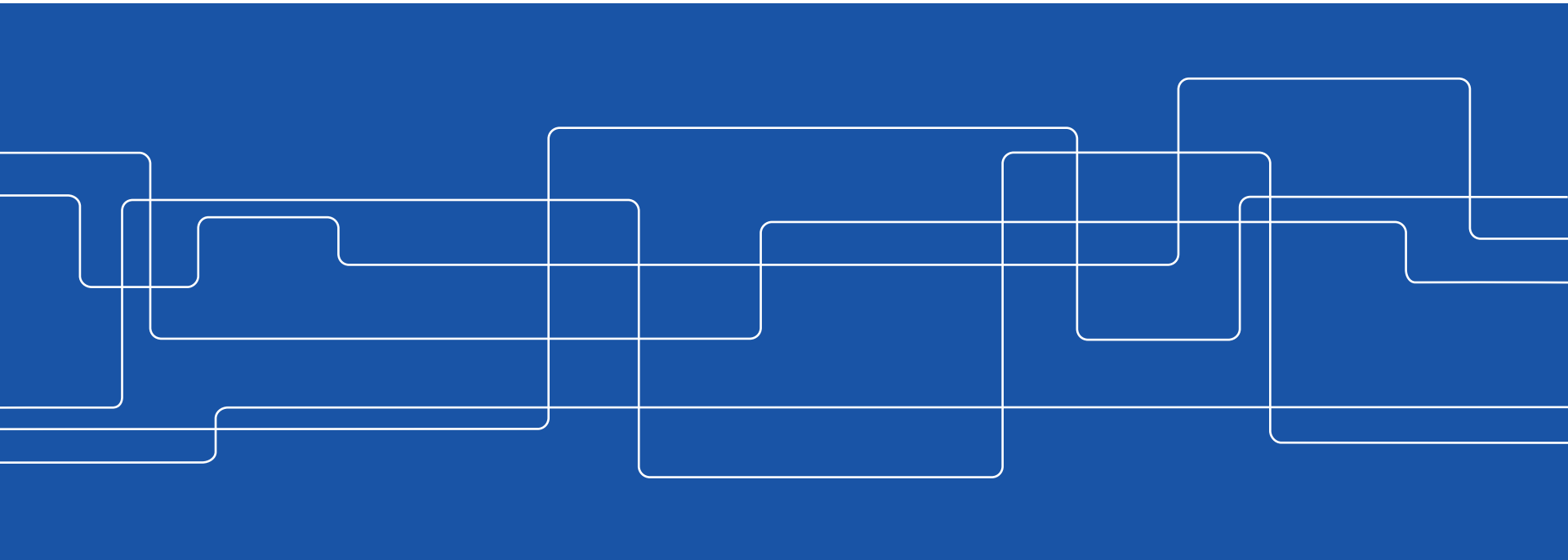
KTH ROYAL INSTITUTE
OF TECHNOLOGY

DD2380 Artificial Intelligence

Making decisions under uncertainty

Iolanda Leite



Additional 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!

