Reinforcement Learning	March 20, 2019
Recitation 4	
Lecturer:Yishay Mansour	*TA: Lee Cohen*

Part 1: Finite Horizon MDP

Theorem 1. Finite-horizon Dynamic Programming. The following holds:

1. Backward recursion: Set $V_T(s) = r_T(s)$ for $s \in S_T$.
   For $k = T-1, ..., 0$, $V_k(s)$ may be computed using the following recursion:

$$V_k(s) = \max_{a \in A_k} \{r_k(s,a) + \sum_{s' \in S_{k+1}} p_k(s'|s,a) V_{k+1}(s')\}, s \in S_k \tag{1}$$

2. Optimal policy: Any Markov policy that satisfies, for $t = 0, ..., T-1$,

$$\pi_t^*(s) \in \arg\max_{a \in A_t} \{r_t(s,a) + \sum_{s' \in S_{t+1}} p_t(s'|s,a) V_{t+1}(s')\}, s \in S_t \tag{2}$$

   is an optimal control policy. Furthermore, maximizes $V_0(s_0)$ simultaneously for every initial state $s_0 \in S_0$.

Note that the theorem species an optimal control policy which is a deterministic Markov policy.

Example 1. Consider a Markov decision problem with two states, 0 and 1, and two decisions, 1 and 2, per state. This means that $S = \{0, 1\}$ and $A(0) = A(1) = \{1, 2\}$. The rewards are given by

$$r(s = 0, a) = 0, \quad r(s = 1, a) = 2 \quad a \in \{1, 2\} \tag{3}$$

and the transition probabilities by

$$P(1) = \begin{pmatrix} 1/2 & 1/2 \\ 2/3 & 1/3 \end{pmatrix}, \quad P(2) = \begin{pmatrix} 1/4 & 3/4 \\ 1/3 & 2/3 \end{pmatrix}, \tag{4}$$

The terminal rewards are given by

$$r_T = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \tag{5}$$

We wish to determine the maximum reward over a period with two decision epochs, i.e. $T = 2$ and the optimal policy for this MDP.
First,

$$V_2^*(0) = 2, \quad V_2^*(1) = 1 \tag{6}$$

Further,

$$\begin{aligned}
V_1^*(0) &= \max\{0 + \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1, 0 + \frac{1}{4} \cdot 2 + \frac{3}{4} \cdot 1\} = \frac{3}{2}, \quad \text{for } \pi_1^*(0) = 1 \\
V_1^*(1) &= \max\{2 + \frac{2}{3} \cdot 2 + \frac{1}{3} \cdot 1, 2 + \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 1\} = \frac{11}{3}, \quad \text{for } \pi_1^*(1) = 1 \\
V_0^*(0) &= \max\{0 + \frac{1}{2} \cdot \frac{3}{2} + \frac{1}{2} \cdot \frac{11}{3}, 0 + \frac{1}{4} \cdot \frac{3}{2} + \frac{3}{4} \cdot \frac{11}{3}\} = \frac{25}{8}, \quad \text{for } \pi_0^*(0) = 2 \\
V_0^*(1) &= \max\{2 + \frac{2}{3} \cdot \frac{3}{2} + \frac{1}{3} \cdot \frac{11}{3}, 2 + \frac{1}{3} \cdot \frac{3}{2} + \frac{2}{3} \cdot \frac{11}{3}\} = \frac{59}{6}, \quad \text{for } \pi_0^*(1) = 2
\end{aligned} \tag{7}$$

Part 2: MDP: discounted infinite horizon

Definitions

Definition 1. Expected Discounted Return defined for each control policy π and initial state $s_0 = s$ as follows:

$$J_\gamma^\pi(s) = E^\pi\left(\sum_{t=0}^\infty \gamma^t r(s_t, a_t) \mid s_0 = s\right) = E^{\pi,s}\left(\sum_{t=0}^\infty \gamma^t r(s_t, a_t)\right)$$

where,

1. $r : S \times A \to \mathbb{R}$ is the (running, or instantaneous) reward function.
2. $\gamma \in (0, 1)$ is the discount factor.

Definition 2. Value Function denote $V^\pi(s) = J_\gamma^\pi(s)$ as value function.

