Markov Decision Process

CS4246/CS5446

AI Planning and Decision Making

Please join:
pollev.com/anarayan

This lecture will be recorded!

Topics

- Markov Decision Process (16.1)
  - Model formulation and solution
  - Bellman Equation and Q-function
- Algorithms for solving MDPs
  - Value iteration (16.2.1)
  - Policy iteration (16.2.2)
  - Online algorithms and Monte Carlo Tree Search

Markov Decision Process (MDP)

- Formally:
  - An MDP $M \triangleq (S, A, T, R, \gamma)$ consists of:
  - A set S of states
  - A set A of actions
  - A transition function $T: S \times A \times S \rightarrow [0,1]$ that satisfies the Markov property such that:

$$\forall s \in S, \forall a \in A: \sum_{s' \in S} T(s, a, s') = \sum_{s' \in S} P(s' | s, a) = 1$$

  - A reward function $R: S \rightarrow \Re$ or $R: S \times A \times S \rightarrow \Re$
  - A discount factor $0 < \gamma < 1$
  - Solution is a policy – a function to recommend an action in each state: $\pi: S \rightarrow A$
    - Solution involves careful balancing of risk and reward

Solving MDPs

- Solution: A policy $\pi(s): S \rightarrow A$ is a function from states to actions
- Factors influencing the solution
  - Horizon: Finite vs Infinite
  - Reward function
- Utility of a state & Bellman equation
  - $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U(s')$
- Q-function
  - Computing the optimal policy: $\pi^*(s) = \arg\max_a Q(s,a)$

Value Iteration

Solution Method

Bellman Equation and Value Iteration

- Value iteration (A solution algorithm for MDP)
  - Bellman equation is basis of the value iteration algorithm for solving MDPs based on MEU
  - $|S|$ nonlinear equations (due to max) with $|S|$ unknowns (utility of states).
  - Iterative approach to solve Bellman equations
  - Propagating information through state space by means of local updates
  - Solutions are unique solutions

Value Iteration

- Repeatedly perform the Bellman Update

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U_i(s')] \quad \text{OR}$$

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

- Simultaneous updates of all states:
  - $|S|$ nonlinear equations (due to max) with $|S|$ unknowns (utility of states).
- Guaranteed to reach an equilibrium through convergence
- Final utility values must be unique solutions to the Bellman equations
- Corresponding policy is optimal

$$\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

Value Iteration Algorithm

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s, a, s')$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum relative change in the utility of any state

repeat
  $U \leftarrow U'$; $\delta \leftarrow 0$
  for each state s **in** S **do**
    $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U) $= \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$
    if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta \leq \epsilon(1-\gamma)/\gamma$
return U

Bellman update

Termination condition

Source: RN Figure 16.6

Example: Solving Bellman Equations

$R(s) = -0.04$

Numbers are $U(s)$

How to compute the numbers?
What is the optimal policy?

3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	−1
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

Assume $\gamma = 1$

For $U(1,1)$: $$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U_i(s')]$$

$$
\begin{aligned}
\max\{ \; & [0.8(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,1))], && \text{(U)} \\
& [0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(1,2))], && \text{(L)} \\
& [0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(2,1))], && \text{(D)} \\
& [0.8(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(1,1))]\} && \text{(R)}
\end{aligned}
$$

$\pi^*((1,1)) = \text{Up}$

Source: RN Figure 16.3

Some Terminology

- Definitions:
  - A fixed point is any input unchanged by a function
  - A contraction is a function, when applied to two inputs, produces two outputs that are closer together
    - E.g., “divide by two” is a contraction
    - A contraction has only one fixed point
    - When contraction is applied to any argument, the value must get closer to the fixed point; repeated application reaches fixed point in the limit
  - Max norm or distance between two vectors is the maximum difference between any two corresponding elements
    - It measures “length” of a vector by the absolute value of its biggest component