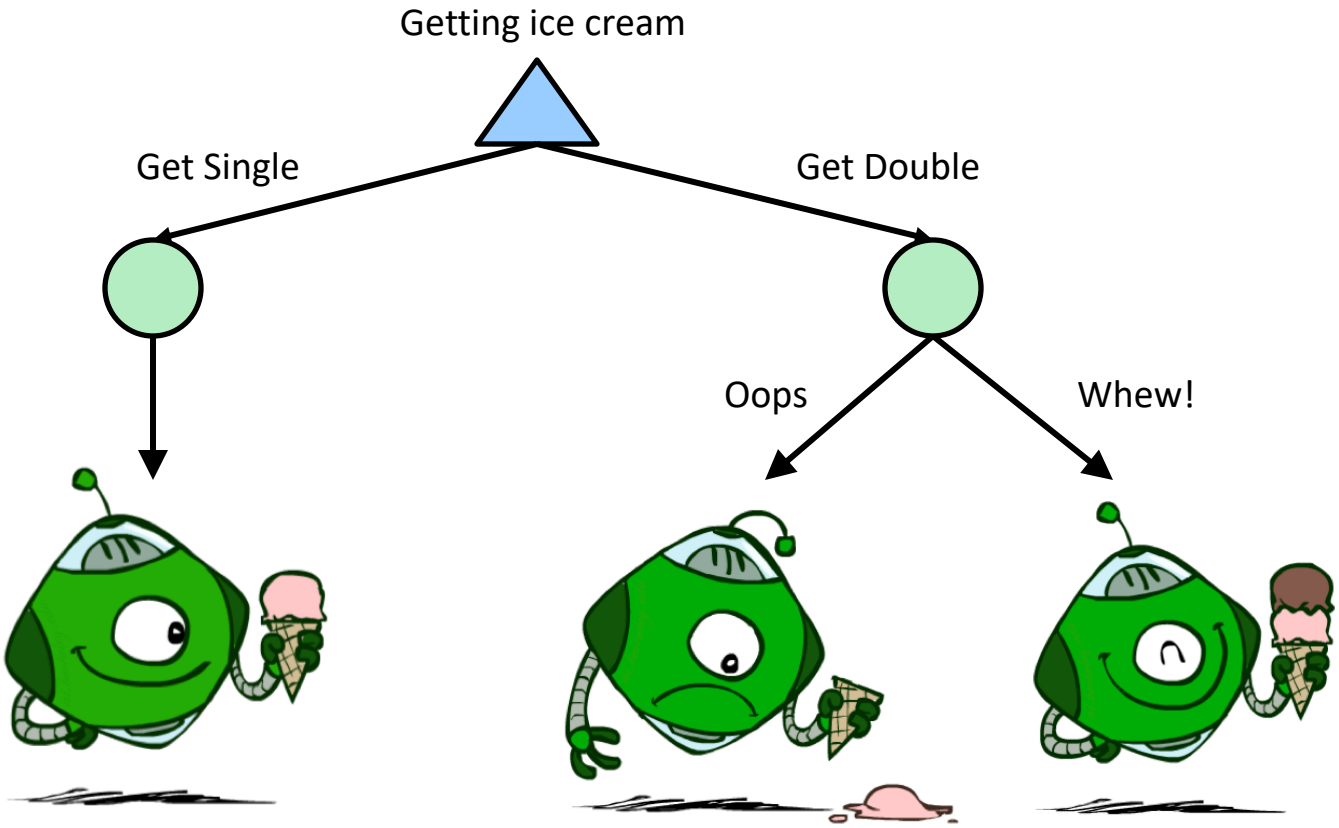
Foundation for many of the cool things happening now in reinforcement learning

Utility function

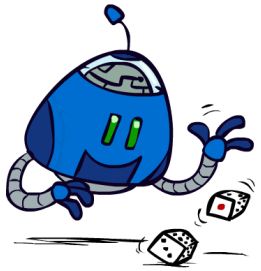
- Captures an agent's preference between world states
  - See also module **Problem solving**

- Assigns single number to express desirability of a state
- The utility of state S is denoted by *U(S)*

Utilities: Uncertain Outcomes



Maximum Expected Utility (MEU)



- Outcome of action is non-deterministic
- Result of action A is **Result $_i$ (A)** (a state)
- Given evidence **E (measurements)** the probability for each result is

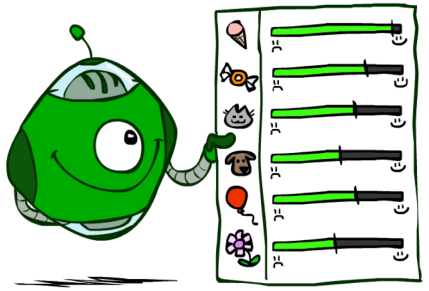
$$P(Result_i(A) \mid Do(A), E)$$

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

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

MEU not so easy to apply

- State of the world? (partially observable at best)
- How to compute $P(Result_i(A)|Do(A),E)$? 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?

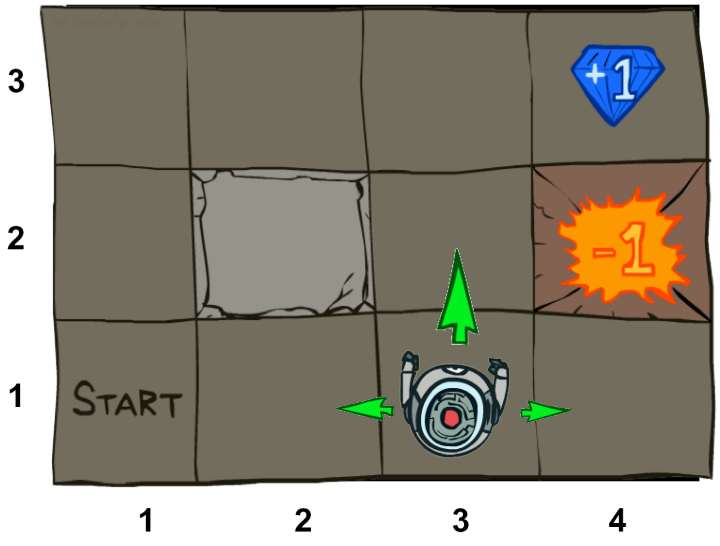
- Information has **value** if it might change your action
- Value = difference between **expected value with and without** the information

