Reinforcement Learning
Lecture 3: Dynamic Programming

Lecturer: Prof. Dr. Mathias Niepert

Institute for Artificial Intelligence
Machine Learning and Simulation Lab

University of Stuttgart
Germany

imprs-is

April 25, 2024

Outline

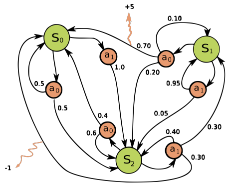
1. Markov Decision Process
2. Iterative Policy Improvement
3. Generalized Policy Iteration

Markov Decision Process

Recap: Markov Decision Process (MDP)

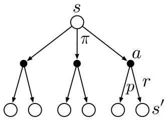
An MDP is a 5-tuple (S, A, T, r, γ):

1. States $\mathbf{s} \in S$ (green circles)
2. Actions $\mathbf{a} \in A$ (orange circles)
3. Transition model (or dynamics) $T(\mathbf{s}, \mathbf{a}, \mathbf{s}') = p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
4. Reward function $r(\mathbf{s}, \mathbf{a}, \mathbf{s}') \in \mathbb{R}$
5. Discount factor $\gamma \in [0, 1]$



MDP with 3 states and 2 actions, rewards as orange arrows

Bellman equation for v_π



Recursive relationship for v_π

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\
&= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi [G_{t+1} \mid S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma v_\pi(s') \Big] \quad \text{for all } s \in \mathcal{S}
\end{aligned}
$$

Tansition Matrix

For an MDP, the state transition probability is defined by $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$:

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

State transition matrix $\mathcal{P}$ (for action a) defines **transition probabilities** from all states s to all successor states s' when taking action a

$$\mathcal{P} = \text{from} \; \begin{array}{c} \text{to} \\ \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ldots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \end{array}$$

where each row of the matrix sums to 1. This is for **one action** a!

Bellman equation in matrix form

The Bellman equation can be expressed concisely using matrices [1],

$$\underbrace{\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}}_{v} = \underbrace{\begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix}}_{\mathcal{R}} + \gamma \underbrace{\begin{bmatrix} p_{11} & \dots & p_{1n} \\ \vdots & \dots & \vdots \\ p_{n1} & \dots & p_{nn} \end{bmatrix}}_{\mathcal{P}} \underbrace{\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}}_{v}$$

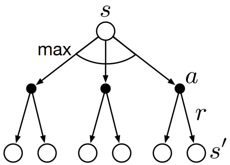
- The Bellman equation is a **linear equation**
- It can be solved directly:

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ v(I - \gamma \mathcal{P}) &= \mathcal{R} \\ v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R} \leftarrow \textbf{Prediction} \end{aligned}$$

- Computational complexity is $O(n^3)$ for n states

[1] Marginalized over actions

Bellman equation for optimal value function v_*

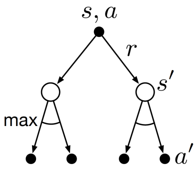


Value under optimal policy = expected return for best action from that state.

$$
\begin{aligned}
v_*(s) &= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_*(s')\Big]
\end{aligned}
$$

Bellman equation for optimal action-value function q_*

$$q_*(s,a) = \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right]$$



Solving MDPs

- **Policy evaluation / prediction:** given an MDP and a policy $\pi(a \mid s) = \Pr\{A_t = a \mid S_t = s\}$ find the value and action-value functions
- **Policy improvement:** given an MDP and value function, find a better (optimal) policy π_*
- **Dynamic programming**
  Given access to the perfect model $p(r, s' \mid s, a)$
  compute v_*, q_* – the optimal value and action-value functions

Policy evaluation

- We have some policy π which tells the agent which action a to choose in state s
- Find the value function $v_\pi(s)$ of this policy, i.e. **evaluate** the policy π
- **Bellman equation** for v_π:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma\, v_\pi(S_{t+1}) \mid S_t = s\right] \\ &= \textstyle\sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma\, v_\pi(s')\Big] \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

- System of linear equations
- Solvable, but ...
  - $|\mathcal{S}|$ equations
  - $|\mathcal{S}|$ unknowns

Policy evaluation

- Each iteration is one *sweep* through the state space
- Value estimates for each state are updated using the previous estimate (*backup*)

$$v_0 \to v_1 \to \ldots \to v_k \to v_{k+1} \to \ldots \to v_\pi$$

- **Bellman equation** is used as an **iterative backup update**

$$\begin{aligned} v_{k+1}(s) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s \right] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma v_k(s') \Big] \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

