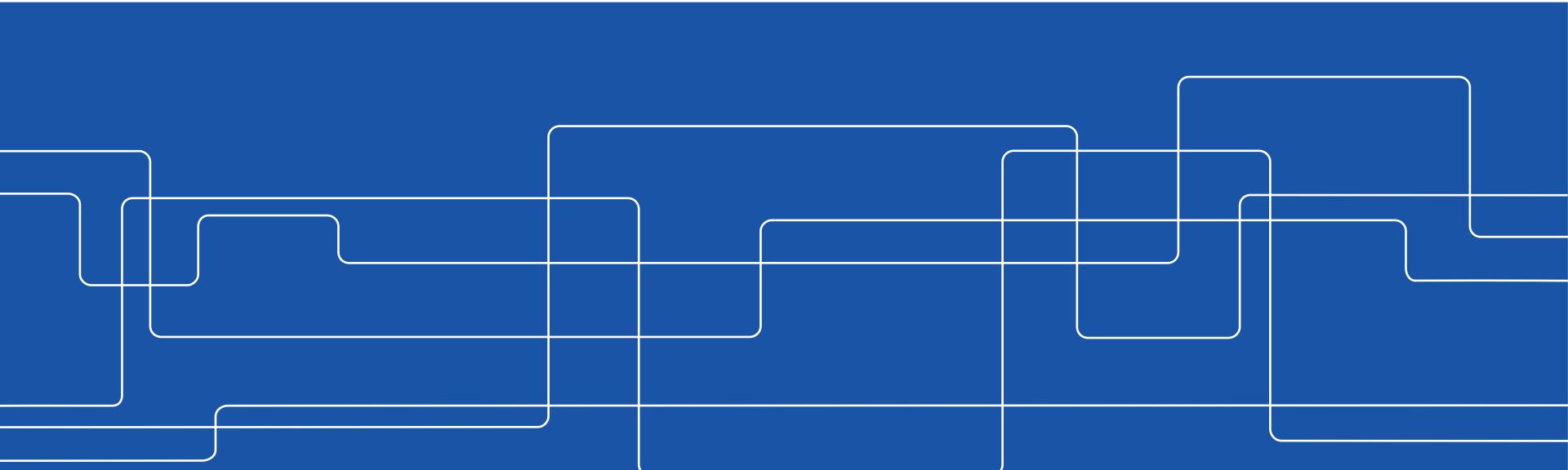




DD2380 Artificial Intelligence Making decisions under uncertainty

Florian Pokorny





Credits

Slides are minor modifications of previous DD2380 slides by:

- Patric Jensfelt, KTH
- Iolanda Leite, KTH

which were based partly on materials from:

- Berkeley AI
- Duke University AI (CPS 270)



Material & Reading

- Reading: Chapters **16** and **17** in the book



Motivation: Uncertainty Everywhere!

Not just for games of chance!

- I'm sick: will I sneeze this minute?
- Email contains “FREE!”: is it spam?
- Tooth hurts: have cavity?
- 60 min enough to get to the airport?
- Robot rotated wheel three times, how far did it advance?
- Safe to cross street? (Look both ways!)

Sources of uncertainty:

- Inherently random process (dice, etc)
- Insufficient or weak evidence
- Ignorance of underlying processes
- Unmodeled variables
- The world's just noisy – it doesn't behave according to plan!

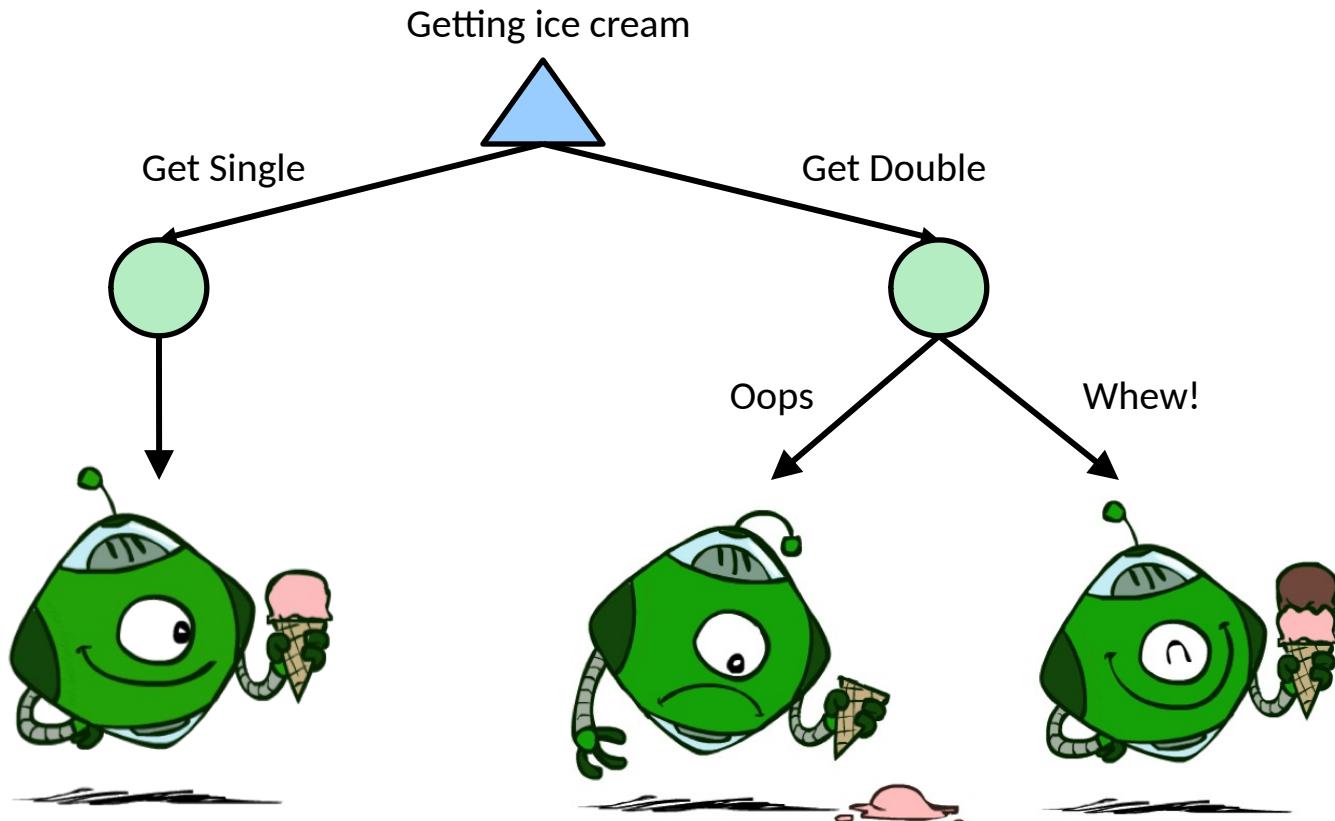
Foundation for many of the cool things happening now in reinforcement learning

Utility function

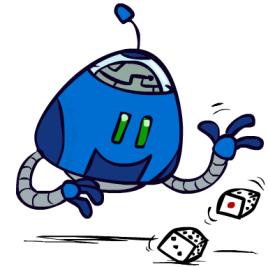
- Captures an agent's preference between world states
 - See also module **Problem solving**
- Assigns single number to express desirability of a state
- The utility of state S is denoted by $U(S)$



Uncertain Outcomes & Utility



Maximum Expected Utility (MEU)