Theorem 2. Bellman's Optimality Equation The following statements hold:

1. V^* is the unique solution of the following set of (nonlinear) equations:
   $V(s) = \underset{a\in A}{max}\left\{r(s,a) + \gamma\sum_{s'\in S} p(s' \mid s, a)V(s')\right\}, s \in S$
2. Any stationary policy π^* that satisfies
   $\pi^*(s) \in \underset{a\in A}{argmax}\left\{r(s,a) + \gamma\sum_{s'\in S} p(s' \mid s, a)V(s')\right\}$
   is an optimal policy (for any initial state $s_0 \in S$).

Algorithm 1. Value iteration

1. Let $V_0 = (V_0(s))_{s\in S}$ be arbitrary.
2. For $n = 0, 1, 2, ...$, set $V_{n+1}(s) = \underset{a\in A}{max}\left\{r(s,a) + \gamma\sum_{s'\in S} p(s' \mid s, a)V_n(s')\right\}, \forall s \in S$

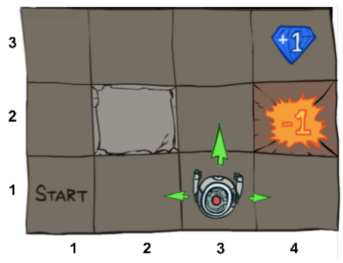
Algorithm 2. Policy iteration

1. Initialization: choose some stationary policy π_0
2. For $k = 0, 1, ...$
   (a) Policy evaluation: compute V^{π_k}
   (b) Policy Improvement: compute π_{k+1} a greedy policy with respect to V^{π_k}:
   $$\pi_{k+1}(s) \in \underset{a\in A}{argmax}\left\{r(s,a) + \gamma\sum_{s'\in S} p(s' \mid s, a)V^{\pi_k}(s')\right\}, \forall s \in S$$
   (c) Stop if $V^{\pi_k} = V^{\pi_{k+1}}$, else continue.

Exercises

Exercise 1. Value iteration Grid World is a 2D rectangular grid with an agent starting off at one grid square and trying to move to another grid square located elsewhere. In our implementation of Grid World we start the agent at the bottom-left grid corner with the aim of arriving at top-right grid corner at in a minimal number of steps. The size of our grid world is (3,4). The agent is only allowed actions of moving in up, left, right directions by 1 grid square. The agent isn't allow to step on (2,2), it get +1 reward for reaching the final state (3,4) and -1 reward for the state (2,4). With probability of 0.8 the agent action is preform exactly, the discount factor is 0.9. Find an optimal policy with value iteration algorithm.

Note regrading the noise in the actions - the noise can move the agent to the other direction **with respect to the chosen action**. For example, if the agent is at state (2,3), and choose to move left (2,2) then the noise can move the agent to states (3,3) or (1,3). In this case all of the equations that we wrote in the class are valid.



Solution 1. First we will state the equations of the value function algorithm:

$V_0^*(s) =$ optimal value for state s when $k = 0$
$V_0^*(s) = 0, \forall s$

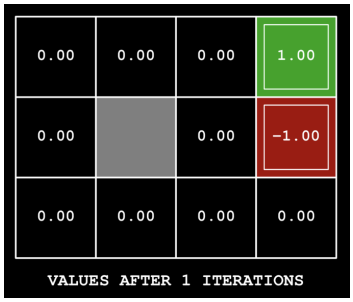
$V_1^*(s) =$ optimal value for state s when $k = 1$
$V_1^*(s) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s, a)(r(s, a, s') + \gamma V_0^*(s'))$

$V_2^*(s) =$ optimal value for state s when $k = 2$
$V_2^*(s) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s, a)(r(s, a, s') + \gamma V_1^*(s'))$

$V_k^*(s) =$ optimal value for state s, for iteration k
$V_k^*(s) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s, a)(r(s, a, s') + \gamma V_{k-1}^*(s'))$

for $k = 1$:
$V_1(3, 4) = 1$
$V_1(2, 4) = -1$

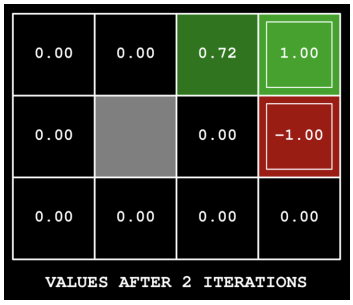
$V_1(s) = \underset{a \in \{u,l,r\}}{max} (0 + \sum_{s'} p(s' \mid s, a) V_0^*(s')) = 0, s \notin (2, 4), (3, 4)$



for $k = 2$:
$V_2(3,4) = 1$
$V_2(2,4) = -1$
$V_2(3,3) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma V_1^*(s')) = 0.8(0 + 0.9 * 1) = 0.72$
$V_2(s) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma V_1^*(s')) = 0.8(0 + 0.9 * 0) = 0, s \notin (2,4), (3,4), (3,3)$