Convergence

Bellman update operator

- Value iteration:
  - Converges to the (unique) utility function for discounted problems with $\gamma < 1$
- The Bellman update $U_{i+1} \leftarrow BU_i$, is a contraction by a factor of γ on the space of utility vectors:

$$\|BU_i - BU_i{}'\| \leq \gamma \|U_i - U_i{}'\|$$

  - where max norm $\|U\| = \max_s U(s)$; $\|U - U'\| = \max_s \|U(s) - U'(s)\|$
- For Bellman equations, the utility or value function U is a fixed point

$$U = BU$$

Convergence

- Repeated application of a contraction reaches a unique fixed point U

  $\|U_{i+1} - U\| = \|BU_i - BU\| \leq \gamma\|U_i - U\| \leq \gamma^i\|U_0 - U\|$ for any initial U_0

- Reaching fixed point:
  - $U = BU$ is the fixed point utility function
  - Error $\|U_i - U\|$ reduced by a factor of at least γ on each iteration; converges exponentially fast
  - Utilities of all states bounded by $\pm R_{max}/(1-\gamma)$:

$$\|U_0 - U\| \leq 2R_{max}/(1-\gamma)$$

Convergence

- Factors influencing convergence:
  - N iterations to reach error of at most ϵ:

$$\|U_N - U\| \leq \gamma^N \cdot 2R_{max}/(1-\gamma) \leq \epsilon$$

$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

- Termination condition:
  - If the update is small (i.e., no state utility changes by much), then the error, compared with the true utility function, also is small

$$\text{if} \quad \|U_{i+1} - U_i\| < \frac{\epsilon(1-\gamma)}{\gamma} \quad \text{then} \quad \|U_{i+1} - U\| < \epsilon$$

Convergence: Calculations

Remember: Values at each state bounded by $\pm\frac{R_{max}}{1-\gamma}$
The 2 in the numerator reflects the bounds

- If we run N iterations, we get:

Error

$$\|U_N - U\| \le \frac{\gamma^N R_{max}}{1-\gamma}$$

- To get error at most ϵ, we have:

$$\frac{\gamma^N R_{max}}{1-\gamma} \le \epsilon$$

$$\gamma^N R_{max} \le \epsilon(1-\gamma)$$

$$\frac{R_{max}}{\epsilon(1-\gamma)} \le \frac{1}{\gamma^N} = \left(\frac{1}{\gamma}\right)^N$$

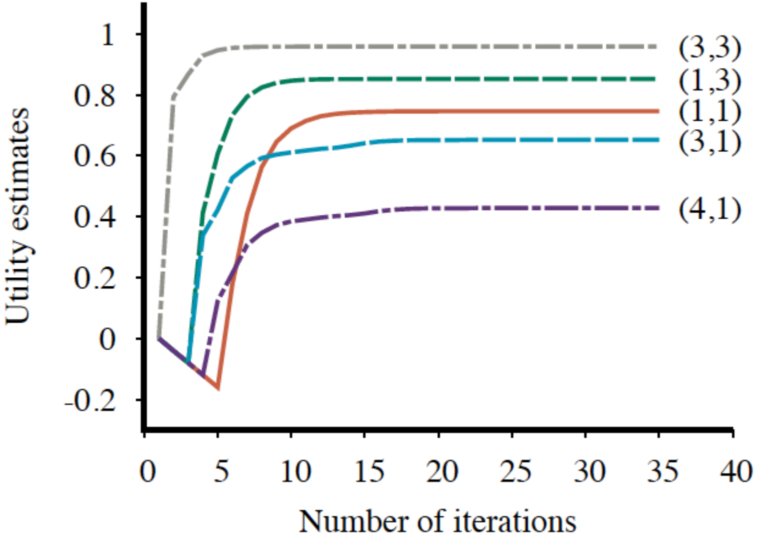
- Take log:

$$N \log\left(\frac{1}{\gamma}\right) \ge \log\left(\frac{R_{max}}{\epsilon(1-\gamma)}\right)$$