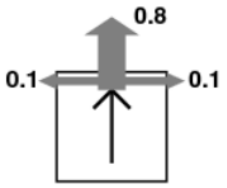
Sequential decisions

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

Seq. decisions example: Grid World

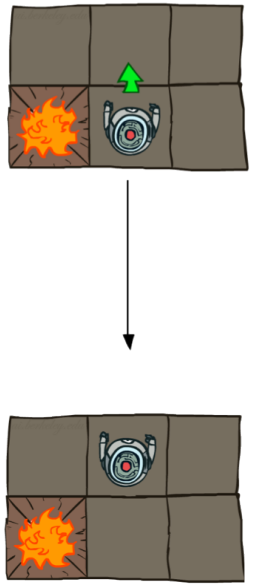
- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions (Up, Down, Left, Right) do not always go as planned
  - 80% of the time, the intended outcome occurs (if there is no wall there)
  - 20% of the time, the agent moves at right angles to the intended 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





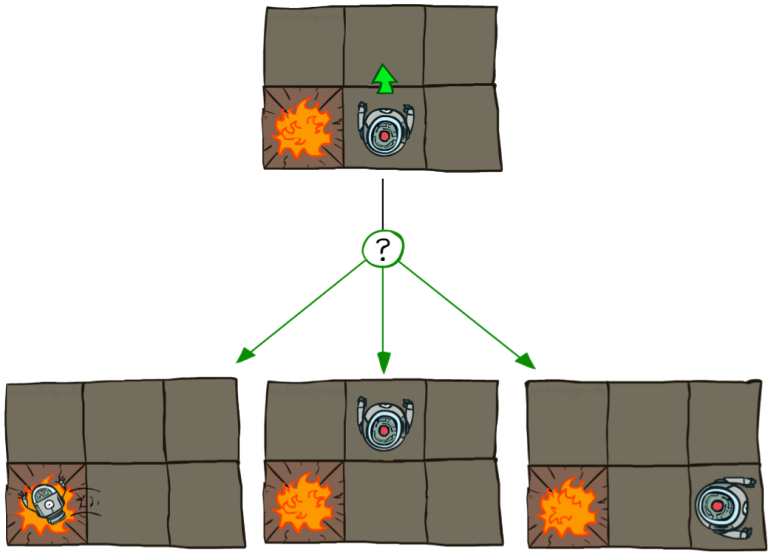
Grid World Actions

Deterministic Grid World



Plan: Up, Up, Right, Right, Right

Stochastic Grid World



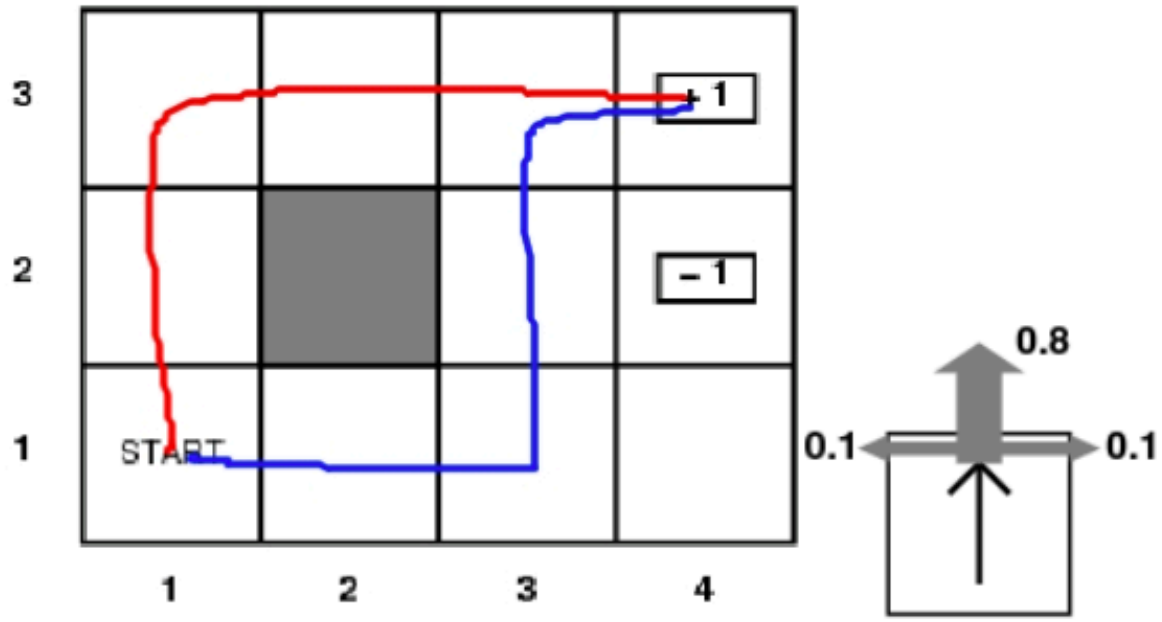
Cannot create a plan ahead of time!

