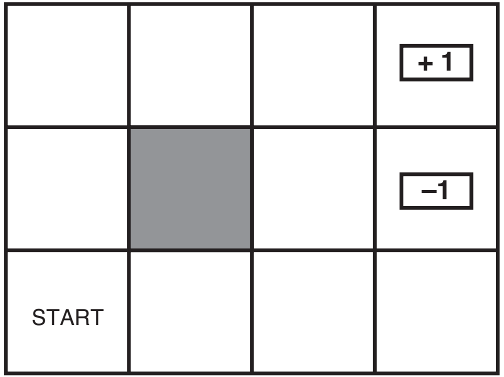
Sequential decisions under uncertainty
Markov Decision Processes (MDP)

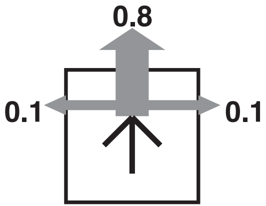
Tomáš Svoboda

Department of Cybernetics, Vision for Robotics and Autonomous Systems,
Center for Machine Perception (CMP)

March 21, 2019

Unreliable actions in observable grid world





States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$

Model $T(s, a, s') \equiv p(s'|s, a)$ = probability that a in s leads to s'

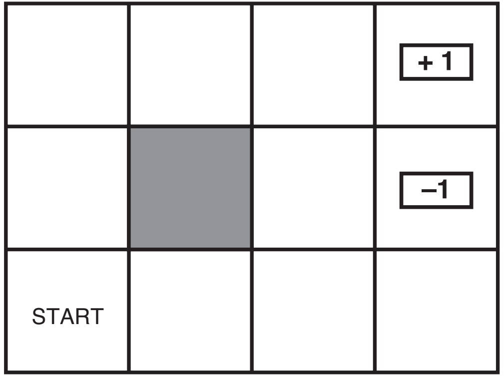
Observable - agent knows where it is. However, it does not always obey the command.

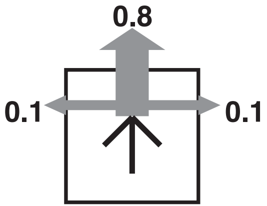
There is a treasure (desired goal/end state) but there is also some danger (unwanted goal/end state).

The danger state - think about a mountaineous are with safer but longer and shorter but more dangerous paths - a dangerous node may represent a chasm.

Notation note: caligraphic letters like $\mathcal{S}, \mathcal{A}$ will denote the set(s) of all states/actions.

Unreliable actions in observable grid world





States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$

Model $T(s, a, s') \equiv p(s'|s, a)$ = probability that a in s leads to s'

Observable - agent knows where it is. However, it does not always obey the command.

There is a treasure (desired goal/end state) but there is also some danger (unwanted goal/end state).

The danger state - think about a mountaineous are with safer but longer and shorter but more dangerous paths - a dangerous node may represent a chasm.

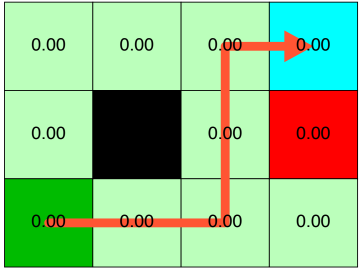
Notation note: caligraphic letters like $\mathcal{S}, \mathcal{A}$ will denote the set(s) of all states/actions.

Unreliable actions

Actions: go over a glacier bridge or around?

Plan? Policy

- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$.
- An action for each possible state.
- What is the best policy?

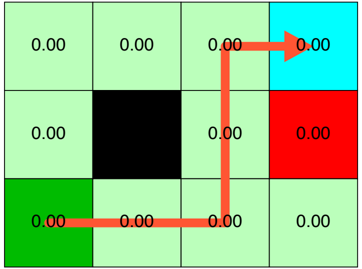


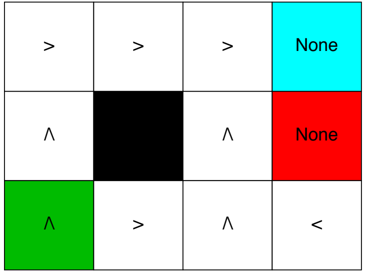
Remember, we can end up in any state. I any state, the robot/agent has to know what to do.