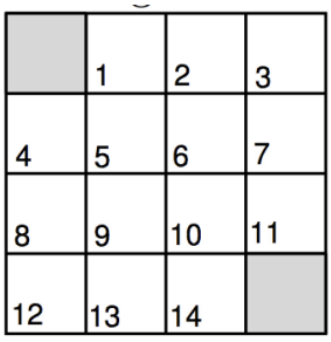
for $k = 3$:
$V_3(3,4) = 1$
$V_3(2,4) = -1$
$V_3(3,3) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma V_2^*(s')) = 0.8(0 + 0.9 * 1) + 0.1(0 + 0.9 * 0.72) + 0.1(0 + 0.9 * 0) = 0.78$
$V_3(2,3) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma V_2^*(s')) = 0.8(0 + 0.9 * 0.72) + 0.1(0 + 0.9 * (-1)) = 0.43$
$V_3(3,2) = \underset{a \in A}{max} \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma V_2^*(s')) = 0.8(0 + 0.9 * 0.72) = 0.52$
$V_2(s) = 0, s \notin (2,4), (3,4), (3,3), (2,3), (3,2)$

calculate till V(s) convergence.



Exercise 2. Policy iteration Grid world problem with:

1. action - right, left, up, down
2. Undiscounted episodic MDP $\gamma = 1$
3. Nonterminal states 1, ..., 14
4. One terminal state (shown twice as shaded squares)
5. Actions leading out of the grid leave state unchanged
6. Reward is -1 until the terminal state is reached
7. Agent follows uniform random policy



Solution 2. initialize π_0 as random policy, i.e. equal probability to any direction.

Policy Evaluation - with π_0 :

solve for $k = 0, 1, ...$

for $k = 0$:
set $V_0(s) = 0, \forall s \in S$

for $k = 1$:
move left/right/up/down with probability 0.25, get reward of -1 for any direction.
$V_0(s) = -1, \forall s \in S, s$ isn't terminal state.
$V_0(s) = 0$ for terminal states.

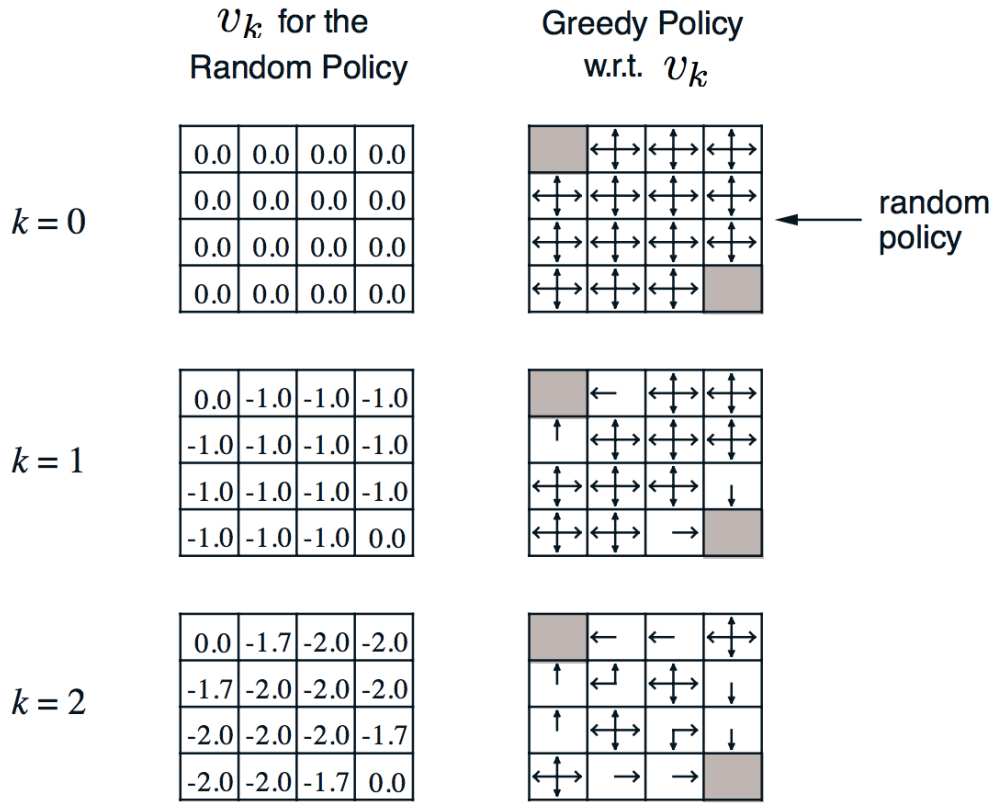
for $k = 2$:
$V_1(s) = -1.75, s = (0,1), (1,0), (3,2), (2,3)$ since $r = -1, p(s'|s,a) = 0.25, V_0(s) = -1$ and $3/4$ directions has -1 reward.
$V_1(s) = -2, s \notin (0,1), (1,0), (3,2), (2,3), (0,0), (3,3)$ since $r = -1, p(s'|s,a) = 0.25, V_0(s) = -1$ and $4/4$ directions has -1 reward.
$V_1(s) = 0$ for terminal states.