What is the probability that a predefined plan succeeds?

Plan: Up, Up, Right, Right, Right

Probability to succeed: 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

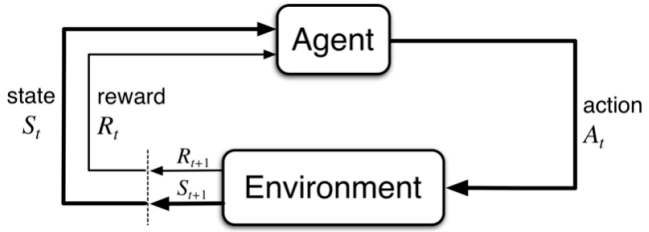
- The transition function defines the probability transitioning from one state to another.
- In decision making, each action is associated with its own transition function
  → select action to control the system to behave in some desired way or maximize the chance of achieving some goal

"Markov" generally means that given the present state, the future and the past are independent

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



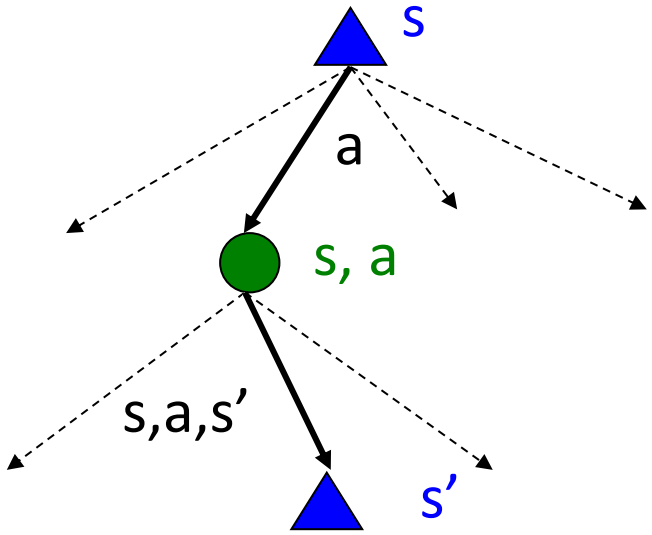
$$T(s, a, s') : p(s_{t+1} = s' \mid s_t = s, a_t = a)$$

$R(s, a, s')$: 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

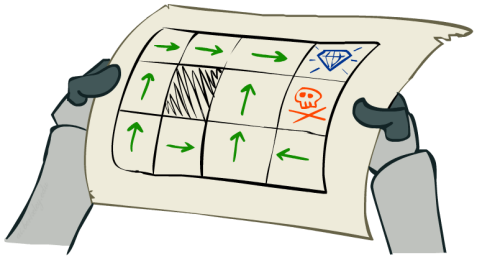
- Now function of the action
- T(s,a,s') (first order Markov assumption*)
  Probability to reach s' starting from s given action a.



* 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)

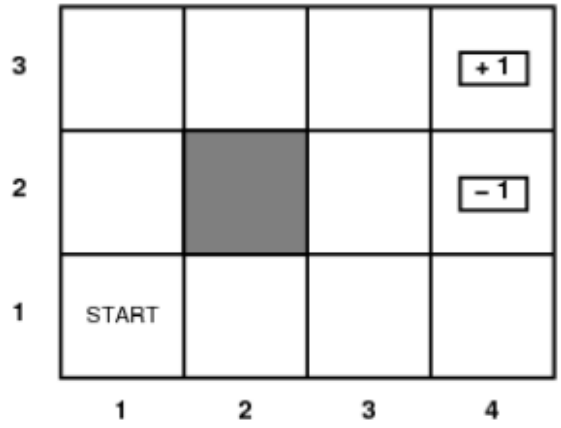
For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$

- A policy π gives an action for any state
- An optimal policy is one that maximizes expected utility if followed

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(rest) = -0.04 (Get to goal quickly)