What is the best policy, we will come to that in a minute, ...

Plan? Policy

- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : \mathcal{S} \to \mathcal{A}$.
- An action for each possible state.
- What is the best policy?



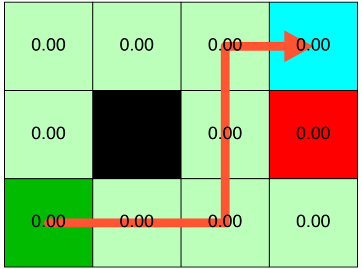


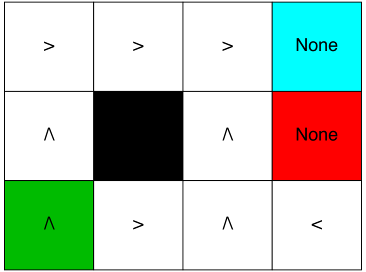
Remember, we can end up in any state. I any state, the robot/agent has to know what to do.

What is the best policy, we will come to that in a minute, ...

Plan? Policy

- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : \mathcal{S} \to \mathcal{A}$.
- An action for each possible state.
- What is the best policy?





Remember, we can end up in any state. I any state, the robot/agent has to know what to do.

What is the best policy, we will come to that in a minute, ...

Rewards

-0.04	-0.04	-0.04	1.00
-0.04		-0.04	-1.00
-0.04	-0.04	-0.04	-0.04

Reward : Robot/Agent takes an action a and it is **immediately** rewarded.

Reward function $r(s)$ (or $r(s,a)$, $r(s,a,s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

What do the rewards express? Reward to an agent to be/dwell in that state? Obviously we want the robot to go to the goal and do not stay too long in the maze.

Thinking about Reward: Robot/Agent takes an action a and it is immediately rewarded for this. The reward may depend on

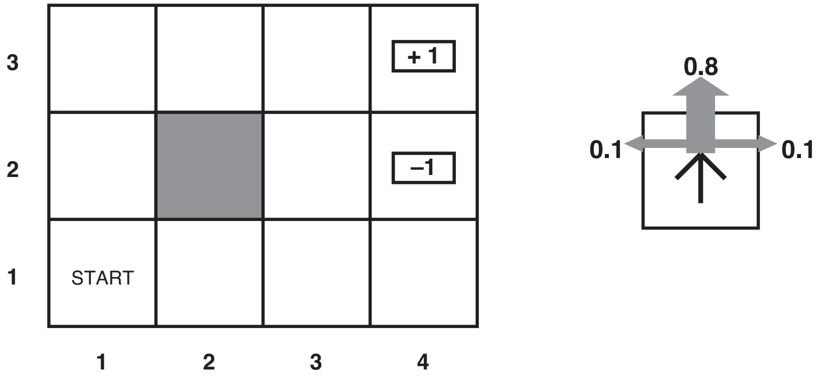
- current state s,
- the action taken a
- the next state s' - result of the action.

Rewards for terminal states can be understood in a way: there is only one action: $a = \text{exit}$. We will come to this soon.

The **reward function** is a property of (is related to) the problem.

Notation remark: lowercase letters will be used for functions like $p, r, v, f, \ldots$

Markov Decision Processes (MDPs)



(a) (b)

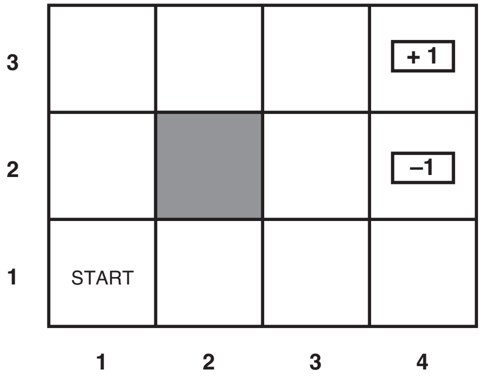
States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$

