement go to 1. Otherwise we have found optimal policy π.

Policy Iteration

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.
- **3.** If we have a strict improvement go to 1. Otherwise we have found optimal policy π.

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

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.
- **3.** If we have a strict improvement go to 1. Otherwise we have found optimal policy π.

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

In a finite MDP this will converge in a finite number of iterations.

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.
- **3.** If we have a strict improvement go to 1. Otherwise we have found optimal policy π.

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

In a finite MDP this will converge in a finite number of iterations.

Some implementation details:

- In E: Start from v for the previous policy to speed up computation.

Start with an initial policy π.

- **1. Policy evaluation (E):** Compute $v_\pi(s)$ for all s. (Iterative policy evaluation).
- **2. Policy improvement (I):** Let $\pi'(s) = \arg\max_a q_\pi(s, a)$ for all s.
- **3.** If we have a strict improvement go to 1. Otherwise we have found optimal policy π.

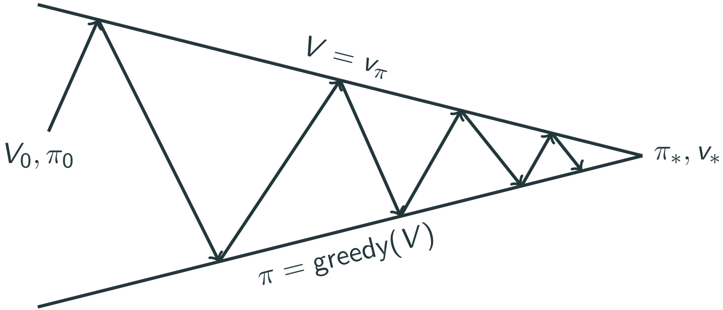
$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*.$$

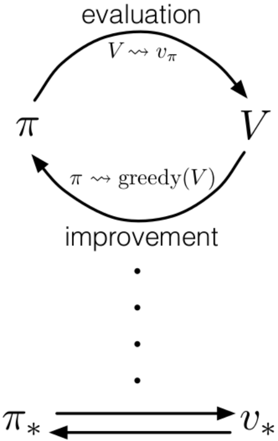
In a finite MDP this will converge in a finite number of iterations.

Some implementation details:

- In E: Start from v for the previous policy to speed up computation.
- In I: If there are several a that maximize $q_\pi(s, a)$, choose arbitrarily (or use a stochastic policy that picks between them with e.g. uniform probability).