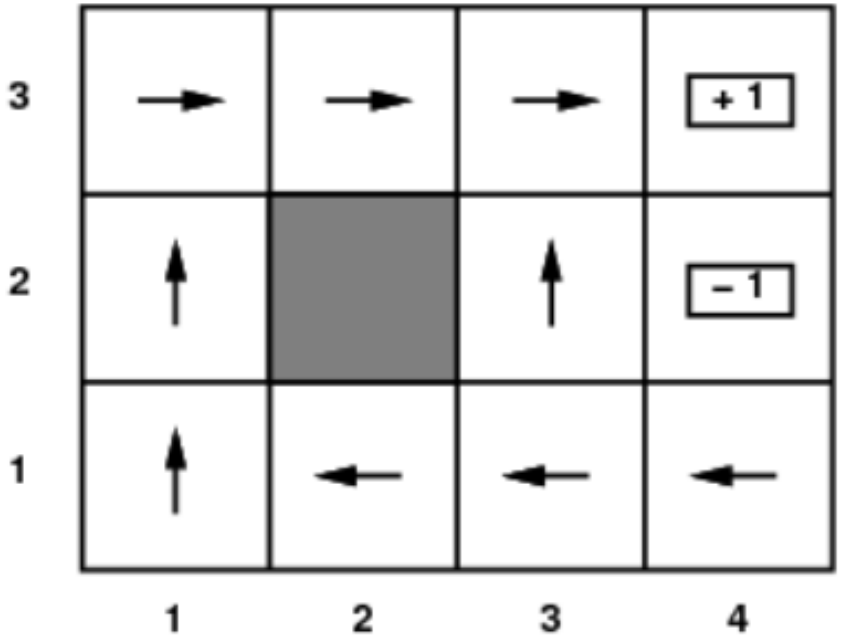


Ex: Optimal policy π^*

Optimal policy for the previous problem

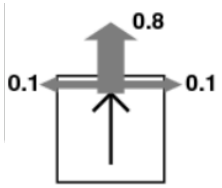
Tells us what to do in each state

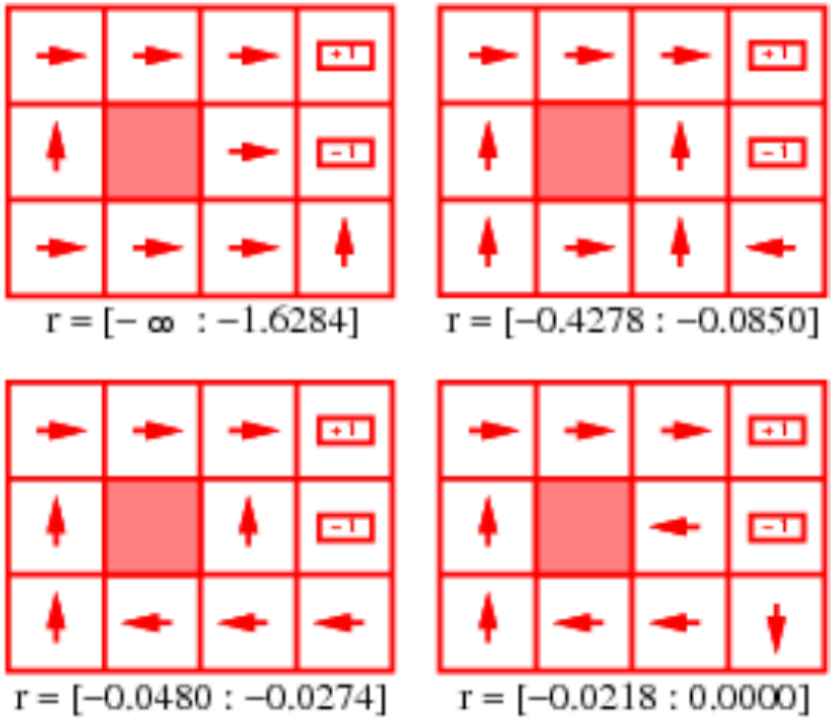
Actual path only known when moving because actions are non-deterministic



Optimal policy depends on R and T

Different behaviors for different R





How about changing T? (see **stationary** vs nonstationary policies)

How good is a policy?

How to measure the quality of a policy?

→ Measure expected utility over the history
(stochastic Env means that we need to use expectations)

Optimal policy, π^*: highest possible expected utility

Utility function?

Typically something that is part of our job as engineers to achieve a certain behavior

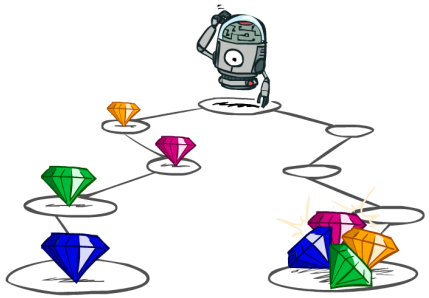
Utility of sequences

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

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]



Utility of sequences

Additive rewards:

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

What about infinite sequences?