$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

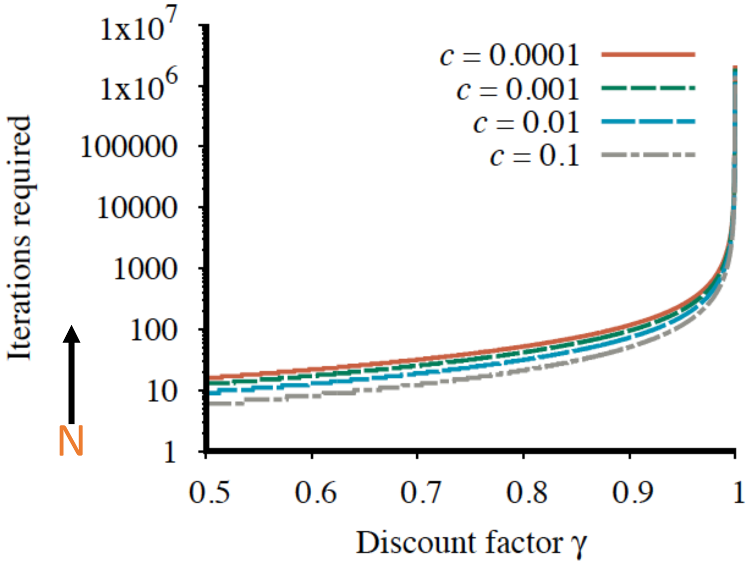
- Terminating condition:
  - $\|U_{t+1} - U_t\| \le \epsilon(1-\gamma)/\gamma$
  - $\Rightarrow \|U_{t+1} - U\| \le \epsilon$

Example: Convergence



(a)

Evolution of the utilities of selected states using VI



(b)

N: No. of iterations required to guarantee an error of at most $\epsilon = c.R_{max}$, for different values of c, as a function of γ

Source: RN Figure 16.7

Computing Optimal Policy

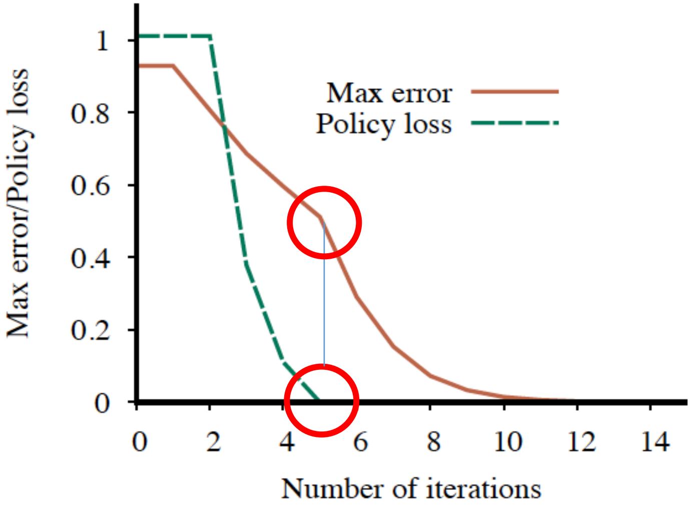
Would an MEU policy based on the estimated utilities behave optimally?

- Policy Loss
  - Let π_i be the MEU policy wrt U_i
  - Recall: $U^{\pi_i}(s)$is utility obtained if π_i is executed starting in state s
  - Policy loss $||U^{\pi_i} - U||$ is the max loss by following π_i instead of π^*.
- Connecting utility error and policy loss

$$\text{if} \quad \|U_i - U\| < \epsilon \qquad \text{then} \quad \|U^{\pi_i} - U\| < 2\epsilon$$

- In practice:
  - π_i often becomes optimal long before convergence of U_i

Example: Computing Optimal Policy



Maximum error $\|U_i - U\|$ of the utility estimates and the policy loss $\|U^{\pi_i} - U\|$, as a function of the number of iterations of VI on the 4×3 world

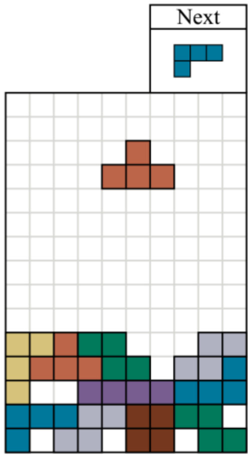
Source: RN Figure 16.8

Summary

- Value iteration
  - Value iteration converges to the correct utilities
  - Can bound the errors in utility estimates if stop after a finite number of iterations
  - Can bound the policy loss from executing the corresponding MEU policy
  - All the results depend on $\gamma < 1$; or similarly derived if $\gamma = 1$ and there are terminal states (i.e., in the case of proper policies)

Exercise: Tetris

- Question:
  - Can you solve Tetris using value iteration?