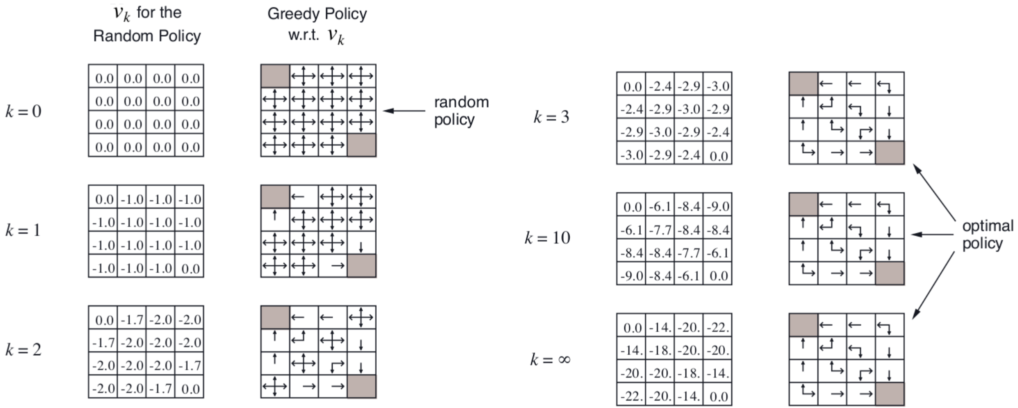
Policy Iteration





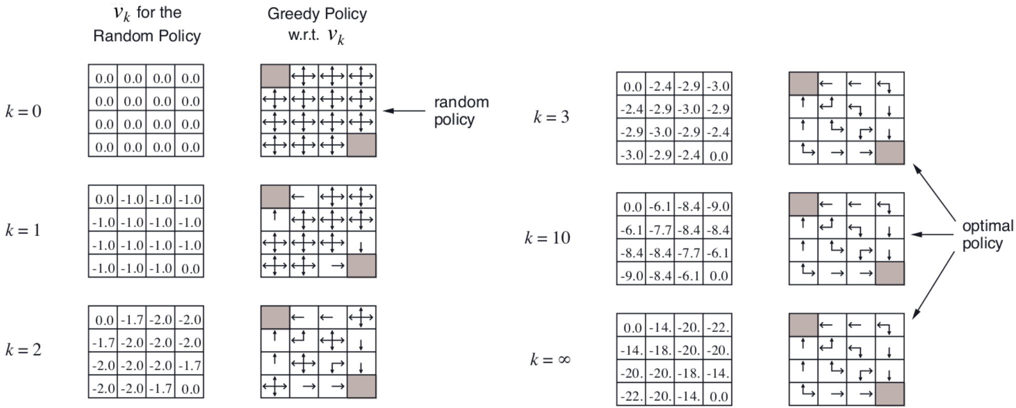
UPPSALA UNIVERSITET

Evaluating the uniform policy.



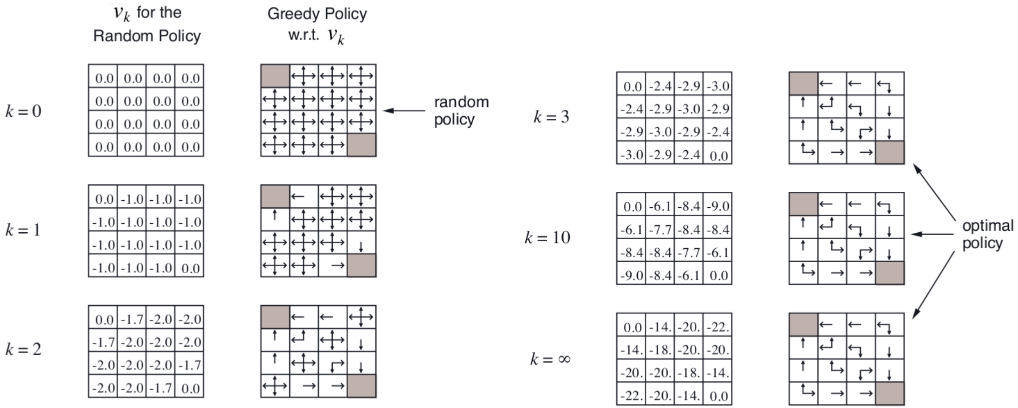
Example 4.1: Policy evaluation

Evaluating the uniform policy.



- Do we really have to wait for the E to converge before we do I?

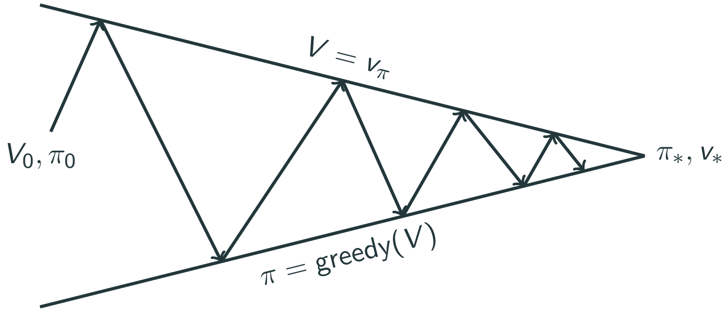
Evaluating the uniform policy.