- Might get ∞ without terminal state.
- How to compare ∞ and ∞?

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< γ<1

$$U_h([s_0, s_1, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

- Gives bounded utility

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

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

Selecting the best policy

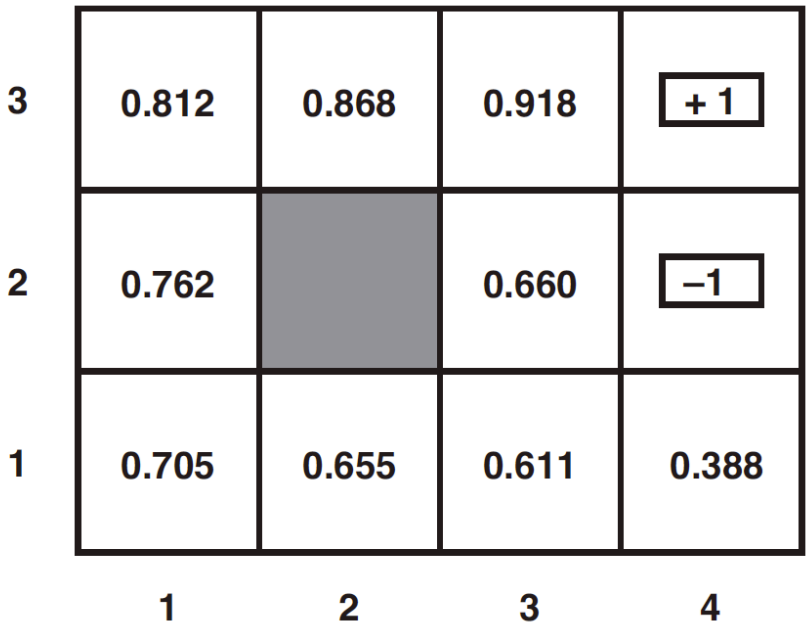
- How do we select the best policy?
  - i.e. choice of action for each state
- Many state sequences to compare…
- As before, maximize expected utility