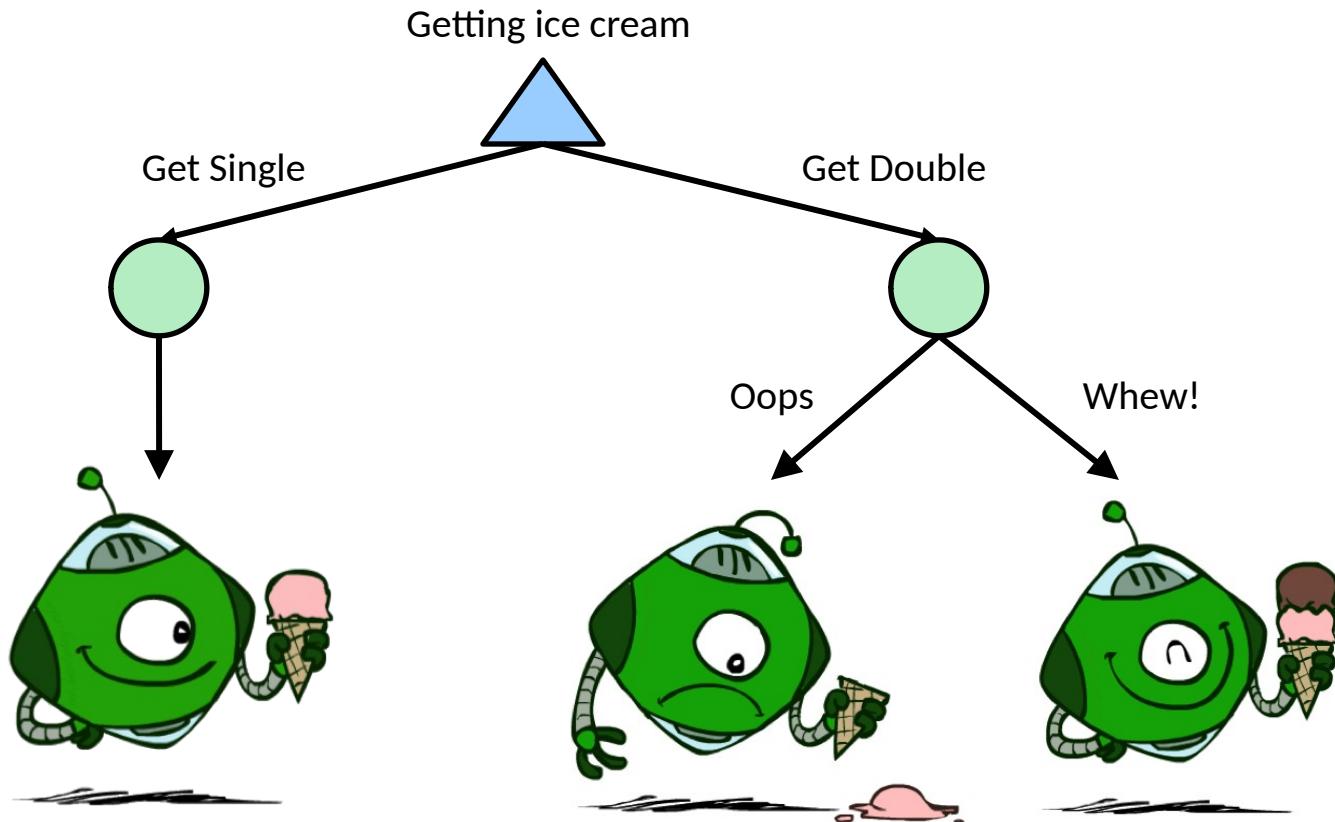
- Outcome of action is non-deterministic
- Result of single action A can be various states **Result_i(A)**, for varying i in I due to nondeterminism!
- Given evidence **E (measurements)** the probability for each result is

$$P(\text{Result}_i(A) \mid \text{Do}(A), E)$$

- Principle of maximum expected utility:
 - A rational agent picks an action that **maximizes the expected utility**, given its knowledge

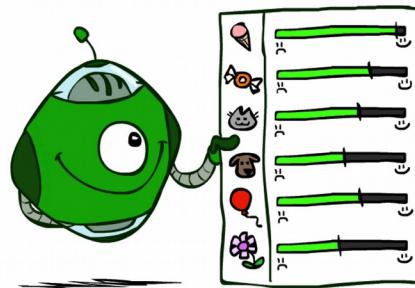
$$E[U(A|E)] = \underset{A}{\operatorname{argmax}} \sum_i P(\text{Result}_i(A)|\text{Do}(A), E) U(\text{Result}_i(A))$$

Utilities: Uncertain Outcomes



MEU not so easy to apply

- State of the world? (partially observable at best)
- How to compute $P(\text{Result}_i(A)|\text{Do}(A), E)$? Requires a model of the world
- Also need to consider one action (one shot decision) vs. sequential decisions
- Utility of a state?





The value of information

- Asking for information is one of the most important actions
 - Information is acquired through “sensing actions”
 - Information typically has a cost
 - Agent must ask itself, what information to ask for?
-
- Information has **value** if it might change your action
 - Information value = difference between **expected value with and without** the information

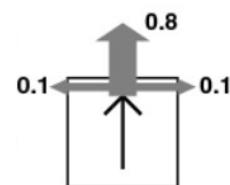
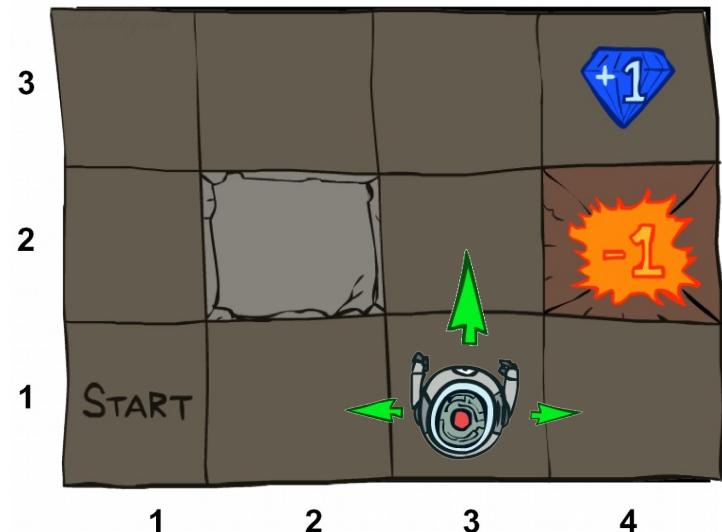


Sequential decisions

- A one step horizon (one action) often not good enough
- Sequential environments!

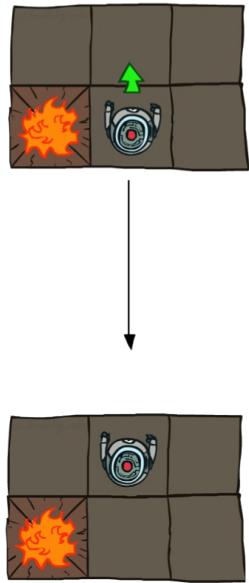
Seq. decisions example: Grid World

- A maze-like problem
 - The agent lives in a grid
 - Walls block the agent's path
- Noisy movement: actions (Up, Down, Left, Right) do not always go as planned
 - 80% of the time, the intended outcome occurs (if there is no wall there)
 - 20% of the time, the agent moves at right angles to the intended direction
 - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards each time step
 - Small “living” reward each step (can be negative)
 - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards



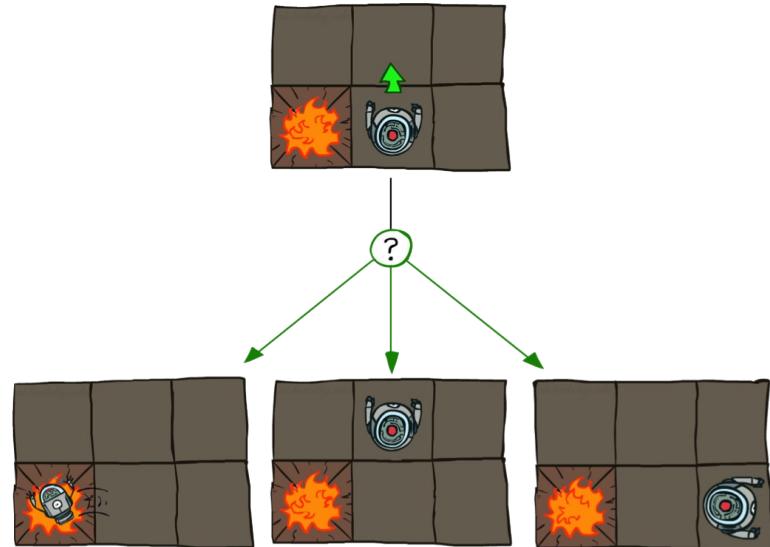
Grid World Actions

Deterministic Grid
World



Plan: Up, Up, Right, Right, Right

Stochastic Grid World

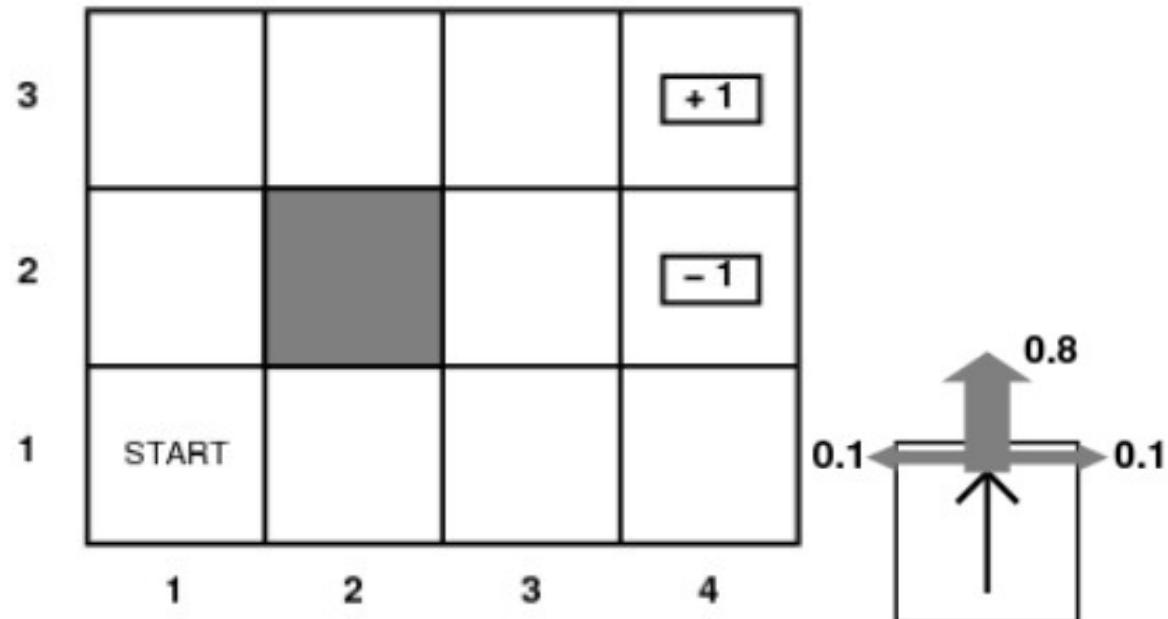


Cannot create a plan ahead of time!

What is the probability that a predefined plan succeeds?

Plan: Up, Up, Right, Right, Right

Probability to succeed: ?

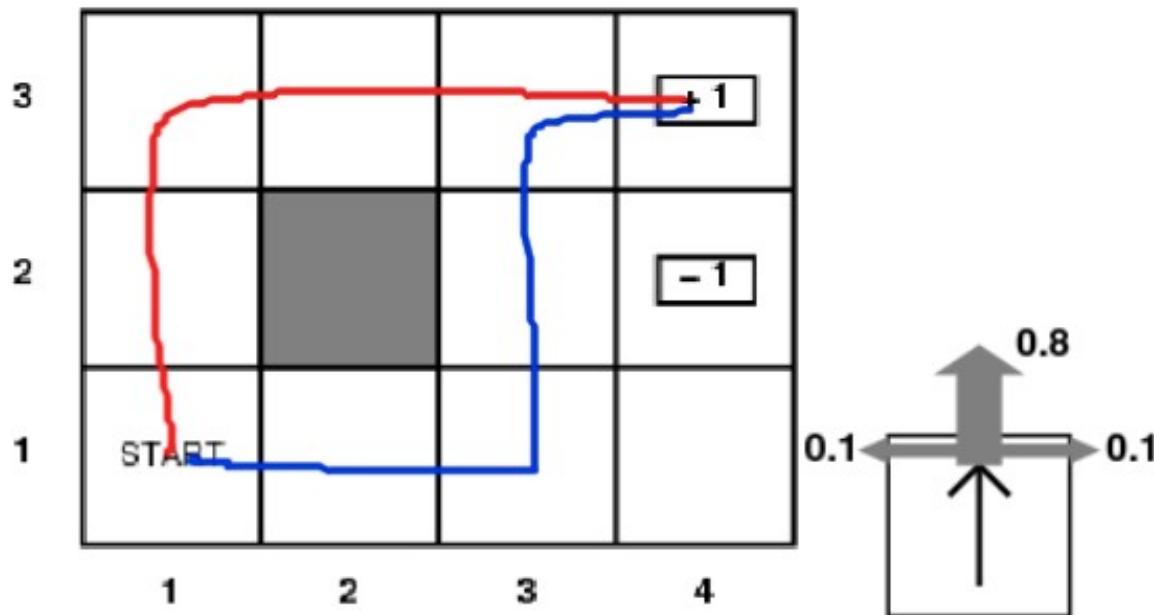


What is the probability that a predefined plan succeeds?

Plan: Up, Up, Right, Right, Right

Probability to succeed: 0.32776

$$0.8^5 + (0.1^4 \cdot 0.8) = 0.32776$$





Connection to past lectures?

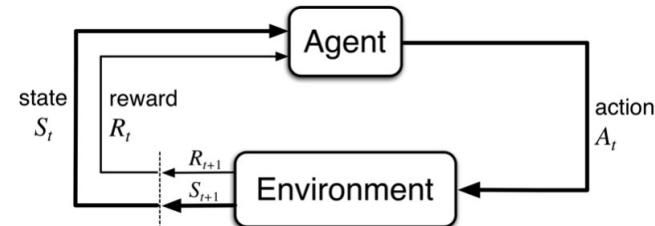
- Remember Markov Models
 - Each square is a state in the previous example
- The transition function defines the probability transitioning from one state to another.
- In decision making, each **action is associated with its own transition function**
 - select action to control the system to behave in some desired way or maximize the chance of achieving some goal

“Markovian” here generally means that the probability of reaching the next state s' depends only on current state s and action a (and **not** the full history of past states and actions!)

Markov Decision Process (MDP) Formulation

Mathematical model: Markov Decision Process (S, A, T, R, γ)

- State space S
- Action space A
- Environment Transition model T
- Reward function R
- Discount factor γ
- A start state S_0
- Maybe a terminal state



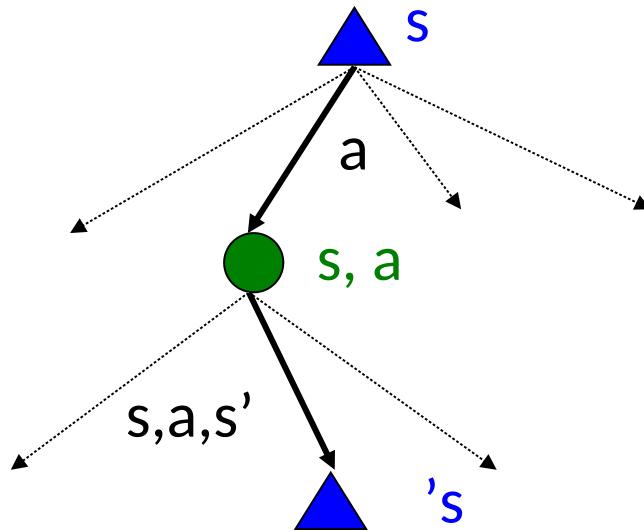
$$T(s, a, s') : p(s_{t+1} = s' | s_t = s, a_t = a)$$

$R(s, a, s')$: immediate reward for transitioning from state s to state s' due to action a

Frequently reward depends only on the state, so we usually write $R(s)$

Transition model

- Now function of the action
- $T(s,a,s')$ (first order Markov assumption*)
Probability to reach s' starting from s given action a .

