University of Stuttgart

Reinforcement Learning

Dynamic Programming

Daniel Hennes

08.05.2017

University Stuttgart - IPVS - Machine Learning & Robotics

Dynamic Programming

- Policy evaluation
- Policy improvement
- Policy iteration = policy evaluation + policy improvement
- Value iteration
- Asynchronous dynamic programming
- Generalised policy iteration

Recap: finite MDPs

- A **Markov Decision Process** is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, T)$
- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- p is a state transition probability function

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

- r is a reward function

$$r(s, a, s') = \mathbb{E}\Big[R_{t+1} | S_t = s, A_t = a\Big]$$

- $\gamma \in [0, 1)$ is a discount factor
- T is the final time step (possibly $T = \infty$)

Solving MDPs

- **Prediction:** given an MDP and a policy
  $\pi(a \mid s) = \Pr\{A_t = a \mid S_t = s\}$
  find the state and action–value functions
- **Optimal control:** given an MDP, find the optimal policy π_*
  (aka the planning problem)
- **Dynamic programming**
  needs perfect model $p(r, s' \mid s, a)$
  we want to 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:

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

- 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*)

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

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

- Bellman equation is used as an iterative backup update equation
- After many *sweeps*, iterative policy evaluation converges to v_*

Policy evaluation example

actions

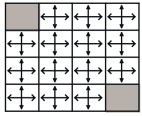
	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

Policy evaluation example

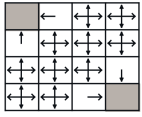
$k = 0$

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



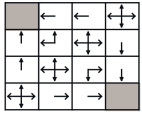
$k = 1$

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



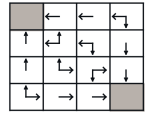
$k = 2$

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



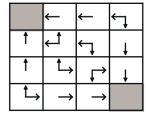
$k = 3$

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



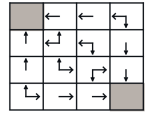
$k = 10$

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
- Note: for terminal states we require a state–value of 0)
- v_π is a fixed point – it solves the Bellman equation
- Next step: **improving** policy π

Contraction mapping theorem

- An operator $\mathcal{T}$ on a normed vector space $\mathcal{X}$ 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 in $\mathcal{V}$:

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

Value function space

- For the vector space $\mathcal{V}$ over value functions
- $|\mathcal{S}|$ dimensions
- Each point in the space fully specifies a value function v
- Bellman backup brings 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

$$\|v - v'\|_\infty = \max_s |v(s) - v'(s)|$$

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

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

Policy improvement

- What is policy improvement?
- 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:

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

- 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?
  - $\pi' = \pi = \pi_*$
  - $\Rightarrow v_\pi = v_*$

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

Policy iteration

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

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

Issues with policy evaluation

- In policy evaluation we need to keep v_k to calculate v_{k+1}
  - twice the memory requirement
- *In-place* updates
  - some updates already use updated v_k estimates
  - often converges faster than synchronized updates
- Each iteration of policy iteration requires policy evaluation to converge
  - possibly many *sweeps* through the state space

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

   repeat
   $\Delta \leftarrow 0$
   for each $s \in \mathcal{S}$ **do**
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \Big[r + \gamma V(s') \Big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   end for
   until $\Delta < \theta$ (a small positive number)
3. **Policy improvement**

   stable $\leftarrow$ *true*
   for each $s \in \mathcal{S}$ **do**
   old $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma V(s') \Big]$
   if *old* $\neq \pi(s)$ **then** *stable* $\leftarrow$ *false*
   end for
   if *stable* **then** stop and **return** $V \approx v_*$ and $\pi \approx \pi_*$ **else** go to 2

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

Initialization

$V(s) \in \mathbb{R}$ arbitrarily for all $s \in \mathcal{S}^+$

Policy evaluation

repeat

$\Delta \leftarrow 0$

for each $s \in \mathcal{S}$ **do**

$v \leftarrow V(s)$

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

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

end for

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that

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

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

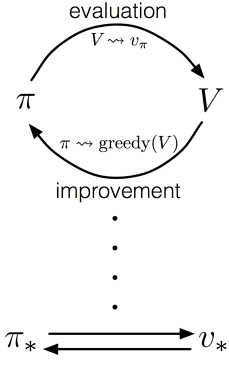
Real–time dynamic programming

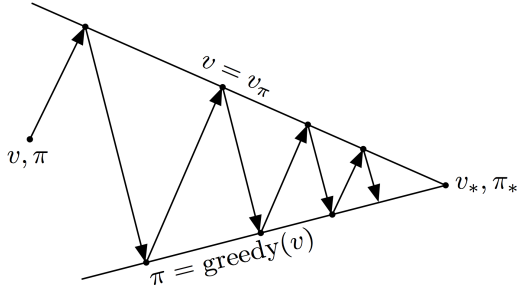
- Update only states that are relevant to agent
- Use agent's experience to guide the selection of states
- After each time-step $S_t = s, A_t = a, R_{t+1} = r$
- Backup for state s:

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

Generalised policy iteration

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





Bellman's optimality principle

- Any optimal policy can be subdevided into two components:
  - an optimal first action $A_t = a_*$
  - optimal policy from successor state $S_{t+1} = s'$
- **Theorem:** Optimality
  A policy $\pi(a \mid s)$ achieves optimal value from state s,
  $v_{\pi_*}(s) = v_*(s)$, iff
  for any state s' reachable from s, π achieves the optimal value
  from state s': $v_{\pi_*}(s') = v_*(s')$

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
- **Bootstrapping:** updating estimates based on other estimates
- **Full backups:** each backup takes into account all the states one can reach from the current state in calculating the backup