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

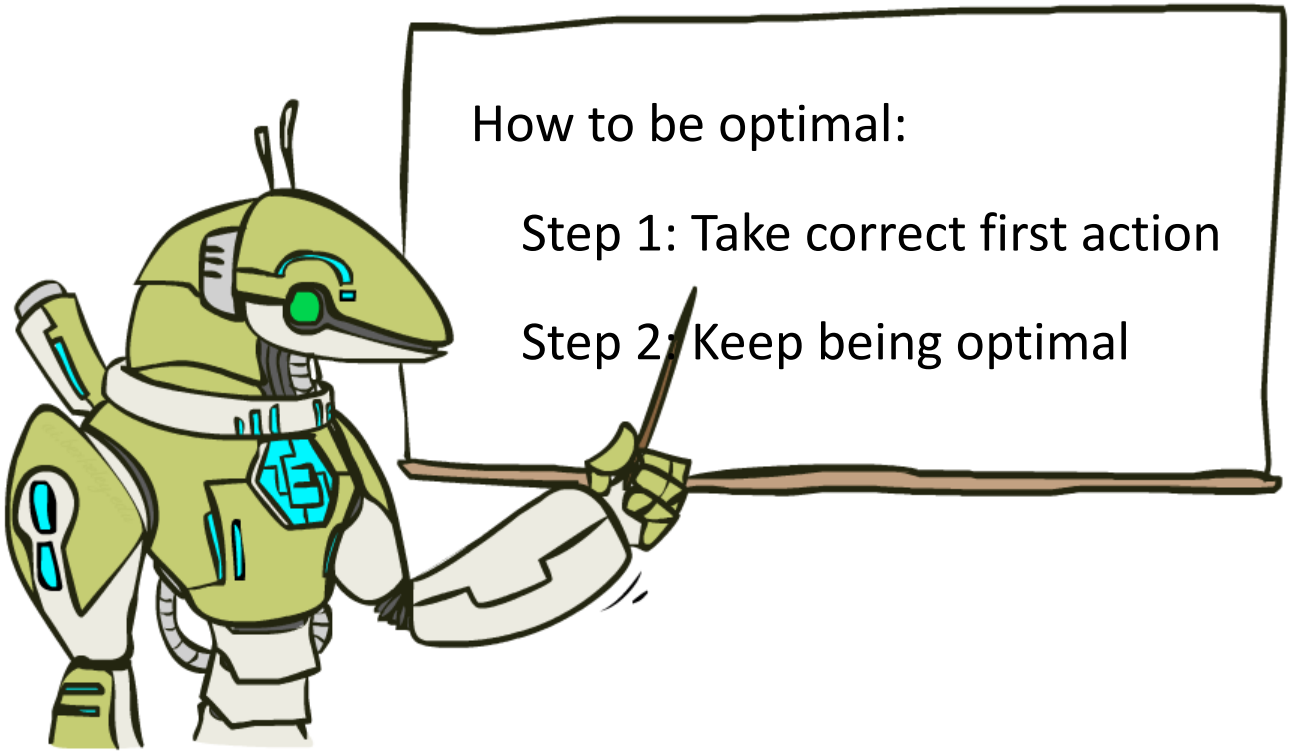
sum of discounted rewards

Utilities of States

The utilities of the states in the 4X3 world, calculated with γ =1 and R(s)= − 0.04 for nonterminal states.



Bellman Equation



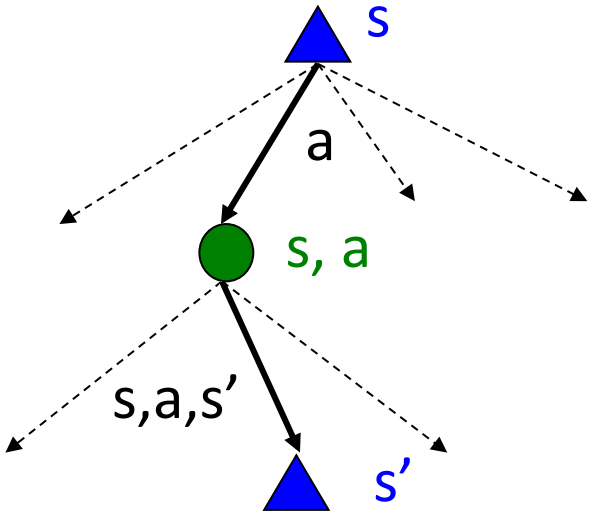
Bellman equation

Bellman equation

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

immediate reward

discounted expected utility of the next state, assuming optimal action



Converges to unique optimal solution

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

Can show that

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

Value iteration

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

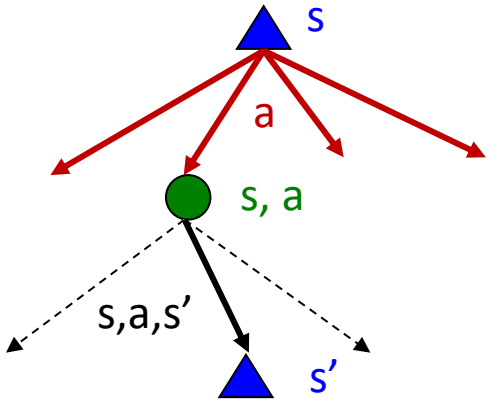
$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$

(assuming that we choose the optimal policy)

Idea: Iterate

- Calculate utility of each state
- Use utilities to select optimal decision in each state

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

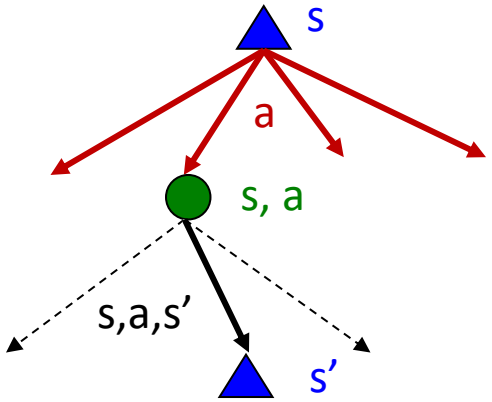
end

NOTE: Q-function comes back in Reinforcement learning

Problems with Value Iteration

Value iteration repeats the Bellman updates:

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



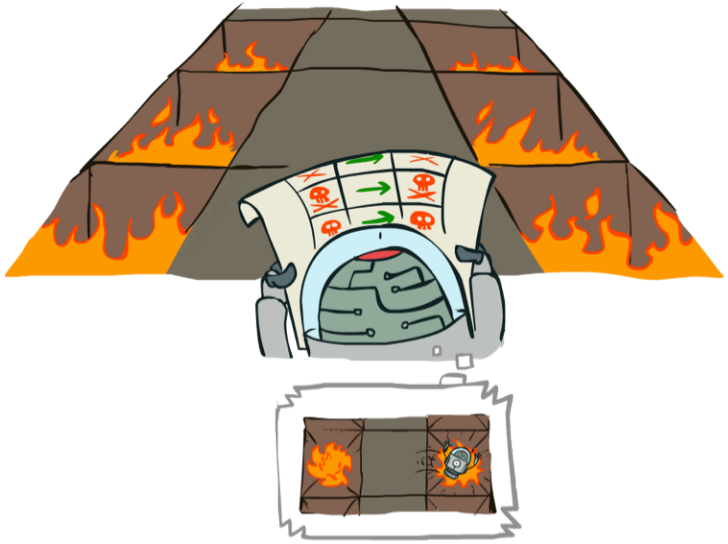
Problem 1: It's slow – $O(S^2A)$ per iteration