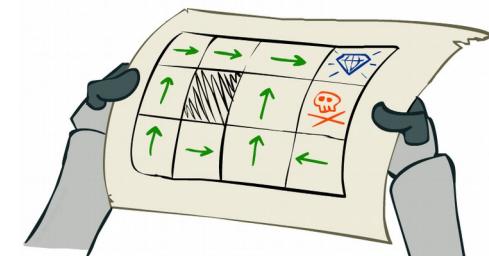


* T depends only on the previous state s and not the rest of the history

Solution to MDP

In deterministic single-agent search problems, we wanted an optimal **plan**, or sequence of actions, from start to a goal

A solution to an MDP cannot be a fixed plan (non-deterministic world, need to sense state)



For MDPs, we want an optimal **policy** $\pi^*: S \rightarrow A$

- A policy π gives an action for any state
 - Need to **define utility function for any sequence of states visited**
 - An optimal policy is then one that maximizes expected utility if followed
- Sidenote: it is better to say “an optimal” rather than “the optimal” as there may be multiple optimal policies!*



Utility of sequences

Additive rewards:

$$U_h([s_0, s_1, \dots, s_n]) = R(s_0) + R(s_1) + \dots + R(s_n)$$

What about infinite sequences?

- Infinite sum might not converge!

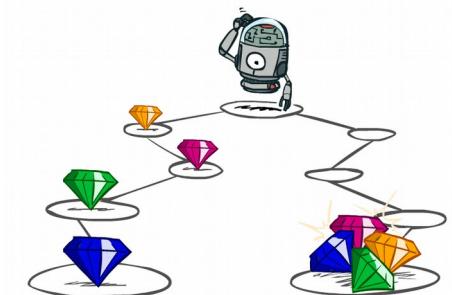
Utility of sequences

$$U_h([s_0, s_1, \dots, s_n]) = R(s_0) + R(s_1) + \dots + R(s_n)$$

What preferences should an agent have over reward sequences?

More or less? [1, 2, 2] or [2, 3, 4]

Now or later? [0, 0, 1] or [1, 0, 0]



Discounted rewards

- Idea: Give less weight to future rewards (e.g. rewards decay exponentially)
- Captures that rewards “tomorrow” is less certain
- Use discount factor γ , $0 < \gamma < 1$



1

Worth Now



γ

Worth Next Step

- If rewards bounded, it the sum converges and bounded by:

$$R(s) \leq R_{max} \Rightarrow U_h \leq \sum_{i=0}^{\infty} \gamma^i R_{max} = R_{max} \frac{1}{1-\gamma}$$



γ^2

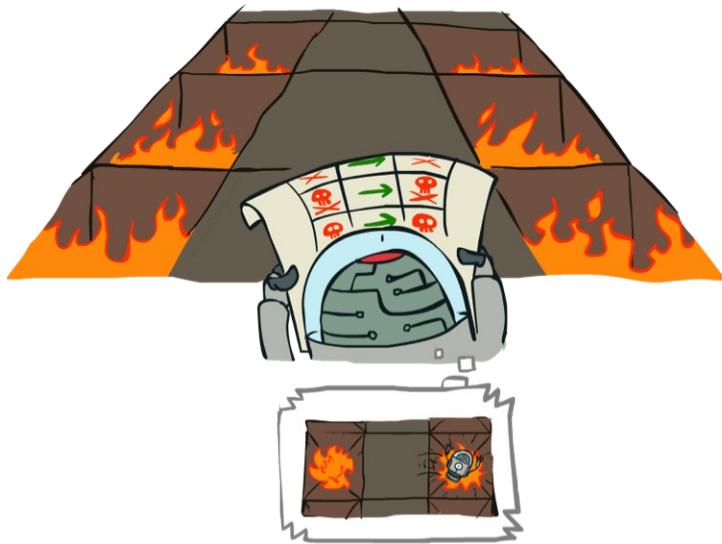
Worth In Two Steps

Key quantities we will now utilize to understand policies:

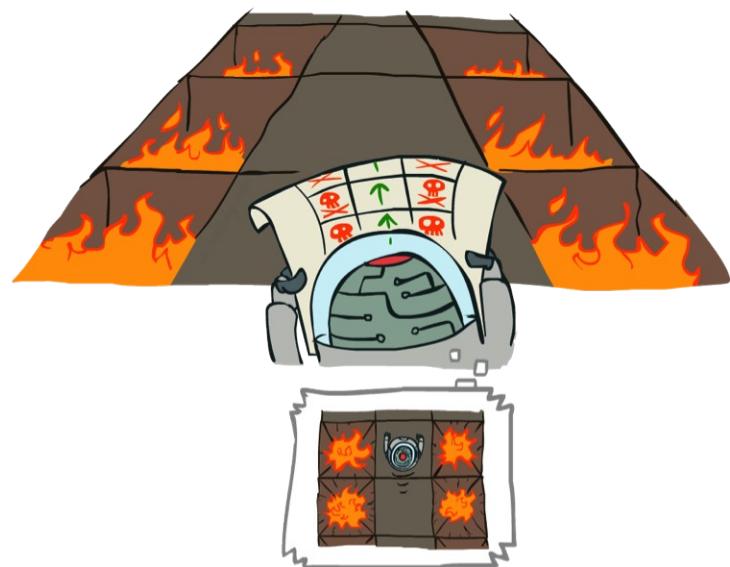
- The **value function** for a policy $V^\pi : S \rightarrow \mathbb{R}$ maps each state s to the expected utility of executing that policy starting at s (*expectation using utility of sequences of states!*). Sometimes we may just write V when the policy is specified.
- An **optimal policy** is a policy that maximizes the value function at each state.
- The **utility function** $U : S \rightarrow \mathbb{R}$ at a state s is defined to be the value function of that state under an optimal policy.

Policy Evaluation

Always Go Right



Always Go Forward



Intuition: Value Functions for Two Policies

Always Go Right



Always Go Forward



Selecting the best policy

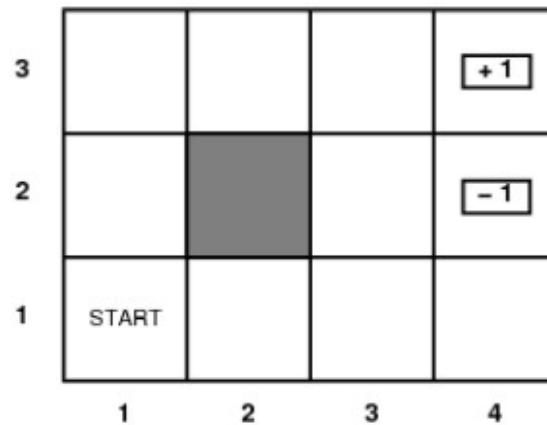
- How do we select the best policy?
 - i.e. choice of action for each state
- Many state sequences to compare...
- As before, maximize expected utility to obtain an optimal policy

$$\pi^* = \arg \max_{\pi} E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi \right]$$


sum of discounted rewards

Ex: Optimal policy π^*

- An example from the book
- Assume agent gets reward $R(s)$ for being in state s
 - $R([4,3]) = +1$ (Go to goal)
 - $R([4,2]) = -1$ (Avoid trap)
 - $R(\text{other states}) = -0.04$ (Get to goal quickly)