Model $T(s, a, s') \equiv p(s'|s, a)$ = probability that a in s leads to s'

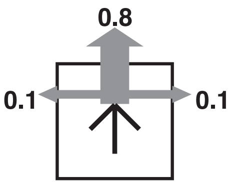
Reward function $r(s)$ (or $r(s, a)$, $r(s, a, s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Markov Decision Processes (MDPs)



(a)



(b)

States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$

Model $T(s, a, s') \equiv p(s'|s, a)$ = probability that a in s leads to s'

Reward function $r(s)$ (or $r(s, a)$, $r(s, a, s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Markovian property

- Given the present state, the future and the past are independent.
- MDP: Markov means action depends only on the current state.
- In search: successor function (transition model) depends on the current state only.

Optimal(?) policies

On-line demos.

- $r(s) = \{-0.04, 1, -1\}$
- $r(s) = \{-2, 1, -1\}$
- $r(s) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

We run `mdp_agents.py` changing reward functions.

Optimal(?) policies

On-line demos.

- $r(s) = \{-0.04, 1, -1\}$
- $r(s) = \{-2, 1, -1\}$
- $r(s) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

We run `mdp_agents.py` changing reward functions.

Optimal(?) policies

On-line demos.

- $r(s) = \{-0.04, 1, -1\}$
- $r(s) = \{-2, 1, -1\}$
- $r(s) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

We run `mdp_agents.py` changing reward functions.

Optimal(?) policies

On-line demos.

- $r(s) = \{-0.04, 1, -1\}$
- $r(s) = \{-2, 1, -1\}$
- $r(s) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

We run `mdp_agents.py` changing reward functions.

Utilities of sequences

- State reward reward value at time/step t, R_t.
- State at time t, S_t. State sequence $[S_0, S_1, S_2, \ldots,]$

Typically, consider stationary preferences on reward sequences:

$$[R, R_1, R_2, R_3, \ldots] \succ [R, R'_1, R'_2, R'_3, \ldots] \Leftrightarrow [R_1, R_2, R_3, \ldots] \succ [R'_1, R'_2, R'_3, \ldots]$$

If stationary preferences:
Utility (h-history)
$U_h([S_0, S_1, S_2, \ldots,]) = R_1 + R_2 + R_3 + \cdots$

We consdier discrete time t. S_t, R_t notation emphasises the time sequence - not a sequence of particular states. The reward is for an action (transition)

Utilities of sequences

- State reward reward value at time/step t, R_t.
- State at time t, S_t. State sequence $[S_0, S_1, S_2, \ldots,]$

Typically, consider stationary preferences on reward sequences:

$$[R, R_1, R_2, R_3, \ldots] \succ [R, R'_1, R'_2, R'_3, \ldots] \Leftrightarrow [R_1, R_2, R_3, \ldots] \succ [R'_1, R'_2, R'_3, \ldots]$$

If stationary preferences:
Utility (h-history)
$U_h([S_0, S_1, S_2, \ldots,]) = R_1 + R_2 + R_3 + \cdots$

We consdier discrete time t. S_t, R_t notation emphasises the time sequence - not a sequence of particular states. The reward is for an action (transition)

Utilities of sequences

- State reward reward value at time/step t, R_t.
- State at time t, S_t. State sequence $[S_0, S_1, S_2, \ldots,]$

Typically, consider stationary preferences on reward sequences:

$$[R, R_1, R_2, R_3, \ldots] \succ [R, R'_1, R'_2, R'_3, \ldots] \Leftrightarrow [R_1, R_2, R_3, \ldots] \succ [R'_1, R'_2, R'_3, \ldots]$$

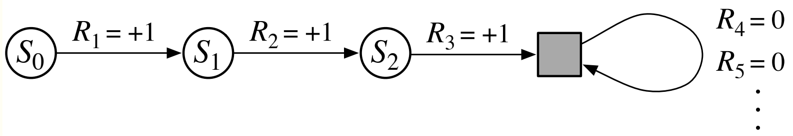
If stationary preferences:
Utility (h-history)
$U_h([S_0, S_1, S_2, \ldots,]) = R_1 + R_2 + R_3 + \cdots$