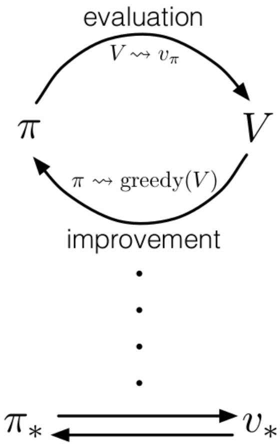


- Do we really have to wait for the E to converge before we do I?
- Definitely not in this case!

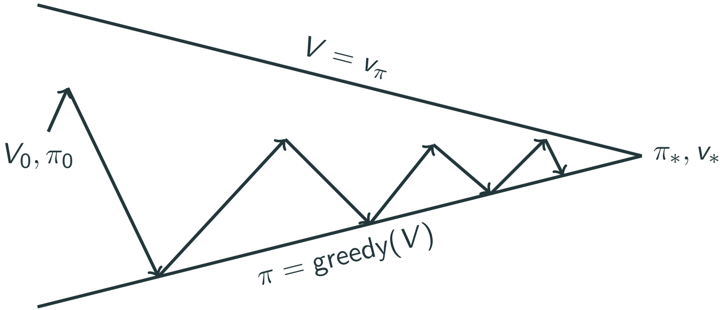
Value iteration

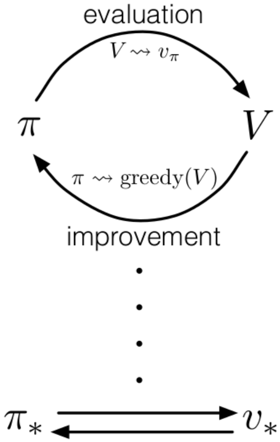
Generalized Policy Iteration



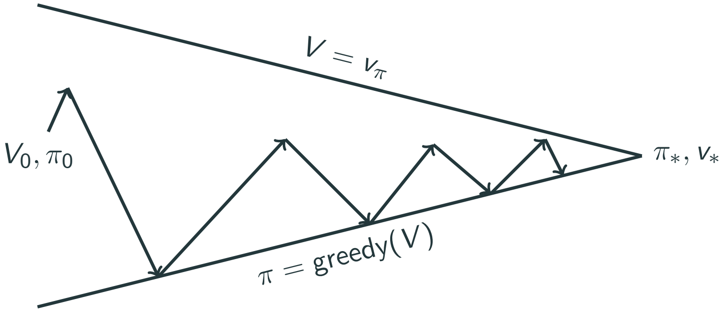


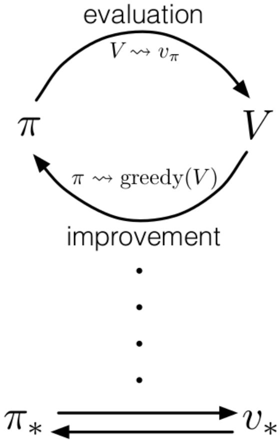
- **Policy iteration:** Evaluation complete before we improve.





- **Policy iteration:** Evaluation complete before we improve.
- **Generalized policy iteration**.





- **Policy iteration:** Evaluation complete before we improve.
- **Generalized policy iteration**.
- **Value iteration:** Stop evaluation after one sweep over all states!

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.

Value iteration

UPPSALA UNIVERSITET

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s, a) = \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_{a} q_{k+1}(s,a)$$

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_a q_{k+1}(s,a)$$

$$v_{k+1}(s) = q_{k+1}(s,\pi_{k+1}(s)).$$

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s, a) = \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_{a} q_{k+1}(s, a)$$

$$v_{k+1}(s) = q_{k+1}(s, \pi_{k+1}(s)).$$

- Or in one equation

$$v_{k+1}(s) =$$

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_a q_{k+1}(s,a)$$

$$v_{k+1}(s) = q_{k+1}(s, \pi_{k+1}(s)).$$

- Or in one equation

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')].$$

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s, a) = \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_a q_{k+1}(s, a)$$

$$v_{k+1}(s) = q_{k+1}(s, \pi_{k+1}(s)).$$

- Or in one equation

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

- With this iteration we will converge to the optimal $v_*(s)$.

