CS 5100 02/20

Recall

name of the algorithm we talked about last time

(Value Iteration) Algorithm

Iterative process

$$U_{i+1}(s) = \underbrace{R(s) + \max_{a} \sum_{s'} P(s'|s,a)\, U_i(s')}_{g(U_i(s))}$$

g is a contraction

$$\|g(\vec{U}) - g(\vec{U}')\|_\infty < \|\vec{U} - \vec{U}'\|_\infty$$

$$\boxed{\|g(\vec{U}_{i+1}) - g(\vec{U}_i)\|_\infty \le \|\vec{U}_{i+1} - \vec{U}_i\|_\infty}$$

$$\| \vec{U}_{i+2} - \vec{U}_{i+1} \|_\infty \leq \| \vec{U}_{i+1} - \vec{U}_i \|_\infty$$

difference between consecutive terms of the algorithm is becoming smaller and smaller

$\Rightarrow$ convergence

Optimal solution: $\vec{U} = g(\vec{U})$

need to make sure that it has only one solution.

Let's show that $\vec{U} = g(\vec{U})$ has a unique solution.

Assume by contradiction that there are two solutions to the equation

$\vec{U}_1 = g(\vec{U}_1)$ $\qquad$ $\vec{U}_2 = g(\vec{U}_2)$ with $\vec{U}_1 \neq \vec{U}_2$

$$\| g(\vec{U}_1) - g(\vec{U}_2) \|_\infty < \| \vec{U}_1 - \vec{U}_2 \|_\infty$$

$$\| \vec{U}_1 - \vec{U}_2 \|_\infty$$

no way
this cannot happen
$(1 < 1)$

$\Rightarrow$ contradiction

$\Rightarrow$ we can't have more than one solution

$\Rightarrow$ $\boxed{\text{uniqueness}}$

$$S_1 \to S_2 \to S_3 \to \cdots \to S_n$$

$$\text{Utility} = R(S_1) + R(S_2) + \cdots + R(S_n)$$

additive reward

There are examples in which you might not want to put the same weight

incorporate discounts in rewards in future states

γ: discount rate

Utility $= R(S_1) + \gamma R(S_2) + \gamma R(S_3) \ldots + R(S_n)$

$\gamma = 1 \longrightarrow$ get back additive rewards (same importance on Rewards at any state)

$\gamma = 0 \longrightarrow$ no reward from any successor state

small $\gamma \rightarrow$ less weight on successor states

large $\gamma \longrightarrow$ higher weight on successor states

$\gamma = \frac{1}{2}$ $\quad \gamma^2 = \frac{1}{4}$

$\gamma = \frac{1}{10}$ $\quad \gamma^2 = \frac{1}{100}$

Exercise

Consider a Markov Decision process (MDP)

Three states

discount rate $= 1$

$s=1$ $\quad s=2$ $\quad s=3$ terminal state

$R(s=1)=-1$ $\quad R(s=2) = -2$ $\quad R(s=3)=0$

In states $s=1$, $s=2$, there are two possible actions:

Action a:

* State $s=1$ $\xrightarrow{\text{proba}=0.8}$ State 2

You'll stay in state 1 with proba 0.2

* State $s=2$ $\xrightarrow{\text{proba}=0.8}$ State 1

you'll stay in state 2 with proba 0.2

Action b

In either state $s=1$ or $s=2$

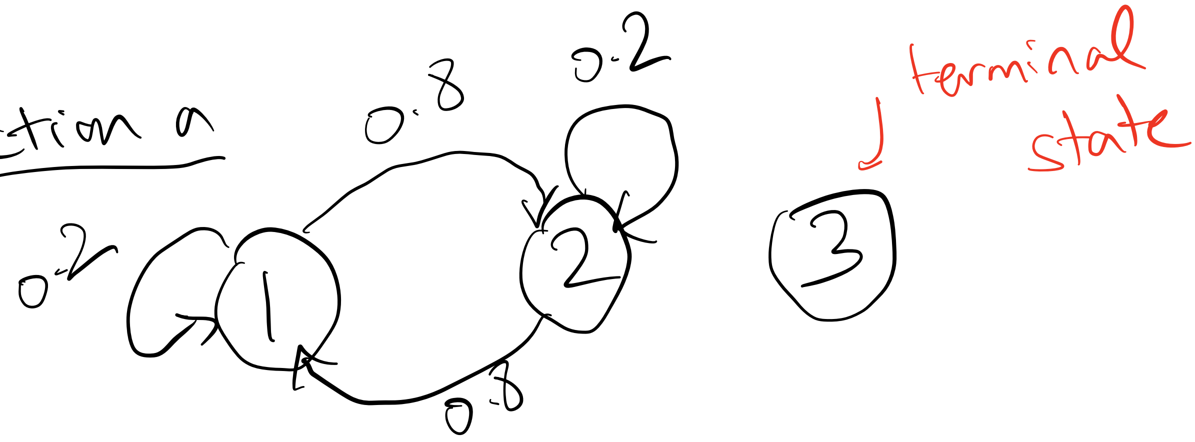
you will move to state 3 with proba 0.1 and stay put with proba 0.9

Compute the optimal utilities at states $s=1$ and $s=2$ for both actions a and b.

What happens with action a?

What happens if we implement discounting?