We consdier discrete time t. S_t, R_t notation emphasises the time sequence - not a sequence of particular states. The reward is for an action (transition)

Returns and Episodes

- Executing policy - sequence of states and **rewards**.
- Episode starts at t, ends at T (ending in a terminal state).
- Return (Utility) of the episode (policy execution)

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$



Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1$, $R_t \leq R_{\max}$

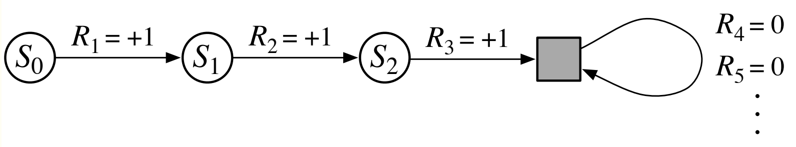
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime ⇒ additive utilities are infinite.

- Finite horizon: termination at a fixed time ⇒ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1$, $R_t \leq R_{\max}$

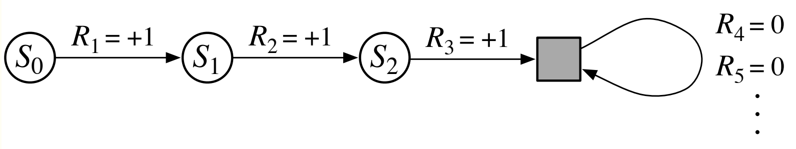
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime ⇒ additive utilities are infinite.

- Finite horizon: termination at a fixed time ⇒ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1, R_t \leq R_{\max}$

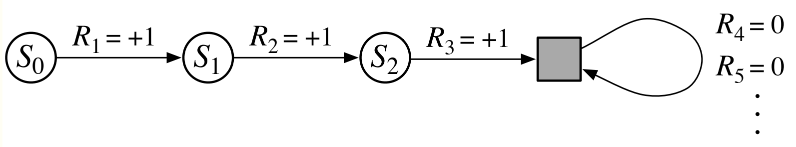
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1, R_t \leq R_{\max}$

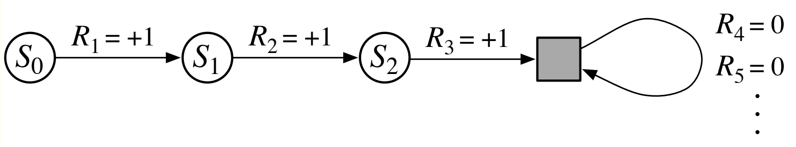
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1, R_t \leq R_{\max}$

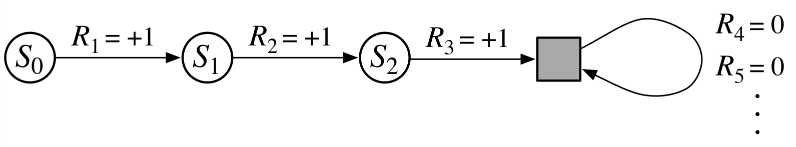
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime ⇒ additive utilities are infinite.

- Finite horizon: termination at a fixed time ⇒ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1$, $R_t \leq R_{\max}$

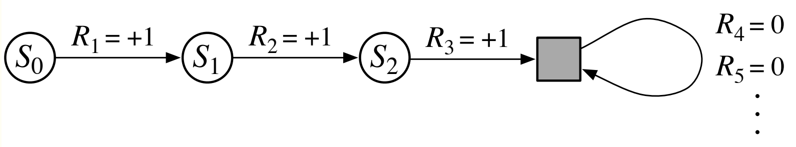
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime ⇒ additive utilities are infinite.

