Reinforcement Learning – COMS4061A/7071A

Dynamic Programming

Prof. Benjamin Rosman
Benjamin.Rosman1@wits.ac.za / benjros@gmail.com

Based heavily on slides by Steve James

Sutton and Barto [2018], Chapter 4

Last week

- MDP: $\langle S, A, P, R, \gamma \rangle$

- Policies: $\pi(a \mid s)$ or $\pi : S \to A$

- Value Functions: $v_\pi(s), q_\pi(s, a)$

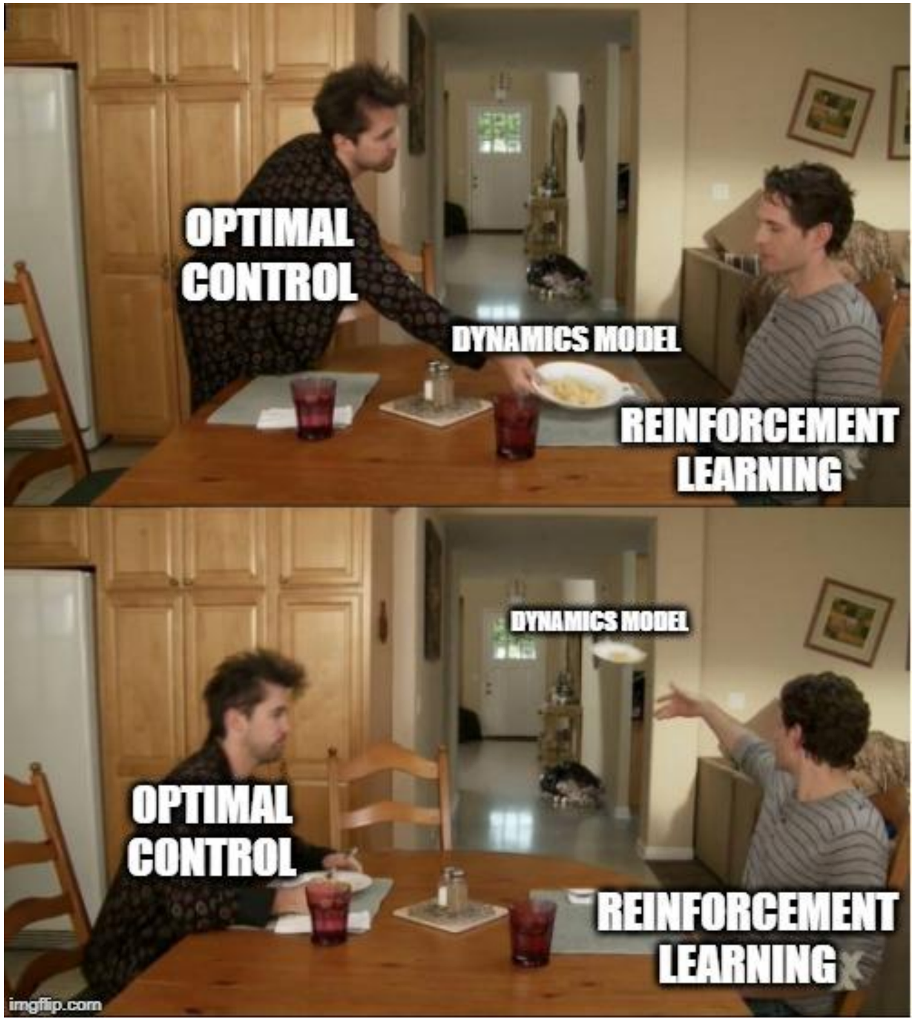
- Optimality: $v_*(s), q_*(s, a), \pi_*$

This week

- How do we compute these quantities?
  - (Optimal) value functions
  - (Optimal) policies

- We will assume known dynamics and reward function: $\langle S, A, P, R, \gamma \rangle$
  - Pros and cons?