Do the same for k=3,... till value function converge.

Policy Improvement - Calculate π_1 :

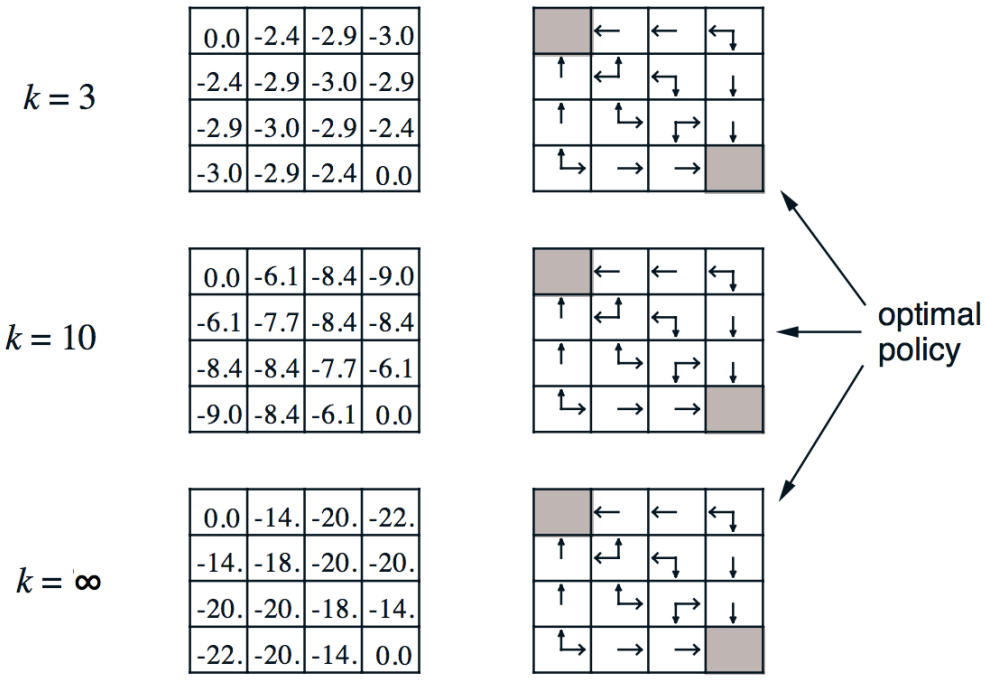
Improve the policy by acting greedy with respect to the value function from the step above.

In this problem of small grid world the improved policy was optimal, after 1 iteration of policy iteration. In general, we may need more iterations of improvement / evaluation.



References:

1. RL course, UCL, David Silver, Lecture 3
2. RL bootcamp, Berkeley, Lecture 1
3. "Learning and Planning in Dynamical Systems" by Shie Manor


$k = 3$
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
optimal policy