- Finite horizon: termination at a fixed time ⇒ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounted return , $\gamma < 1, R_t \leq R_{\max}$

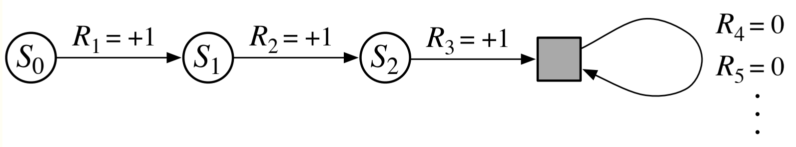
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing (terminal) state.

Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Discounting is quite natural choice. Think about your preferences/rewards. Go to pub with friends tonight, studying (for the far future reward of getting A in the course)?



Returns are successive steps related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

MDPs recap

Markov decition processes (MDPs):

- Set of states $\mathcal{S}$
- Set of actions $\mathcal{A}$
- Transitions $p(s'|s,a)$ or $T(s,a,s')$
- Reward function $r(s,a,s')$; and discount γ

MDP quantities:

- (deterministic) Policy $\pi(s)$ – choice of action for each state
- Return (Utility) of an episode (sequence) – sum of (discounted) rewards.

MDPs recap

Markov decition processes (MDPs):

- Set of states $\mathcal{S}$
- Set of actions $\mathcal{A}$
- Transitions $p(s'|s,a)$ or $T(s,a,s')$
- Reward function $r(s,a,s')$; and discount γ

MDP quantities:

- (deterministic) Policy $\pi(s)$ – choice of action for each state
- Return (Utility) of an episode (sequence) – sum of (discounted) rewards.

Value functions

- Executing policy π - sequence of states (and rewards).
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected return of a policy π.

Starting at time t, i.e. S_t,

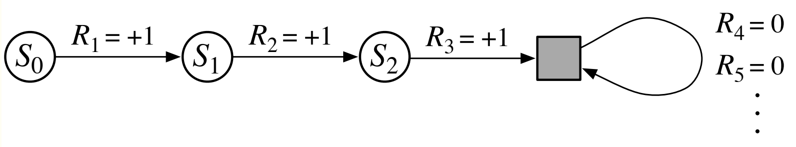
$$U^{\pi}(S_t) = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right]$$

Value function

$$v^{\pi}(s) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Action-value funtion (q-function)

$$q^{\pi}(s,a) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s, A_t = a\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$



Expected value can be also computed by running (executing) the policy many times and then computing average - Monte Carlo simulation methods.

Value functions

- Executing policy π - sequence of states (and rewards).
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected return of a policy π.

Starting at time t, i.e. S_t,

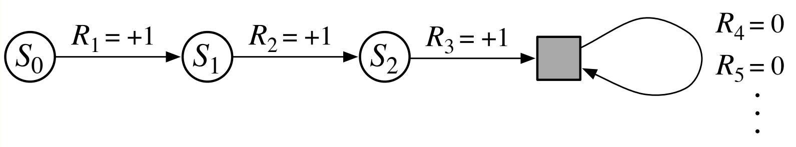
$$U^{\pi}(S_t) = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right]$$

Value function

$$v^{\pi}(s) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Action-value funtion (q-function)

$$q^{\pi}(s,a) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s, A_t = a\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$



Expected value can be also computed by running (executing) the policy many times and then computing average - Monte Carlo simulation methods.

Value functions

- Executing policy π - sequence of states (and rewards).
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected return of a policy π.

Starting at time t, i.e. S_t,

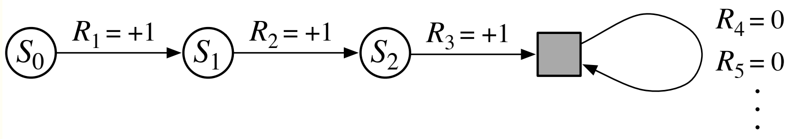
$$U^{\pi}(S_t) = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right]$$

Value function

$$v^{\pi}(s) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Action-value funtion (q-function)

$$q^{\pi}(s, a) = \mathrm{E}^{\pi}\left[G_t \mid S_t = s, A_t = a\right] = \mathrm{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$



Expected value can be also computed by running (executing) the policy many times and then computing average - Monte Carlo simulation methods.