- Later relax this condition


OPTIMAL CONTROL
DYNAMICS MODEL
REINFORCEMENT LEARNING
DYNAMICS MODEL
OPTIMAL CONTROL
REINFORCEMENT LEARNING
imgflip.com

Dynamic Programming

- Class of algorithms for problems with two properties

1) Optimal substructure
   - Can decompose into subproblems to solve optimally

2) Overlapping subproblems
   - Subproblems recur many times

Shortest Path?

					G

$$distance(s) = 1 + \min_{neighbours(s)} distance(s)$$

The Bellman Equation

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

1) Recursive decomposition

2) Value function stores and reuses solutions

Prediction vs control

- Prediction:
  - Given a policy, what will happen?
  - How much return will I get?
  - Policy evaluation $\rightarrow v_\pi / q_\pi$?
- Control:
  - How to act optimally?
  - Compute $v_* / q_* / \pi_*$

Policy evaluation

I already know the policy π

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

- Solve this iteratively.
  - Turn Bellman equation into an update rule

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

with v_0 arbitrary

Iterative policy evaluation

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

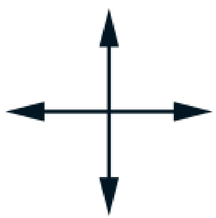
Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

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

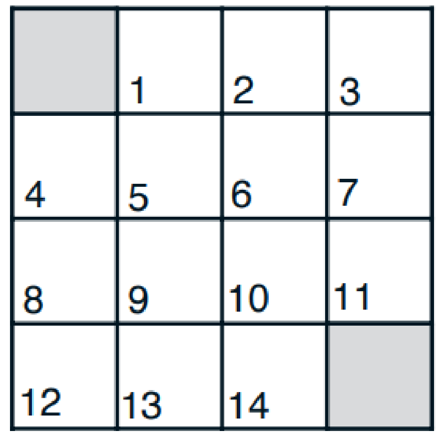
Known

Bellman update

Gridworld Example



actions



$R_t = -1$
on all transitions

- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states $1, \ldots, 14$
- One terminal state (shown twice as shaded squares)
- Agent follows uniform random policy

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

$k = 3$

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

Control

- How can we use v_π to learn optimal policies?

- For some state s, is it better to pick $a \neq \pi(s)$?
  - But for all other states stick with π?

- The value of picking this action a at s?

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(s_{t+1}) \mid S_t = s, A_t = a]$$
$$= \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')]$$

better than $v_\pi(s)$?

Policy improvement

- Consider new policy π' where $q_\pi(s, \pi'(s)) \geq \; v_\pi(s)$
- Then $v_{\pi'}(s) \geq v_\pi(s)$ (Policy Improvement thm)
- Simplest case: imagine $\pi' = \pi$ except at state s
  - If it is better to pick action according to π' here, then $\pi'$is as good as or better than π: $v_{\pi'} \geq v_\pi$

Policy improvement

- Extend to all states and all actions
- Consider $\pi'(s) = \operatorname{argmax}_a q_\pi(s, a)$
- By construction, $q_\pi(s, \pi'(s)) \geq v_\pi(s)$

$$\Rightarrow v_{\pi'}(s) \geq v_\pi(s)$$

- So the new policy π' is at least as good as π

Policy improvement

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

- This is a one-step lookahead according to v_π
- If $v_{\pi'} = v_\pi$, then from above we have

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

Bellman optimality eqn!