Value iteration

- Lets see what happens if we only do one iteration of evaluation before we improve.
- That is, for all s

$$q_{k+1}(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_a q_{k+1}(s,a)$$

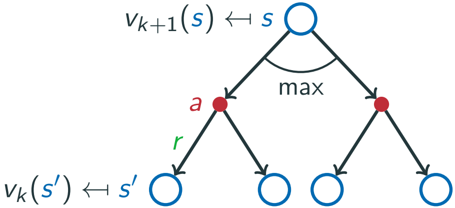
$$v_{k+1}(s) = q_{k+1}(s, \pi_{k+1}(s)).$$

- Or in one equation

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s',r|s,a)[r + \gamma v_k(s')].$$

- With this iteration we will converge to the optimal $v_*(s)$.
- Can use e.g. in place updates instead of synchronous updates.

Value Iteration



$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Value Iteration and the Bellman Optimality Equation

UPPSALA UNIVERSITET

- **Value iteration:**

$$v_{k+1}(s) = \max_{a} \sum_{r,s'} p(s', r | s, a)[r + \gamma v_k(s')]$$

Value Iteration and the Bellman Optimality Equation

- **Value iteration:**

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s

- **Value iteration:**

$$v_{k+1}(s) = \max_{a} \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then

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

This is the *Bellman optimality equation*, so $v_k(s)$ is the optimal value function.

Value Iteration and the Bellman Optimality Equation

- **Value iteration:**

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')]$$

- **Fixed point:** If $v_k(s) = v_{k+1}(s)$ for all s, then

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

  This is the *Bellman optimality equation*, so $v_k(s)$ is the optimal value function.

- **Optimal policy:** When converged to v_* we can find an optimal policy

$$\pi_*(s) = \arg\max_a q_*(s, a).$$

Example: GridWorld

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Discount: $\gamma = 1$, Reward: -1 for each action.

v_0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Example: GridWorld

$$v_{k+1}(s) = \max_{a} \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Discount: $\gamma = 1$, Reward: -1 for each action.

v_0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

v_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

Example: GridWorld

$$v_{k+1}(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Discount: $\gamma = 1$, Reward: -1 for each action.

v_0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

v_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

v_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-1
-2	-2	-1	0

$$v_{k+1}(s) = \max_{a} \sum_{r,s'} p(s', r|s, a)[r + \gamma v_k(s')].$$

Discount: $\gamma = 1$, Reward: -1 for each action.

v_0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

v_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

v_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-1
-2	-2	-1	0

$v_3 = v_*$

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

Summary

Problem	**Based on**	**Algorithm**
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π $+$ Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

Summary

UPPSALA UNIVERSITET

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.

Summary

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π $+$ Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.
- All methods could also be applied to $q(s, a)$.

Summary

UPPSALA UNIVERSITET

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.
- All methods could also be applied to $q(s, a)$.

Next:

Summary

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π $+$ Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.
- All methods could also be applied to $q(s, a)$.

Next:

- What to do if $p(s', r|s, a)$ is unknown?

Summary

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π $+$ Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.
- All methods could also be applied to $q(s, a)$.

Next:

- What to do if $p(s', r|s, a)$ is unknown?
- Before that: Look at Tinkering Notebook 2.

Summary

UPPSALA UNIVERSITET

Problem	Based on	Algorithm
Prediction	Bellman Equation for v_π	Iterative Policy Evaluation
Control	Bellman Equation for v_π + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality equation	Value Iteration

- For all methods the main idea is to apply the right-hand side of the corresponding Bellman equation repeatedly until convergence.
- All methods could also be applied to $q(s, a)$.

Next:

- What to do if $p(s', r|s, a)$ is unknown?
- Before that: Look at Tinkering Notebook 2.
- You are now ready for Assignment 1!