Optimal policy π^*, and optimal value $v^*(s)$

$v^*(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions.

We still do not know how to compute the optimality, ... right?

Optimal policy π^*, and optimal value $v^*(s)$

$v^*(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions.

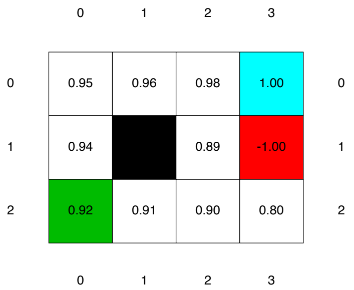
	0	1	2	3
0	0.81	0.87	0.92	1.00
1	0.76		0.66	-1.00
2	0.70	0.65	0.61	0.39

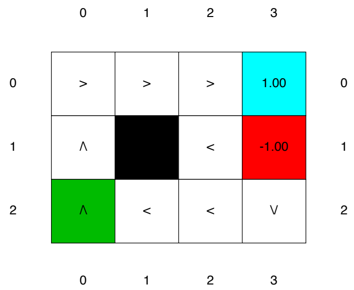
	0	1	2	3
0	>	>	>	1.00
1	∧		∧	-1.00
2	∧	<	<	<

We still do not know how to compute the optimality, ... right?

Optimal policy π^*, and optimal value $v^*(s)$

$v^*(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions.





We still do not know how to compute the optimality, ... right?

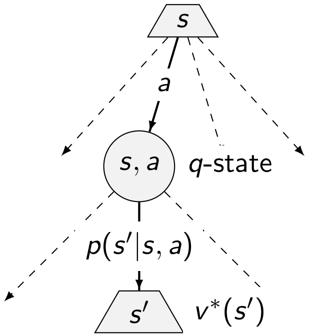
MDP search tree

The value of a q-state (s, a):

$$q^*(s,a) = \sum_{s'} p(s'|a,s) \left[r(s,a,s') + \gamma\, v^*(s')) \right]$$

The value of a state s:

$$v^*(s) = \max_a q^*(s,a)$$



$$\begin{aligned} v^\pi(s) &= \mathrm{E}^\pi \left[G_t \mid S_t = s \right] \\ &= \mathrm{E}^\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\ &= \sum_{s'} p(s'|a,s) \Big[r(s,a,s') + \gamma \mathrm{E}^\pi \left[G_{t+1} \mid S_{t+1} = s' \right] \Big] \end{aligned}$$

Remind Expectimax algorithm from the last lecture.
How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state s just anyhow.

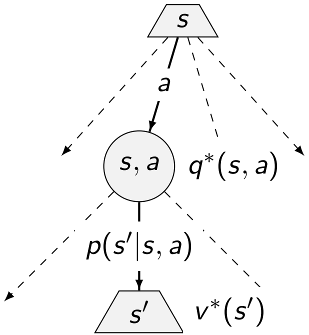
MDP search tree

The value of a q-state (s, a):

$$q^*(s,a) = \sum_{s'} p(s'|a,s)\left[r(s,a,s') + \gamma\, v^*(s')\right]$$

The value of a state s:

$$v^*(s) = \max_a q^*(s,a)$$



$$\begin{aligned} v^\pi(s) &= \mathrm{E}^\pi\left[G_t \mid S_t = s\right] \\ &= \mathrm{E}^\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_{s'} p(s'|a,s)\Big[r(s,a,s') + \gamma \mathrm{E}^\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\Big] \end{aligned}$$

Remind Expectimax algorithm from the last lecture.
How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state s just anyhow.