Summary of policy improvement

- Given a policy π, compute π' by acting greedily w.r.t. v_π

- π' is better or equal to π

- If $\pi' = \pi$ or $v_{\pi'} = v_\pi$, then π' must be optimal

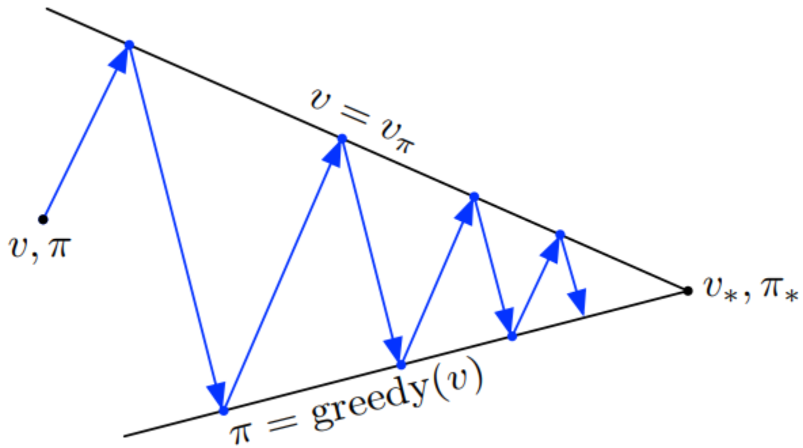
Policy iteration

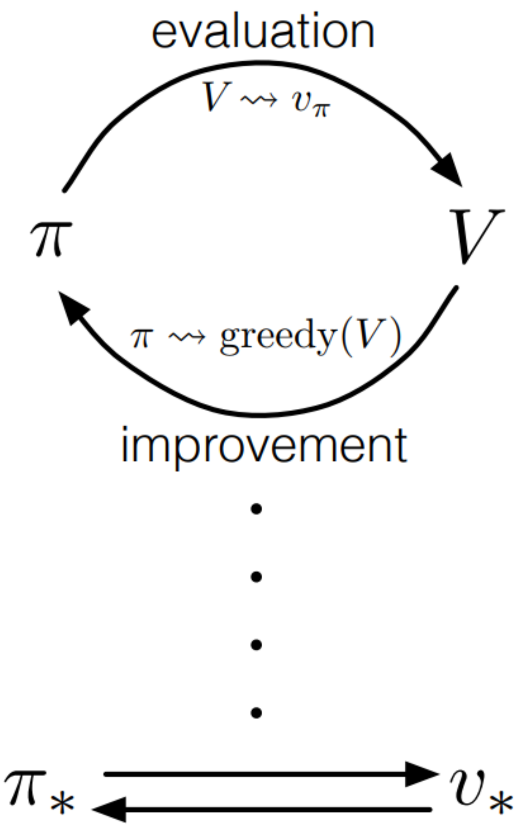
- This gives us a way of learning an optimal policy:

1. Start with an arbitrary policy
2. Compute its value function (policy evaluation)
3. Compute a new greedy policy w.r.t the value function (policy improvement)
4. Go to step 2 with the new policy

- Guaranteed to converge to optimal policy

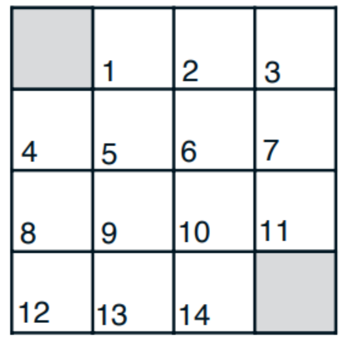
Policy iteration



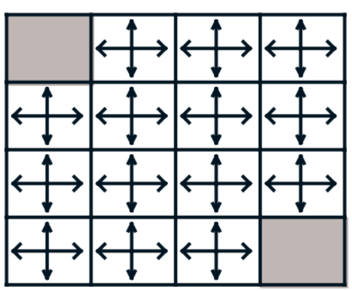


- Policy evaluation
  - Estimate value function
- Policy improvement
  - Act greedily

Policy iteration (Gridworld)



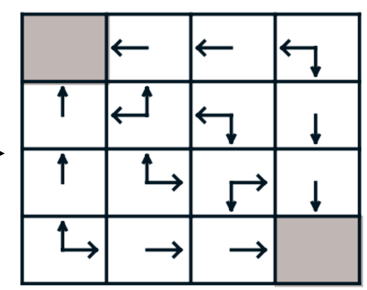
$R_t = -1$
on all transitions



Eval

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

Improve



Early stopping?