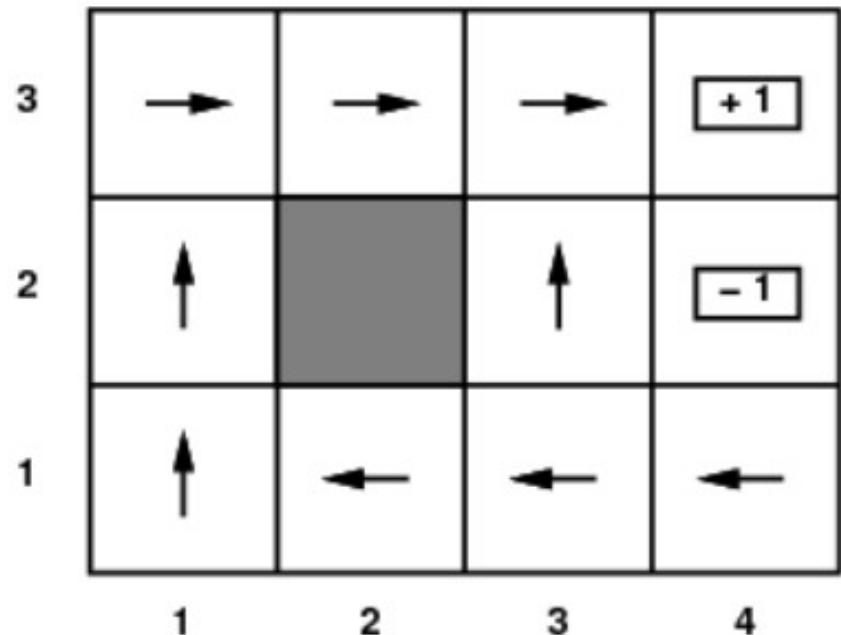


Example: Optimal policy π^*

Optimal policy for the previous problem

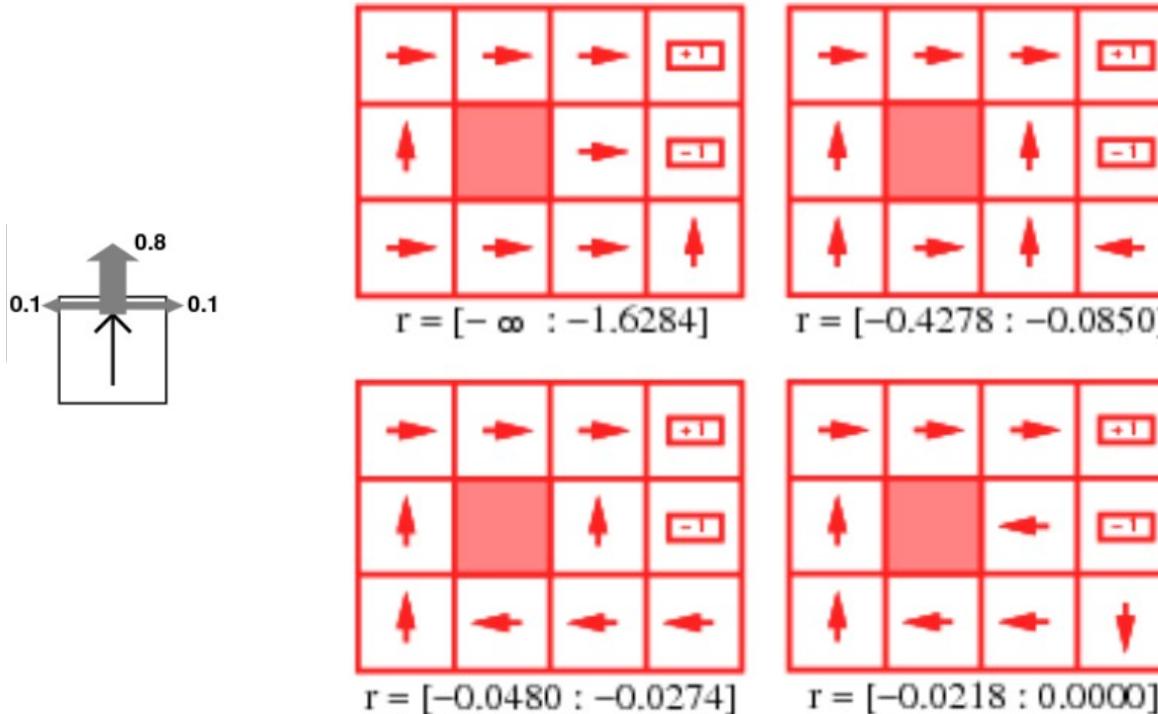
Tells us what to do in each state

Actual path only known when moving because actions are non-deterministic



Optimal policy under depends on R, T!

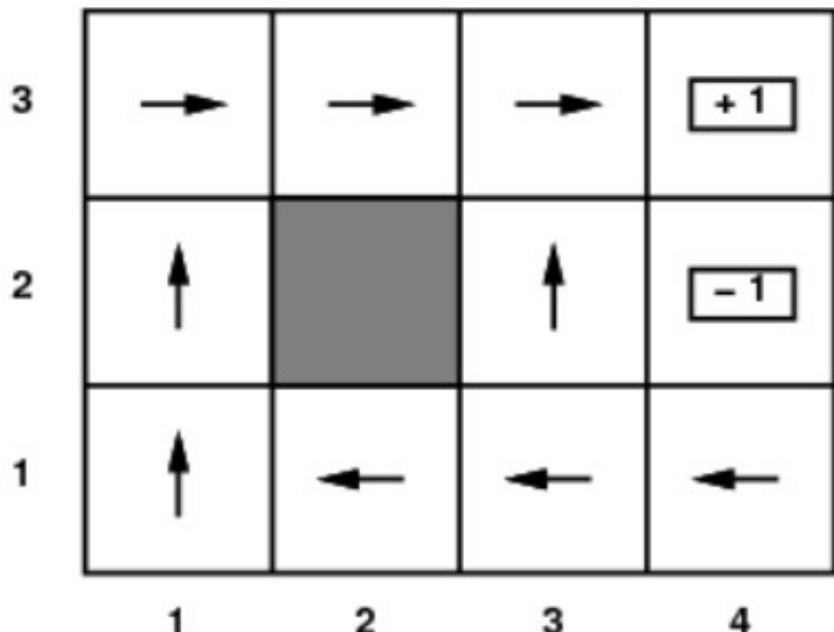
Different behaviors for different value of R(remaining states)=r



How about changing T? (see **stationary** vs nonstationary policies)
 What if R is very large? What if we change discount factor?

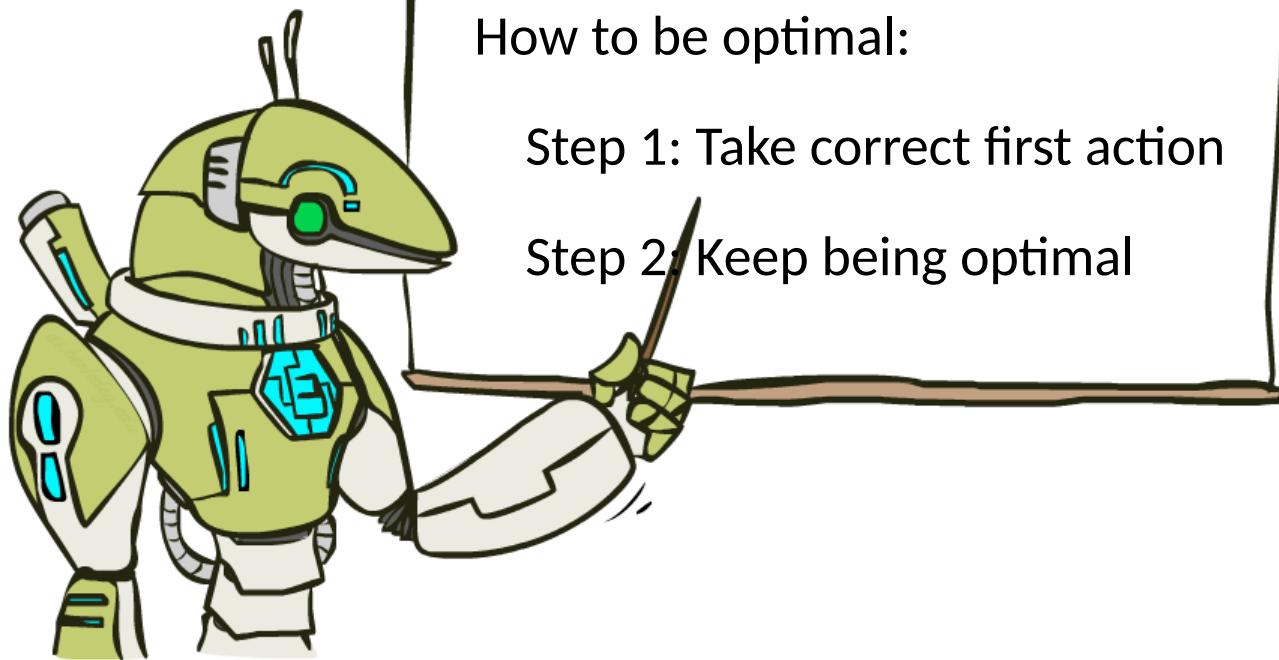
Example: Utilities of States

The utilities of the states in the 4X3 world, calculated with $\gamma = 1$ and $R(s) = -0.04$ for nonterminal states.