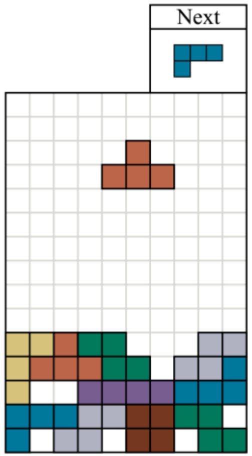


Quiz

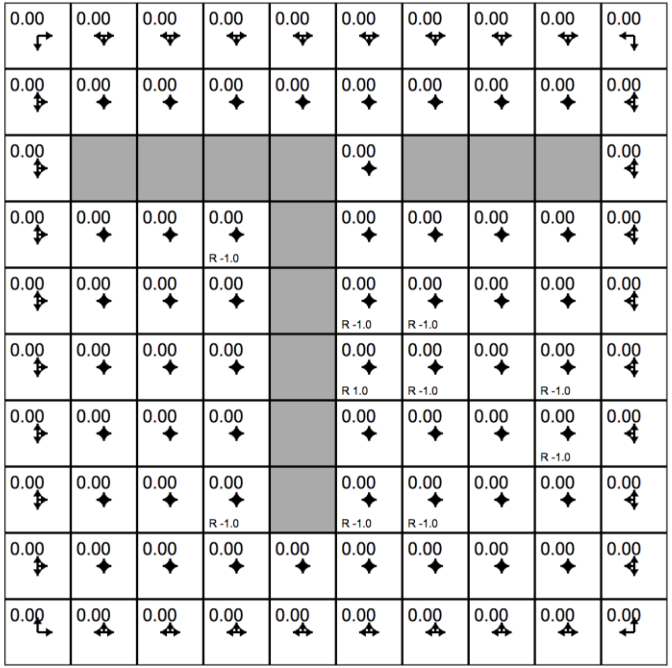
Quiz answer

Exercise: Tetris

- Question:
  - Can you solve Tetris using value iteration?
- Answer:



Value Iteration Demo



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Policy Iteration

Solution Method

Motivation for Policy Iteration

- Value iteration:
  - Exact algorithm
  - Scales poorly
- Observations:
  - If one action is clearly better than all others, then exact magnitude of utilities on the states need not be precise
  - Utility function estimate can be inaccurate to derive optimal policy
  - There is another way to find optimal policies: Policy Iteration

Policy Iteration

- Begin with Initial policy: π_0
- Alternate between:

  Policy evaluation:

  - Given a policy π_i, calculate $U_i = U^{\pi_i}$, utility of each state if π_i is executed

  $$U_i(s) = \sum_{s'} P(s'|s, \pi_i(s))[R(s, \pi_i(s), s') + \gamma U_i(s')]$$

  Policy improvement:

  - Calculate new MEU policy π_{i+1} using one-step look-ahead based on U_i.
- Terminate when there is no change in utilities for policy improvement step
  - Reaching fixed point of the Bellman update
  - Finite number of policies, hence algorithm must terminate
- Complexity:
  - Assuming $|S| = n$ linear equations with n unknows – can be solved in $O(n^3)$ time.

Policy Iteration Algorithm

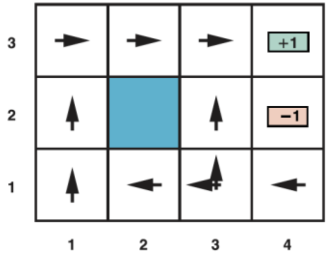
function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      $a^* \leftarrow \underset{a \in A(s)}{\operatorname{argmax}}$ Q-VALUE(mdp, s, a, U) $= \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$
      if Q-VALUE(mdp, s, a^*, U) $>$ Q-VALUE($mdp, s, \pi[s], U$) **then**
        $\pi[s] \leftarrow a^*$; $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Policy Improvement