$k = 0$

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

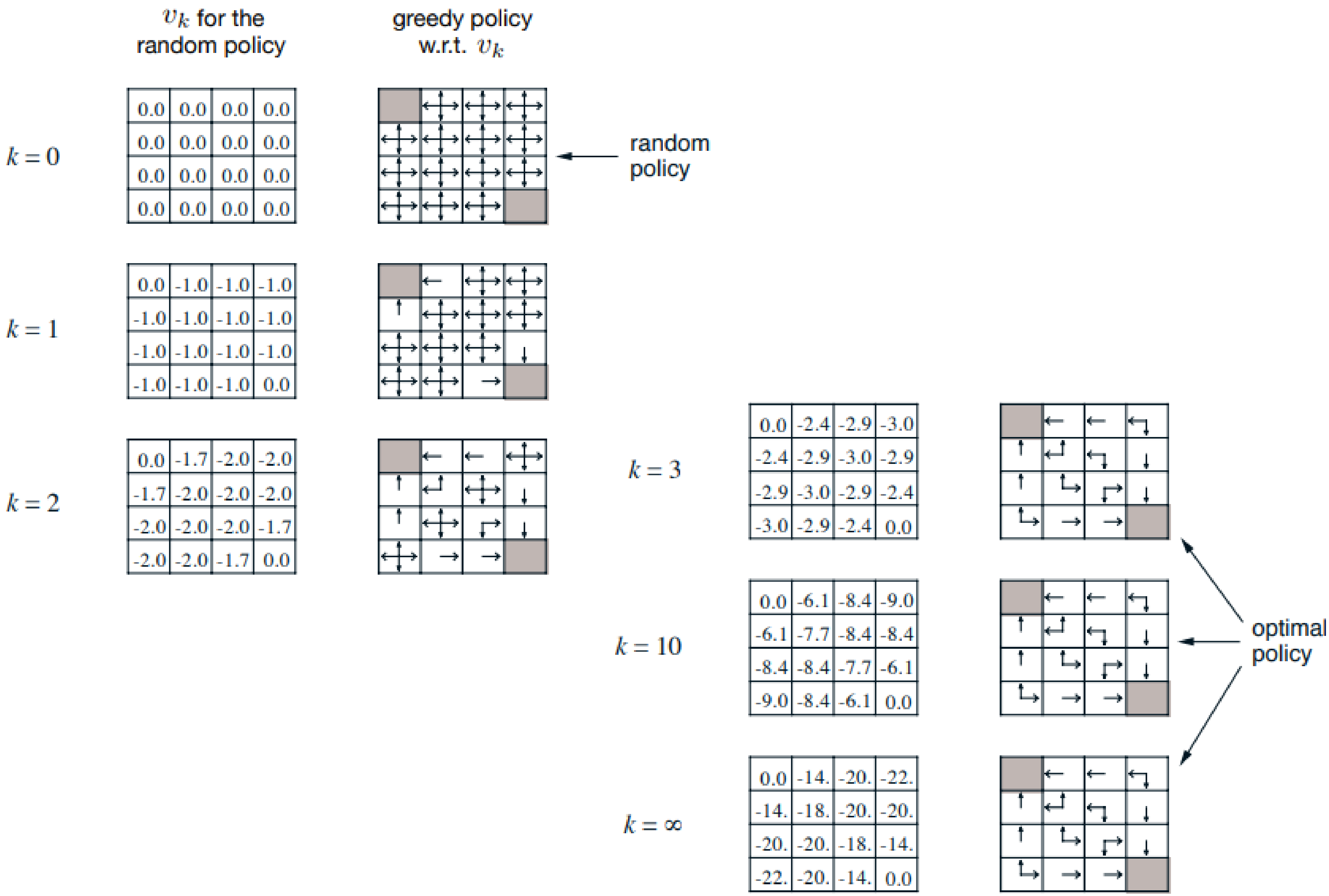
$k = 10$

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

We evaluated the policy,
then acted greedily.
Must we wait?



$k = 0$

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

Value Iteration

- One iteration of policy evaluation (one update of each state)
- Followed by greedy improvement step

- Combine these into a single update state
  - No explicit policy!