3	0.812	0.868	0.918	+1
2	0.762		0.660	-1
1	0.705	0.655	0.611	0.388

Computation using Bellman Equation





How to Compute all this ?

Answer: Bellman Equation

$$\begin{aligned}V^\pi(s) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i) | \pi, s_0\right] = \mathbb{E}[R(s_0) + \gamma \sum_{i=0}^{\infty} \gamma^i R(s_{i+1}) | \pi, s_0] \\&= R(s_0) + \gamma \sum_{s'} T(s, \pi(s), s') V^\pi(s')\end{aligned}$$

How to determine the value function for a policy?



How to Compute all this ?

Answer: Bellman Equation

$$\begin{aligned}V^\pi(s) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i) | \pi, s_0\right] = \mathbb{E}[R(s_0) + \gamma \sum_{i=0}^{\infty} \gamma^i R(s_{i+1}) | \pi, s_0] \\&= R(s_0) + \gamma \sum_{s'} T(s, \pi(s), s') V^\pi(s')\end{aligned}$$

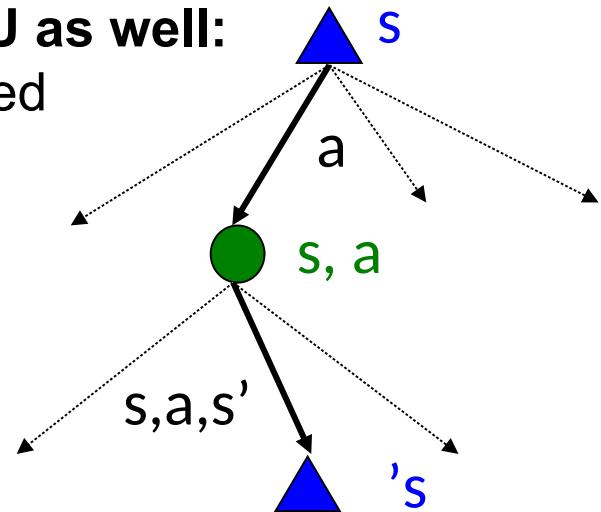
These are just linear equations in the unknown value function values!

Bellman Equation

Key insight we can exploit recursive nature of U as well:

Utility of a state is immediate reward plus discounted
expected utility of next states

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$



Strategy to solve MDP:

- Calculate U of each state based on above decomposition
- Use utility values to simply select that action with highest future expected reward at any state to find optimal policy.

Value Iteration Idea

Initialize U_0 (to zero or even randomly) then iterate:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s')$$

immediate reward

discounted expected utility of the next state, assuming optimal action

Fact: Converges to optimal solution

Once s, a determined still potentially many possible s' due to stochasticity

Stop iterations when largest change in utility for any state is small enough

Can show that

$$\|U_{i+1} - U_i\| < \epsilon \frac{1 - \gamma}{\gamma} \Rightarrow \|U_{i+1} - U\| < \epsilon$$

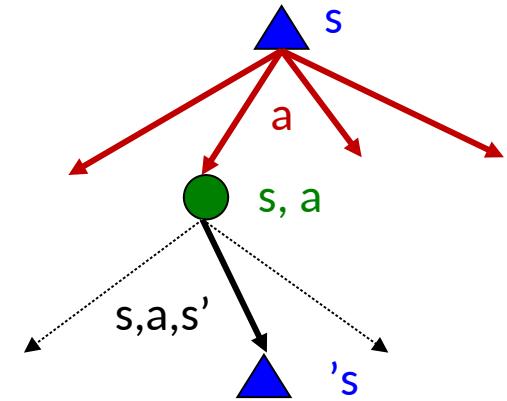
Algorithm: Value iteration

Initialize $U(s)$ arbitrarily for all s

Loop until policy has converged

 loop over all states, s

 loop over all actions, a



$$Q(s, a) := R(s) + \gamma \sum_{s'} T(s, a, s') * U(s')$$

end

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

end

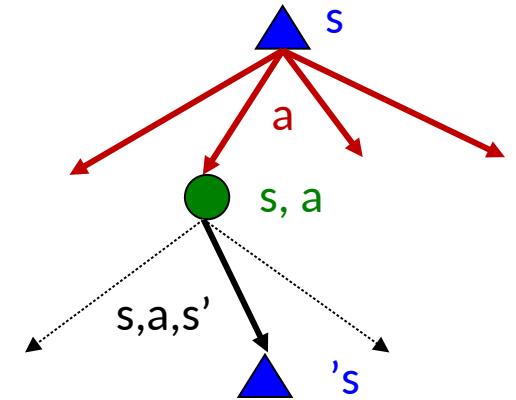
end

NOTE: Q-function comes back in Reinforcement learning

Problems with Value Iteration

Value iteration repeats the Bellman updates:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$



Problem 1: May be slow – $O(S^2A)$ per iteration

Problem 2: The “max” at each state rarely changes

Problem 3: The policy often converges long before the values

Algorithm: Policy iteration

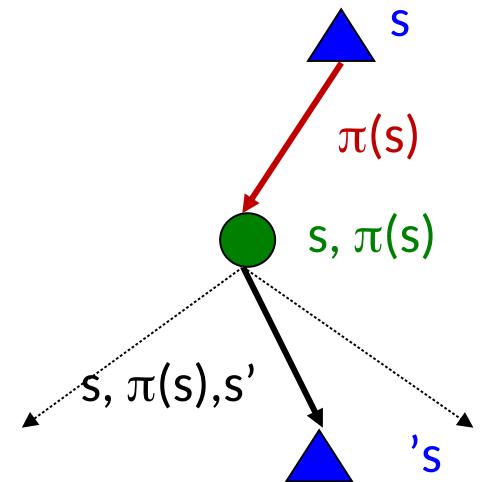
Choose an arbitrary policy

Loop until policy does not change any more

1. **Policy evaluation:** Compute the value function $V(s)$ for current policy estimate π exactly or using iterative approximation

2. **Policy improvement:** Given this value function, improve the policy for each state

$$\pi(s) \leftarrow \arg \max_a \sum_{s' \in \mathcal{S}} T(s, a, s') [R(s, a, s') + \gamma V(s')]$$





Value Iteration vs. Policy Iteration

Both value iteration and policy iteration compute the same thing (optimal policy)
Both are dynamic programs for solving MDPs

In **value iteration**:

- Every iteration updates both the values and (implicitly) the policy
- We don't track the policy, but taking the max over actions implicitly recomputes it

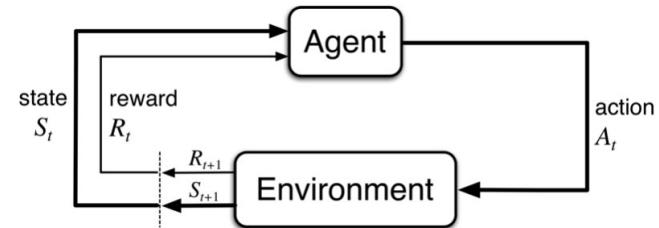
In **policy iteration**:

- We do several passes that update utilities with fixed policy (each pass is fast because we consider only one action, not all of them)
- After the policy is evaluated, a new policy is chosen (slow like a value iteration pass)
- The new policy will be better (or we're done)
- Can converge (much) faster under some conditions (large nr of actions but where max action doesn't change much)

MDP Recap

Mathematical model: Markov Decision Process (S, A, T, R, γ)

- State space S
- Action space A
- Environment Transition model T
- Reward function R
- Discount factor γ
- A start state S_0
- Maybe a terminal state



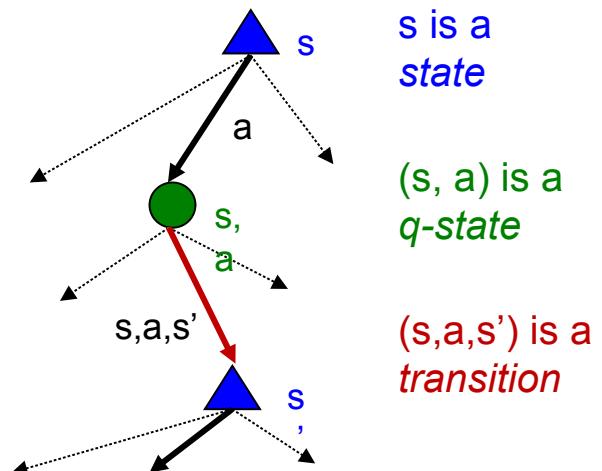
$$T(s, a, s') : p(s_{t+1} = s' | s_t = s, a_t = a)$$

$R(s, a, s')$: immediate reward for transitioning from state s to state s' due to action a

Frequently reward depends only on the state, so we usually write $R(s)$

Optimal Quantities

- The utility/value of a state s :
 $U(s) = V^*(s)$ = expected utility starting in s and acting optimally
- The value of a q-state (s,a) :
 $Q^*(s,a)$ = expected utility starting out having taken action a from state s and (thereafter) acting optimally
- The optimal policy:
 $\pi^*(s)$ = optimal action from state s



Partially observable environment

What if the environment is not observable?

Remember: there are no sensors that let us see everything, so this is the normal case!





Partially observable environment

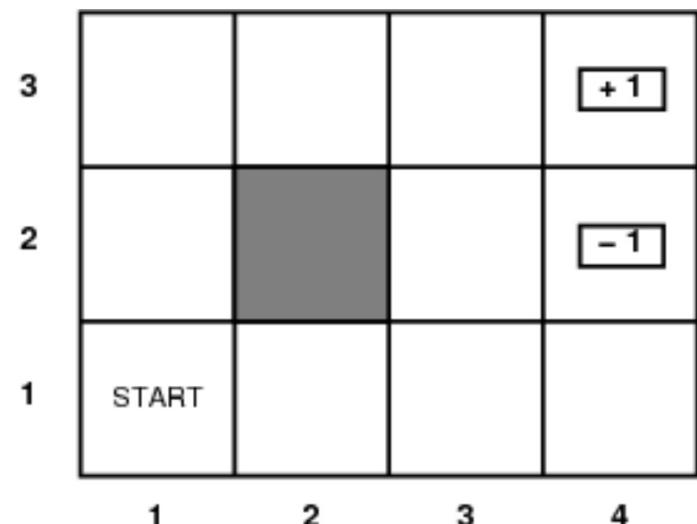
- What if the environment is not observable?
- Cannot execute policy since **the state is unknown!**
- Results in a Partially Observable MDP aka **POMDP**
- Sensors provide observations of the environment
- Observation model $O(s,o)$ gives probability of making observation o in state s

Belief state

- Idea: Use the belief state instead of the actual state
- Definition: Belief state b is a probability distribution over possible states
- $b(s)$ is the probability of being in state s

- Example: Initial belief state assuming that the agent can be anywhere except +1 or -1?

$$\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$$





Belief state update

- Need to update the belief state as we go along
- Assume **belief state b** and **action a**
- Update?
 - $b'(s') = \text{some function of } b(s), a, s \text{ and o....}$

Belief state update

- Assume belief state b , action a and new observation o
- Update:

$$b'(s') = \alpha O(s', o) \sum_s T(s, a, s') b(s)$$

α is a normalization factor such that $\sum_{s'} b'(s') = 1$

Bayes rule with:

- $b(s) = p(s)$ ("prior")
- $b'(s') = p(s'|s, a, o)$ ("posterior")
- $\text{Sum}[T(s, a, s')b(s)] = p(s'|a, s)$ ("prediction")
- $O(s', o) = p(o|s') = p(o|s', s, a)$ { o indep of s and a , given s' }

Solving POMDP

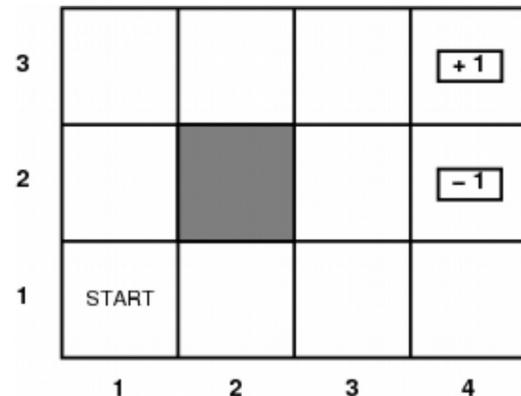
Key insight:

Optimal action depends on belief state and not actual state!

Optimal policy $\pi^*(b)$

Decision cycle:

1. Execute action $a = \pi^*(b)$
2. Receive observation
3. Update belief state





Turn a POMDP into an MDP

Introduce

- $\tau(b,a,b')$ – probability of reaching belief state b' from b given action a
- $\rho(b) = \sum b(s)R(s)$

$\tau(b,a,b')$ and $\rho(b)$ define an observable MDP

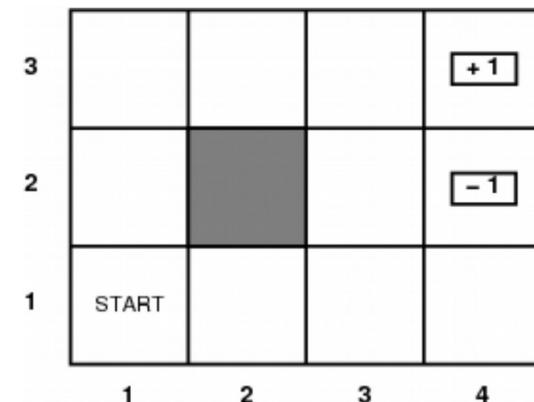
Optimal MDP policy $\pi^*(b)$ is also optimal for the original POMDP

More on Chapter 17.4 of the book

So is it simple?

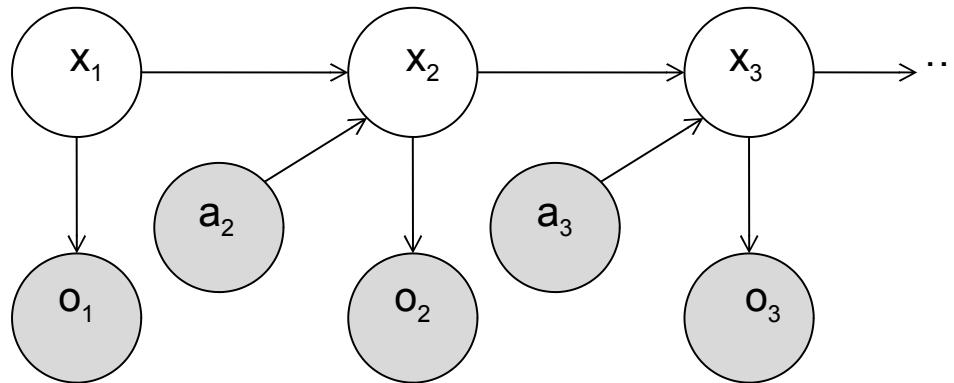
- Sounds “simple” at first, BUT...
- Belief state is a probability distribution!
- Compare MDP and POMDP in the 4x3 world
 - MDP state : The position of the agent, i.e. 1 discrete variable with 11 possible values.
 - POMDP belief state: 11 dimensional vector of continuous variables!!!

$$\left\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \right\rangle$$



POMDP Connection to HMM?