Source: RN Figure 17.9

Example: Implementing Policy Evaluation



- Policy evaluation

$$U_i(s) = \sum_{s'} P(s'|s, \pi_i(s))[R(s, \pi_i(s), s) + \gamma U_i(s')]$$
$$= R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$

No max operator

$$U_i(1,1) = 0.8[-0.04 + U_i(1,2)] + 0.1[-0.04 + U_i(2,1)] + 0.1[-0.04 + U_i(1,1)]$$
$$U_i(1,2) = 0.8[-0.04 + U_i(1,3)] + 0.2[-0.04 + U_i(1,2)]$$
$$\vdots$$

- Policy evaluation equations are linear equations
  - Simplified Bellman equation
  - For n states, can be solved in $O(n^3)$ time
  - For large state-spaces, use iterative method:

$$U_{t+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s'|s, a) U_t(s')$$

Similar, but simpler than Bellman equations

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s, a) U(s')$$

Source: RN 16.2.2

Example: Implementing Policy Improvement

- Policy improvement:
  - For all the states, to find best policy:

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

  - In state (1, 1), compute:
  - If any action is better than $\pi_{old}(1,1)$, update

$$\pi_{new}(1,1) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P((s'|(1,1),a)U(s')$$

$$\sum_{s'} P((s'|(1,1),U)U(s') = 0.8U(1,2) + 0.1U(1,1) + 0.1U(2,1) = \ldots$$

$$\sum_{s'} P((s'|(1,1),L)U(s') = \ldots$$

$$\sum_{s'} P((s'|(1,1),R)U(s') = \ldots$$

$$\sum_{s'} P((s'|(1,1),D)U(s') = \ldots$$

Policy Iteration: Why Does It Work?

- Termination
  - Policy improvement step yields no change in utilities
  - Utility functionU_i is a fixed point of Bellman update, and a solution to the Bellman equations
  - π_i must be an optimal policy
  - Only finitely many policies for a finite state space, each iteration can be shown to yield a better policy, hence policy iteration must terminate
- Correctness
  - Follows from Policy Improvement Theorem [SB 4.2]

Policy Improvement Theorem

- Let:
  - $Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U^{\pi}(s')$ be the action-utility function of π (one-step look-ahead using action a)
- Theorem: [Proof in SB 4.2]
  - Let π and π' be any pair of deterministic policies such that for all $s \in S$,

$$Q^{\pi}(s, \pi'(s)) \geq U^{\pi}(s)$$

Recollect:
$U(s) = \max_a Q(s,a)$

  - Then

$$U^{\pi'}(s) \geq U^{\pi}(s) \text{ for all } s \in S$$

  - If the inequality of $Q^{\pi}(s, \pi'(s))$ is strict for any state s, then corresponding inequality for $U^{\pi'}(s)$ is strict for that s

Source: Sutton and Barto, 2018/2020 Section 4.2

Policy Improvement theorem – Proof

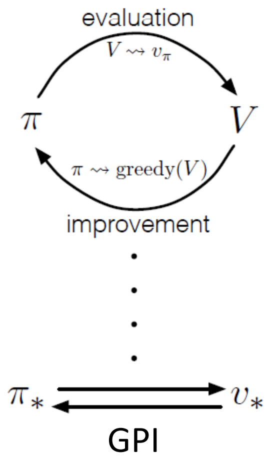
- Start with $Q^\pi\big(s,\pi'(s)\big) \geq U^\pi(s)$ and keep expanding the policy

$$U^\pi(s) \leq Q^\pi\big(s,\pi'(s)\big)$$
$$= R(s) + \gamma \textstyle\sum_{s'} P(s'|s,\pi'(s))U^\pi(s)$$
$$= E_{\pi'}[R(S^t) + \gamma U^\pi(S^{t+1})|S^t = s]$$
$$\leq E_{\pi'}[R(S^t) + \gamma E_{\pi'}[R(S^{t+1}) + \gamma U^\pi(S^{t+2})|S^{t+1}]|S^t = s]$$
$$= E_{\pi'}[R(S^t) + \gamma R(S^{t+1}) + \gamma^2 U^\pi(S^{t+2})|S^t = s]$$
$$\leq E_{\pi'}[R(S^t) + \gamma R(S^{t+1}) + \gamma^2 R(S^{t+2}) + \gamma^3 U^\pi(S^{t+3})|S^t = s]$$
$$\vdots$$
$$\leq E_{\pi'}[R(S^t) + \gamma R(S^{t+1}) + \gamma^2 R(S^{t+2}) + \gamma^3 R(S^{t+3}) + \cdots|S^t = s]$$
$$= U^{\pi'}(s)$$