Action a



Action a

$$U^*(s) = \sum_{s'} P(s'|s,a)\left[R(s') + U^*(s')\right]$$

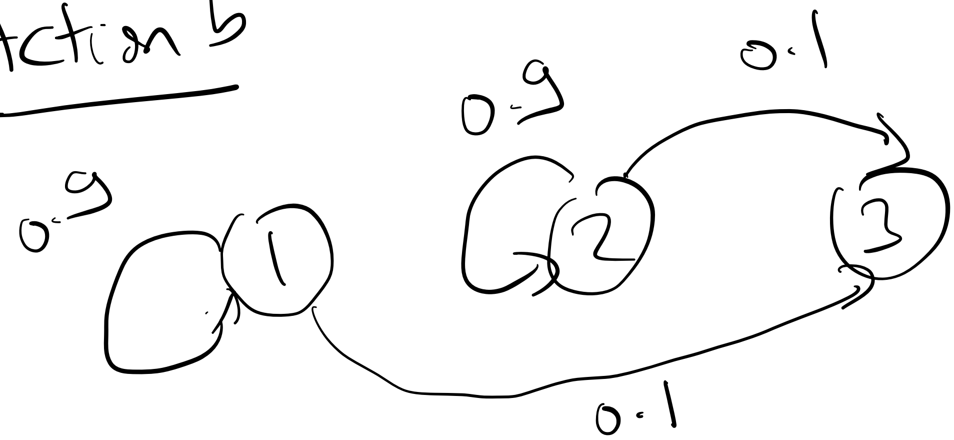
$$\begin{cases} U^*(1) = 0.8\left[-2 + U^*(2)\right] + 0.2\left[-1 + U^*(1)\right] \\ U^*(2) = 0.8\left[-1 + U^*(1)\right] + 0.2\left[-2 + U^*(2)\right] \end{cases}$$

$$\begin{cases} 0.8U^*(1) - 0.8U^*(2) = -1.6 - 0.2 = -1.8 \\ -0.8U^*(1) + 0.8U^*(2) = -0.8 - 0.4 = -1.2 \end{cases}$$

$$\begin{cases} 0.8U^*(1) - 0.8U^*(2) = -1.8 \\ 0.8U^*(1) - 0.8U^*(2) = 1.2 \end{cases}$$

→ do not make sense can't find utility

Action b



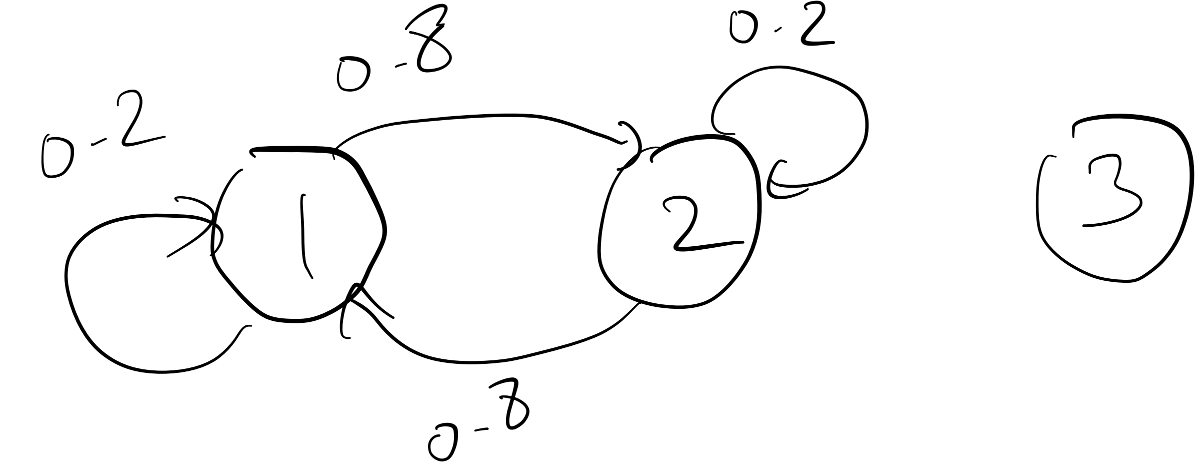
$$\begin{cases} U^*(1) = 0.1\left[0 + U^*(3)\right] + 0.9\left[-1 + U^*(1)\right] \\ U^*(2) = 0.1\left[0 + U^*(3)\right] + 0.9\left[-2 + U^*(2)\right] \\ U^*(3) = 0 \end{cases}$$

(use python or do it by hand if you want)

$$\begin{cases} U^*(1) = -9 \\ U^*(2) = -18 \\ U^*(3) = 0 \end{cases}$$

Let's implement some discounting with action a

$$\begin{cases} U^*(1) = 0.8\left[-2 + \gamma U^*(2)\right] + 0.2\left[-1 + \gamma U^*(1)\right] \\ U^*(2) = 0.8\left[-1 + \gamma U^*(1)\right] + 0.2\left[-2 + \gamma U^*(2)\right] \end{cases}$$



$$U^*(1) = \frac{\gamma + 3}{(\gamma - 1)(\gamma + \frac{5}{3})}$$

$\gamma = 1$ (bad value as well)

$$U^*(2) = \frac{2(\gamma + 1)}{(\gamma - 1)(\gamma + \frac{5}{3})}$$

if $\gamma = -\frac{5}{3}$ $\longrightarrow$ $U^*(1) = 2.1$

$U^*(2) = 2.4$

positive utilities but all rewards are negative $\Rightarrow$ doesn't make sense

bad value