- After many *sweeps*, iterative policy evaluation converges to v_π

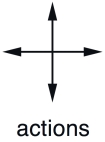
Policy evaluation

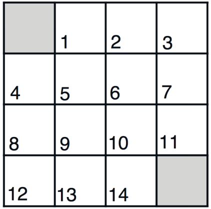
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

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

Policy evaluation example





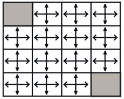
$R_t = -1$
on all transitions

Policy takes actions uniformly at random.

Policy evaluation example

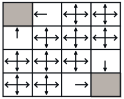
$k = 0$

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



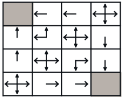
$k = 1$

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



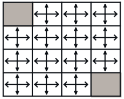
$$v_{k+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_k(s')\right]$$

Typo in example, -1.7 should be -1.75

Policy evaluation example

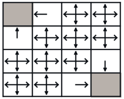
$k = 0$

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



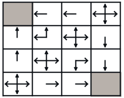
$k = 1$

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



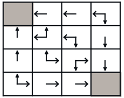
$k = 2$

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



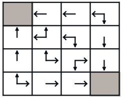
$k = 3$

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



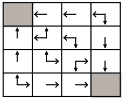
$k = 10$

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



Policy evaluation

- Given some policy π, we can **evaluate** π by determining v_π
  1. starting from arbitrary values v_0
  2. iterating, using the Bellman equation as an update rule
- v_π is a fixed point – it solves the Bellman equation
- Is used as subroutine to **improve** a policy π

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 $\Delta < \theta$

Contraction mapping theorem

- An operator $\mathcal{T} : \mathcal{V} \to \mathcal{V}$ on a normed vector space $\mathcal{V}$ is a γ–contraction, for $0 < \gamma < 1$, wrt to the norm $\|\cdot\|$ iff for all $x, y \in \mathcal{V}$:

$$\|\mathcal{T}x - \mathcal{T}y\| \leq \gamma \|x - y\|$$

Theorem (Contraction mapping)

for a γ–contraction $\mathcal{T}$ in a complete, normed vector space $\mathcal{V}$

- *The sequence $x, \mathcal{T}x, \mathcal{T}(\mathcal{T}x), \mathcal{T}(\mathcal{T}(\mathcal{T}x)) \ldots$ converges for every x*
- *$\mathcal{T}$ converges to a unique fixed point x_* in $\mathcal{V}$:*

$$\mathcal{T}x_* = x_*$$

Value function space

- Consider the vector space $\mathcal{V}$ over value functions
- $|\mathcal{S}|$ dimensions
- Each vector in the space fully specifies a value function v
- Bellman backups bring value functions closer together in this space
- ...and therefore the iterative update must converge to a unique solution

∞–norm

- We measure distances between value functions v and v' by the ∞–norm
- i.e. the largest difference between state values

$$\left\| v - v' \right\|_{\infty} = \max_{s} |v(s) - v'(s)|$$

Bellman backup: a contraction

- Bellman expectation backup operator $\mathcal{T}^{\pi}$:

$$(\mathcal{T}^{\pi}v)(s) = \sum_{a} \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v(s')\Big]$$

- This operator is a γ–contraction, i.e. it moves value function closer together
- The Bellman operator $\mathcal{T}^{\pi}$ has a unique fixed point
- v_{π} is *a* fixed point (Bellman equation)

$\Rightarrow$ Policy evaluation converges to v_{π}

Proof

$$
\begin{aligned}
&\max_s \left|(\mathcal{T}^\pi v)(s) - (\mathcal{T}^\pi v')(s)\right| \\
&= \max_s \Bigg| \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v(s')\Big] \\
&- \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v'(s')\Big]\Bigg| \\
&= \max_s \Bigg| \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v(s') - (r + \gamma v'(s'))\Big]\Bigg| \\
&= \gamma \max_s \Bigg| \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\Big[v(s') - v'(s')\Big]\Bigg| \\
&\leq \gamma \max_s \Bigg| \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \max_s \left|v(s) - v'(s)\right| \Bigg| \\
&= \gamma \max_s \Big| \max_s |v(s) - v'(s)| \Big| = \gamma \max_s \Big| v(s) - v'(s) \Big| = \gamma \left\|v - v'\right\|_\infty
\end{aligned}
$$