- The next state (position) depends only on the previous state **but also the action**
- Our transition matrix is **a function of action!**
- So POMDPs have partial observation **like in HMM case** but unlike MDP **have action dependence**



The big picture

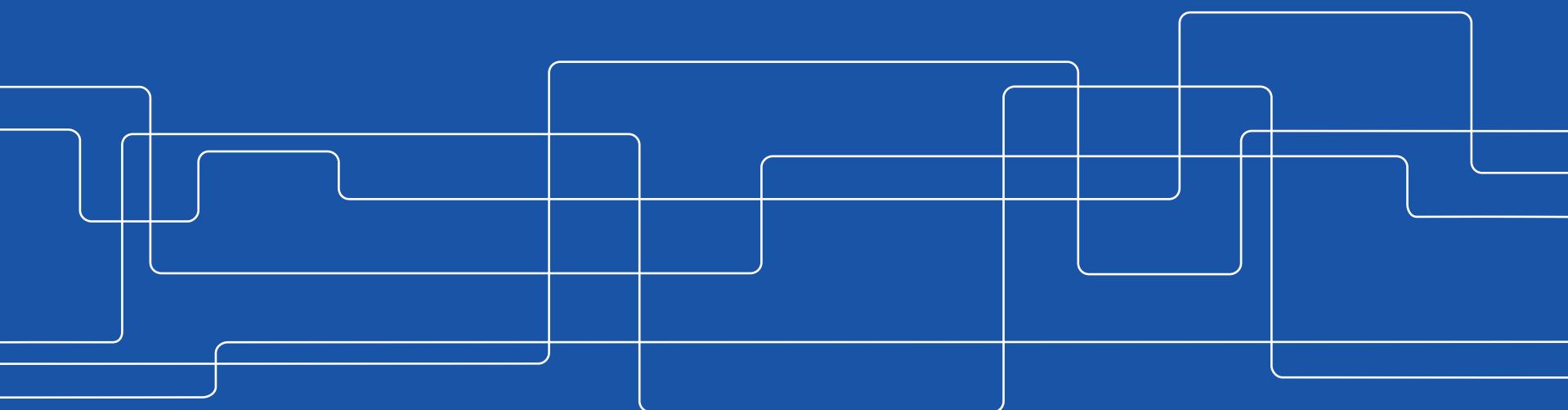
- Markov process + partial observability = HMM
- Markov process + actions = MDP
- Markov process + partial observability + actions = HMM + actions = MDP + partial observability = POMDP

	<i>full observability</i>	<i>partial observability</i>
<i>no actions</i>	Markov process	HMM
<i>actions</i>	MDP	POMDP

Source: https://www2.cs.duke.edu/courses/fall08/cps270/cps270_mdp.pdf



Game Theory





Game theory

- Other agents also introduce uncertainty (what decision do they make?)
- Game theory studies settings where multiple parties (agents) each have
 - different preferences (utility functions)
 - different actions that they can take
- Each agent's **utility** (potentially) depends on all agents' actions
 - What is optimal for one agent depends on what other agents do
- Agents can rationally form **beliefs** over what other agents will do, and (hence) how agents should **act**
 - Important area of research
 - Useful for acting and predicting behavior of others



Game theory

Two main areas:

- Agent design:

What is the best strategy for the individual, assuming others act optimally?

- Mechanism design (define rules of the game):

“Given that agents pick rational strategies, what game should we design?” i.e., How do we construct rules such that the best policy for the individual agents are also for the good of all? (e.g. multi-agent distributed systems), e.g. internet traffic routers



Agent Design: Single Move Games

Main components:

- Players or agents
- Actions
- Payoff-matrix: Gives utility for each player for each combination of actions

Strategy = policy

Example: Prisoner's dilemma

Burglars Alice and Bob are caught
Interrogated separately by the police

Rules of the game:

- Both testify: 5 years in prison each
- Both refuse to confess: 1 year each
- If only one testifies: One person 10 years, other goes free

What to do?



Prisoner's dilemma cont'd

Each agent wants to maximize expected utility

Payoff matrix

	Alice: testify	Alice: refuse
Bob: testify	A=-5, B=-5	A=-10, B=0
Bob: refuse	A=0, B=-10	A=-1, B=-1

Rules of the game:

Both testify: 5 years in prison each

Both refuse to confess: 1 year each

If only one testifies: Person who hasn't testified gets 10 years, but you go free!

Prisoner's dilemma cont'd

Alice analysis:

- Bob testifies: Best to testify ($-5 > -10$)
- Bob refuses: Best to testify ($0 > -1$)

	Alice: testify	Alice: refuse
Bob: testify	A=-5, B=-5	A=-10, B=0
Bob: refuse	A=0, B=-10	A=-1, B=-1

- That is, best to testify! – dominant strategy
- Bob feels the same way.

Nash equilibrium

Nash equilibrium:

- Set of strategies such that no player can benefit by changing her strategy while the other players keep their strategies unchanged

Alice and Bob are stuck in a Nash equilibrium

	Alice: testify	Alice: refuse
Bob: testify	A=-5, B=-5	A=-10, B=0
Bob: refuse	A=0, B=-10	A=-1, B=-1



Game theory

Two main areas:

- Agent design:

What is the best strategy for the individual, assuming others act optimally?

- Mechanism design:

“Given that agents pick rational strategies, what game should we design?” i.e., How do we construct rules such that the best policy for the individual agents are also for the good of all? (e.g. multi-agent distributed systems)



Mechanism design example: Auctions

- Single good being auctioned
- Each player has utility value v_i for the good
- Utility value known only to the bidder
- Bidders make bids b_i , highest bid wins

Example: English auction (ascending bid)

- Bids are incremented
- Continues until only one bidder
- Bidder with the highest v_i wins
- Resulting Strategy: bid as long as price below your value
- Result:
 - Pays $b_m + d$ (b_m highest among other, d =increment)
- Requires a lot of communication



Example: Sealed bid auction

- Each bidder makes a single bid
- Highest bid wins
- Resulting Strategy: bid $\min(v_i, b_m + \varepsilon)$
 - (b_m believed max of other bidders)
- Player with highest v_i might not get the good
- Requires less communication but more “work”
(need to estimate b_m)



Example: Vickrey auction

- Same as sealed-bid auction but winner pays second highest bid
- Resulting Strategy: bid your own value v_i
- Simple and minimal communication



Mechanism design

A good design benefits both the individual and the system!
Can be applied to many areas of society as well

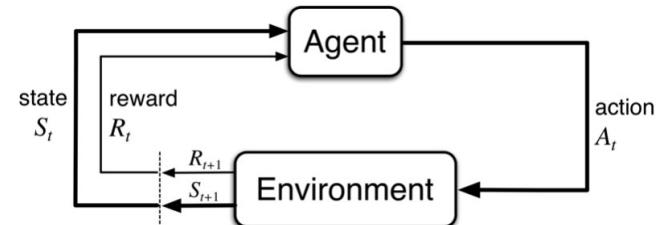


End of lecture

Markov Decision Process (MDP) Formulation

Mathematical model: Markov Decision Process (S, A, T, R, γ)

- State space S
- Action space A
- Environment Transition model T
- Reward function R
- Discount factor γ
- A start state S_0
- Maybe a terminal state



$$T(s, a, s') : p(s_{t+1} = s' | s_t = s, a_t = a)$$

$R(s, a, s')$: immediate reward for transitioning from state s to state s' due to action a

Frequently reward depends only on the state, so we usually write $R(s)$