- If the first inequality is strict, we have $U^\pi(s) < U^{\pi'}(s)$ □

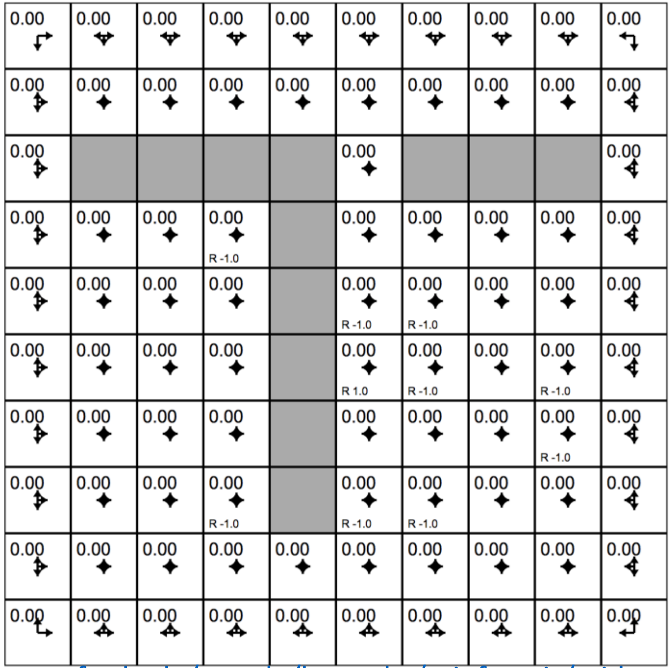
Source: Sutton and Barto, 2018/2020 Section 4.2

General Policy Iteration



- Modified policy iteration:
  - Do only k iterations of (simplified) value iteration instead of reaching convergence
  - Approximate evaluation
- Asynchronous policy iteration:
  - Pick only a subset of states for policy improvement or for updating in policy evaluation
  - Converges as long as continuously update all states
- Generalized policy iteration: (SB) 4.6
  - Update utility according to policy, and improve policy wrt the utility function.
  - VI, PI, asynchronous policy iteration are all special cases.
  - May interleave in different ways based on conditions