Value iteration

- Policy evaluation:
  $v_{k+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$
- Policy improvement:
  - $\pi'(s) = \mathrm{argmax}_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')]$

Combine into:

- Value iteration:
  - $v_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$

Value iteration

- Combines one sweep of policy evaluation and policy improvement
- Converges to v_*

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

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$

Complexity

- For both policy evaluation and value iteration, a single iteration is $O(|S|^2|A|)$
  - Number of iterations is exponential in the discount factor

- Policy iteration is strongly polynomial in the number of state-action pairs (Ye, 2011)
  - Value iteration is not (Feinberg and Huang, 2014)

Y. Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research* 36.4 (2011): 593-603.

E. Feinberg, , and J. Huang. The value iteration algorithm is not strongly polynomial for discounted dynamic programming. *Operations Research Letters* 42.2 (2014): 130-131.

Also see https://rltheory.github.io/lecture-notes/planning-in-mdps/lec4/

Asynchronous DP

- So far, all states are updated evenly
- But what if some states are more important than others?
- Asynchronous DP backs up states in an order
  - Reduce computation
- Converges if all states continue to be selected in the limit

Prioritised Sweeping

- Backup the state with the largest remaining Bellman error (use priority queue)

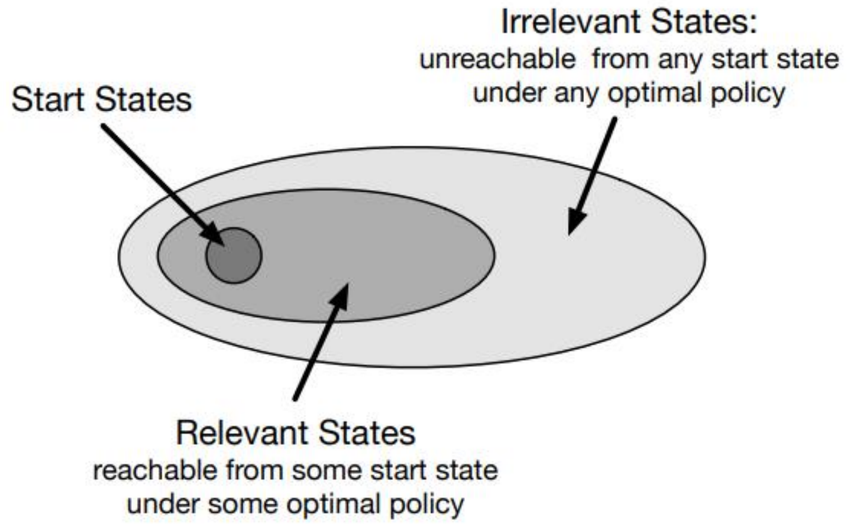
$$\left| \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v(s')] - v(s) \right|$$

- Recompute error of affected states each time

Real-Time DP

- Update only states relevant to agent

- e.g. Keep track of all states ever visited by the agent
  - Update only those states



Summary

- Given full knowledge of the MDP, we can compute the optimal policy with dynamic programming
- Policy iteration interleaves policy evaluation with policy improvement
  - In practice, preferred to value iteration
- Can be used to solve MDPs with millions of states
  - Turns out "millions" isn't that big
- Next lecture will look at when we do not know the dynamics (TD learning)

Homework

- Modify the gridworld you created last week to create a 4x4 version
  - No obstacles
  - Rewards of -1 on all transitions
  - Goal in the top left corner – entering the goal state ends the episode
- Given this environment and a uniform random policy, implement 2 versions of policy evaluation
  - The in-place version, as presented in the book
  - A two-array version, which only updates the value function after looping through all states (see pg 75)
  - Use a threshold value of $\theta = 0.01$
  - NB: Careful of how you handle the terminal state!!
- For a given γ, record the number of iterations of policy evaluation until convergence

By next week's lecture, submit on Moodle (groups of up to 4):

1. A 2d heatmap plot of the value function for $\gamma = 1$
2. A combined plot of both versions of policy evaluation for **different discount rates**
   1. The x-axis should be the discount rate. The range of discounts should be specified by np.logspace(-0.2, 0, num=20)
   2. The y-axis should be the number of iterations to convergence
3. Your code