Policy improvement

- Given value function for policy π, how do we get an improved π'?
- In some state s, we can choose an action a that is *better* than $\pi(s)$
- Value of taking action a in state s under a policy π:

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

- If $q_\pi(s,a) > v_\pi(s)$ then choose a to **improve** the policy
- Apply for all states $s \in \mathcal{S}$

Policy improvement

- We can define a new policy π' which is *greedy* wrt v_π:

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a)\\ &= \arg\max_a \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma v_\pi(s')\Big]\end{aligned}$$

- What if $v_{\pi'}(s) = v_\pi(s)$ for all states?
  - $\pi' = \pi = \pi_*$
  - $\Rightarrow v_\pi = v_*$

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

Iterative Policy Improvement

Policy iteration

- Alternating policy **evaluation** and policy **improvement**
- Evaluate – Improve – Evaluate – Improve – Evaluate ...

$$\pi_0 \xrightarrow{\mathsf{E}} v_{\pi_0} \xrightarrow{\mathsf{I}} \pi_1 \xrightarrow{\mathsf{E}} v_{\pi_1} \xrightarrow{\mathsf{I}} \pi_2 \xrightarrow{\mathsf{E}} \dots \xrightarrow{\mathsf{I}} \pi_* \xrightarrow{\mathsf{E}} v_*$$

Policy iteration

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 $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   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 $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   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

Issues with full policy evaluation

- Each iteration of policy iteration requires policy evaluation to converge
  - possibly many *sweeps* through the state space
- policy evaluation often converges within a few sweeps
- we can truncate (reduce the number of sweeps) of policy evaluation
- Value iteration is extreme case: only one sweep of policy evaluation

Value iteration

- Just update values for *one* iteration and *immediately* improve policy
- Update rule:

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

- One-iteration update + policy improvement

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

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

Generalized Policy Iteration

Asynchronous dynamic programming

- So far: systematic sweeping all states
- An alternative is:
  - pick a state at random and apply the backup
  - repeat until converge criteria is reached
- Guaranteed to converge if all states continue to be selected
- Can you select states to backup intelligently?
- Asynchronous DP:
  - In–place dynamic programming
  - Prioritized sweeping
  - Real–time dynamic programming

Prioritized sweeping

- Use magnitude of *Bellman error* to guide state selection:

$$\left| \max_a \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma v(s') \Big] - v(s) \right|$$

- Backup the state with the largest remaining Bellman error
- Can be implemented efficiently by maintaining a priority queue

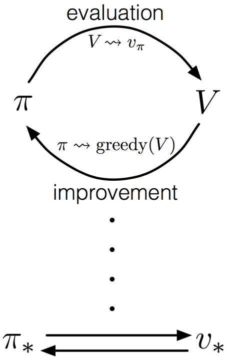
Real–time dynamic programming

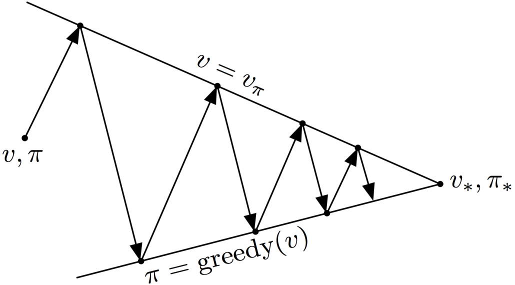
- Update only states that are relevant to agent
- Use agent's experience to guide the selection of states
- After each time-step $S_t = s, A_t = a, R_{t+1} = r$
- Backup for state s:

$$v(s) = \max_{a'} \sum_{s',r} p(s', r \mid s, a') \Big[r + \gamma v(s') \Big]$$

Generalised policy iteration

- Any interleaving of policy evaluation and policy improvement, independent of their granularity





Summary

- Policy iteration = policy evaluation + policy improvement
- Policy evaluation: backups without a $\max$, find the value function for a given policy
- Policy improvement: make policy greedy wrt value function (if only locally)
- Value iteration: backups with a max, i.e. Bellman optimality equation
- Asynchronous dynamic programming: avoids exhaustive sweeps through state space
- Generalised policy iteration: interleaving policy evaluation and improvement at any granularity