Policy Iteration Demo



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

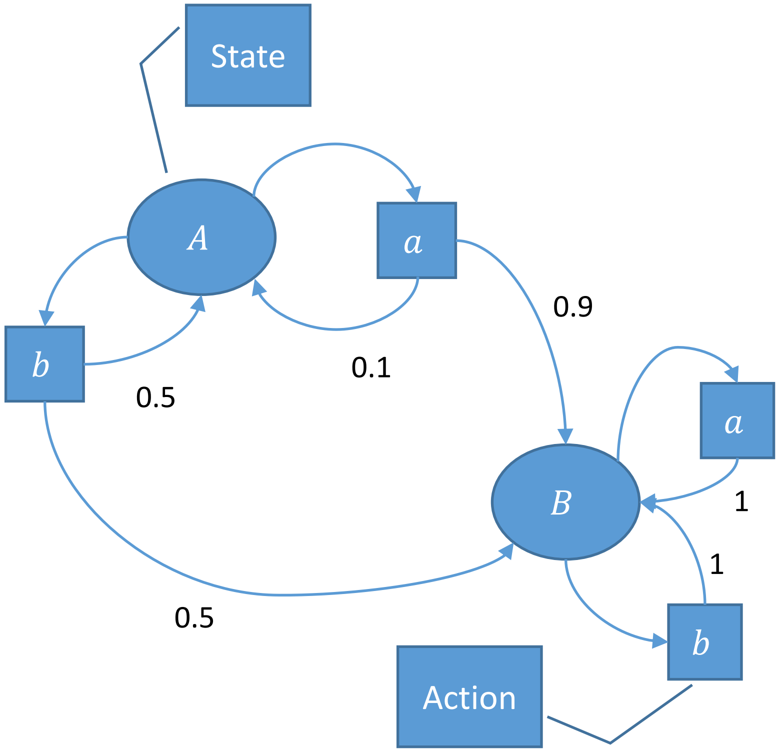
Homework

- Readings
  - [RN] 16.1, 16.2.1, 16.2.2
  - [SB] *4.2 (Policy improvement)*

  - [SB] Sutton, R. S. and A. G. Barto. Reinforcement Learning: An introduction. 2nd ed. MIT Press, 2018, 2020
    [Book website: http://incompleteideas.net/book/the-book.html]
    [e-Book for personal use: http://incompleteideas.net/book/RLbook2020.pdf]

MDP Representation as State Transition Graph

- For a 2-state MDP (States: $\{A, B\}$), with actions $\{a, b\}$, and the following transition function:
  - State A:
    - a takes to B with 90% probability; remain at A with 10% probability
    - b takes to B with 50% probability; remain at A with 50% probability
  - State B:
    - Both actions will loop with 100% probability
- We can represent the MDP as a state transition graph as shown alongside



Value iteration analysis (a detailed view)

- Converges to the (unique) value function for discounted problems with $\gamma < 1$
- We can show that the Bellman update $U_{t+1} \leftarrow BU_t$ is a contraction
- Mainly: $\|BU - BU'\| \leq \gamma \|U - U'\|$ at some iteration t
  - Here the **max** norm is used: $\|U\| = \max_s |U(s)|$
  - Distance between U & U' is the maximum difference between any two corresponding elements
  - A contraction, when applied to two inputs, produces two outputs that are closer together
- The Bellman operator, B, is a contraction by a factor γ

This is the result that proves convergence.

- Repeated application of a contraction reaches a unique fixed point U
  $\|BU_t - BU\| \leq \gamma \|BU_{t-1} - U\| \leq \gamma^t \|U_0 - U\|$
- In the inequality, let's use U, **the true utility**, or the unique fixed point
- Hence, when you apply B to U, you don't get any reduction (as it is already the fixed point)
- Hence, $\|BU_{t-1} - BU\| = \|BU_{t-1} - U\|$
- $\|U_0 - U\|$ is the initial error
- The Bellman operator applied to time t (or t^{th} iteration), results in the difference being lesser than the $(t-1)^{\text{th}}$ iteration, which is showed in the first inequality.

Value iteration analysis (a detailed view)

- Repeatedly applying the Bellman operator and using U, the unique fixed point, we can show that $\|BU_t - BU\| \leq \gamma^t \|U_0 - U\|$, where U_0 is the initial estimate.
- Hence the value function converges exponentially
- Values at each state bounded by $\pm\frac{R_{max}}{1-\gamma}$
  - $\because$ Max possible for U at any state = $R_{max} + \gamma R_{max} + \gamma^2 R_{max} + \cdots$
- Hence, if U_0 is initialized to 0, then $\|U_0 - U\| \leq \frac{2R_{max}}{1-\gamma}$
- If we run N iterations, we get

$$\|U_N - U\| \leq \frac{\gamma^N 2R_{max}}{1-\gamma}$$

- To get error at most ϵ, we have $\frac{\gamma^N 2R_{max}}{1-\gamma} \leq \epsilon$, giving

$$\gamma^N 2R_{max} \leq \epsilon(1-\gamma)$$

$$\frac{2R_{max}}{\epsilon(1-\gamma)} \leq \frac{1}{\gamma^N} = \left(\frac{1}{\gamma}\right)^N$$

Take log.

$$N \log\left(\frac{1}{\gamma}\right) \geq \log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)$$

$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

- Terminating condition in the pseudocode comes from the fact that $\|U_{t+1} - U_t\| \leq \epsilon(1-\gamma)/\gamma \Rightarrow \|U_{t+1} - U\| \leq \epsilon$