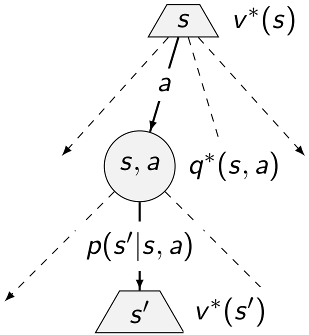
MDP search tree

The value of a q-state (s, a):

$$q^*(s,a) = \sum_{s'} p(s'|a,s)\left[r(s,a,s') + \gamma\, v^*(s')\right]$$

The value of a state s:

$$v^*(s) = \max_a q^*(s,a)$$

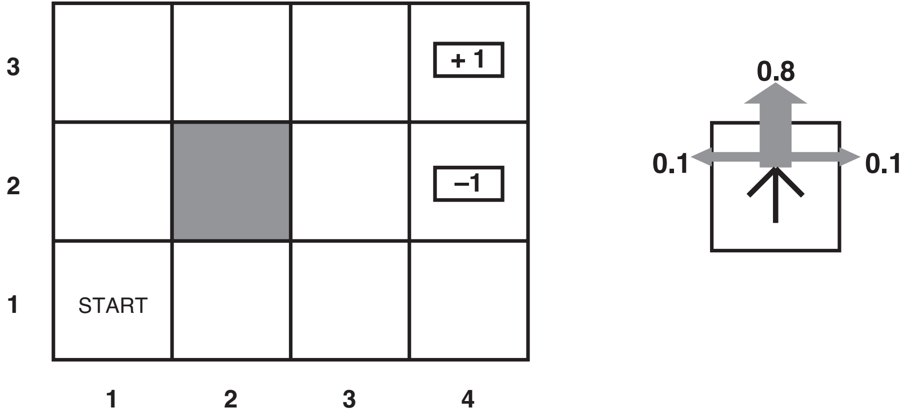


$$\begin{aligned} v^\pi(s) &= \mathrm{E}^\pi\left[G_t \mid S_t = s\right] \\ &= \mathrm{E}^\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_{s'} p(s'|a,s)\Big[r(s,a,s') + \gamma \mathrm{E}^\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\Big] \end{aligned}$$

Remind Expectimax algorithm from the last lecture.
How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state s just anyhow.

Bellman (optimality) equation

$$v^*(s) = \max_{a \in A(s)} \sum_{s'} p(s'|a, s) \left[r(s, a, s') + \gamma v^*(s') \right]$$



(a) (b)

v computation on the table - one row for each action. We got n equations for n unknown - n states. But max is a non-linear operator!

Value iteration

- Start with arbitrary $V_0(s)$ (except for terminals)
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.

What is the complexity of each iteration? $O(S^2 A)$

Value iteration

- Start with arbitrary $V_0(s)$ (except for terminals)
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.

What is the complexity of each iteration? $O(S^2A)$

Value iteration

- Start with arbitrary $V_0(s)$ (except for terminals)
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.

What is the complexity of each iteration? $O(S^2 A)$

Value iteration

- Start with arbitrary $V_0(s)$ (except for terminals)
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.

What is the complexity of each iteration? $O(S^2 A)$

Convergence

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

$$\gamma < 1$$

$$-R_{\max} \leq R(s) \leq R_{\max}$$

Max norm:

$$\|V\| = \max_s |V(s)|$$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

Convergence

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

$$\gamma < 1$$

$$-R_{\max} \leq R(s) \leq R_{\max}$$

Max norm:

$$\|V\| = \max_s |V(s)|$$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

Convergence cont'd

$V_{k+1} \leftarrow BV_k$
$\|BV_k - BV'_k\| \leq \gamma \|V_k - V'_k\|$
$\|BV_k - V_{\text{true}}\| \leq \gamma \|V_k - V_{\text{true}}\|$
Rewards are bounded, at the beginning then Value error is
$\|V_0 - V_{true}\| \leq \frac{2R_{\max}}{1-\gamma}$
We run N iterations and reduce the error by factor γ in each and want to stop the error is below ϵ:
$\gamma^N 2R_{\max}/(1-\gamma) \leq \epsilon$ Taking logs, we find: $N \geq \frac{\log(2R_{\max}/\epsilon(1-\gamma))}{\log(1/\gamma)}$
To stop the iteration we want to find a bound relating the error to the size of *one* Bellman update for any given iteration.
We stop if

$$\|V_{k+1} - V_k\| \leq \frac{\epsilon(1-\gamma)}{\gamma}$$

then also: $\|V_{k+1} - V_{\text{true}}\| \leq \epsilon$ Proof on the next slide

Try to proove that:

$$\|\max f(a) - \max g(a)\| \leq \max \|f(a) - g(a)\|$$

Convergence cont'd

$\|V_{k+1} - V_{\text{true}}\| \leq \epsilon$ is the same as $\|V_{k+1} - V_{\infty}\| \leq \epsilon$
Assume $\|V_{k+1} - V_k\| = \text{err}$
In each of the following iteration steps we reduce the error by the factor γ.
Till ∞, the total sum of reduced errors is:

$$\text{total} = \gamma\text{err} + \gamma^2\text{err} + \gamma^3\text{err} + \gamma^4\text{err} + \cdots = \frac{\gamma\text{err}}{(1-\gamma)}$$

We want to have total $< \epsilon$.

$$\frac{\gamma\text{err}}{(1-\gamma)} < \epsilon$$

From it follows that

$$\text{err} < \frac{\epsilon(1-\gamma)}{\gamma}$$

Hence we can stop if $\|V_{k+1} - V_k\| < \epsilon(1-\gamma)/\gamma$

Value iteration demo

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

	0	1	2	3	
0	0.81	0.87	0.92	1.00	0
1	0.76		0.66	-1.00	1
2	0.70	0.65	0.61	0.39	2
	0	1	2	3	

Run `mdp_agents.py` and try to compute next state value in advance. Remind the $R(s) = -0.04$ and $\gamma = 1$ in order to simplify computation. Then discuss the course of the Values.

Value iration algorithm

function VALUE-ITERATION(env,ϵ) **returns:** state values V
  input: env - MDP problem, ϵ
  $V' \leftarrow 0$ in all states
  repeat ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    for each state s **in** S **do**
      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      if $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    end for
  until $\delta < \epsilon(1-\gamma)/\gamma$
end function

Value iteration algorithm

function VALUE-ITERATION(env,ϵ) **returns:** state values V
  input: env - MDP problem, ϵ
  $V' \leftarrow 0$ in all states
  repeat ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    for each state s **in** S **do**
      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      if $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    end for
  until $\delta < \epsilon(1-\gamma)/\gamma$
end function

Value iteration algorithm

function VALUE-ITERATION(env,ϵ) **returns:** state values V
  input: env - MDP problem, ϵ
  $V' \leftarrow 0$ in all states
  repeat ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    for each state s **in** S **do**
      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      if $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    end for
  until $\delta < \epsilon(1-\gamma)/\gamma$
end function

Value iteration algorithm

function VALUE-ITERATION(env,ϵ) **returns:** state values V

  input: env - MDP problem, ϵ

  $V' \leftarrow 0$ in all states

  repeat ▷ iterate values until convergence

    $V \leftarrow V'$ ▷ keep the last known values

    $\delta \leftarrow 0$ ▷ reset the max difference

    for each state s **in** S **do**

      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V(s')$

      if $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$

    end for

  until $\delta < \epsilon(1-\gamma)/\gamma$

end function

Value iteration algorithm

function VALUE-ITERATION(env,ϵ) **returns:** state values V
  input: env - MDP problem, ϵ
  $V' \leftarrow 0$ in all states
  repeat ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    for each state s **in** S **do**
      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      if $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    end for
  until $\delta < \epsilon(1-\gamma)/\gamma$
end function

References

Some figures from [1] (chapter 17) but notation slightly changed adapting notation from [2] (chapters 3, 4) which will help us in the Reinforcement Learning part of the course.

[1] Stuart Russell and Peter Norvig.
Artificial Intelligence: A Modern Approach.
Prentice Hall, 3rd edition, 2010.
http://aima.cs.berkeley.edu/.

[2] Richard S. Sutton and Andrew G. Barto.
Reinforcement Learning; an Introduction.
MIT Press, 2nd edition, 2018.
http://www.incompleteideas.net/book/the-book-2nd.html.