Problem 2: The "max" at each state rarely changes

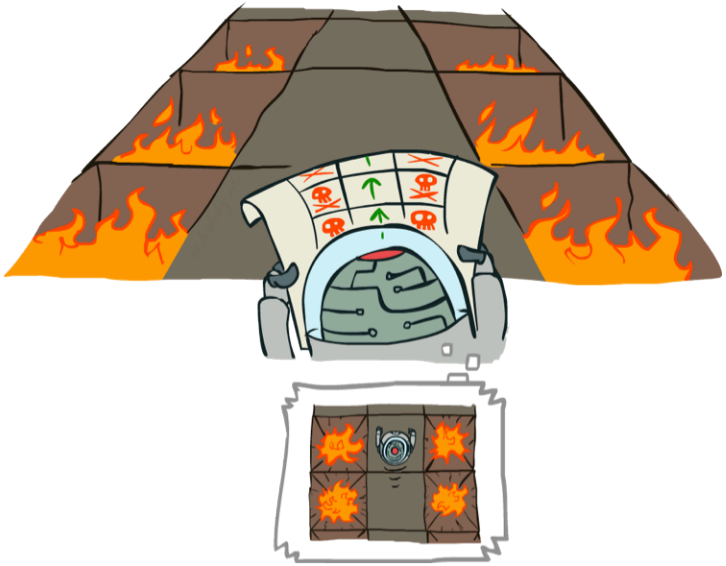
Problem 3: The policy often converges long before the values

Policy Evaluation

Always Go Right



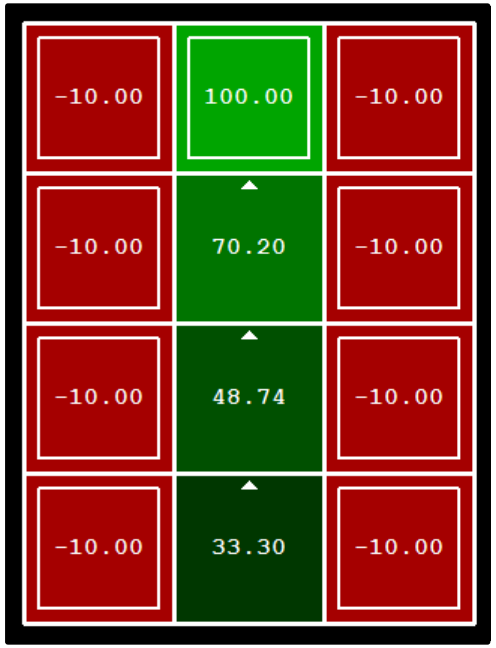
Always Go Forward



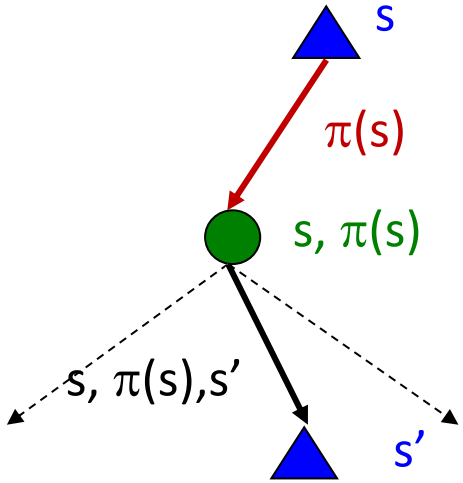
Example: Policy Evaluation

Always Go Right

Always Go Forward



Algorithm: Policy iteration



Choose an arbitrary policy

Loop until policy does not change any more

1. Policy evaluation: Compute the value function, V(s) given the fixed policy (not optimal values):

$$V(s) = \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') \left[R(s, \pi(s), s') + \gamma V(s') \right], \forall s \in \mathcal{S}$$

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

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

Value Iteration vs. Policy Iteration

Both value iteration and policy iteration compute the same thing (all optimal values)
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

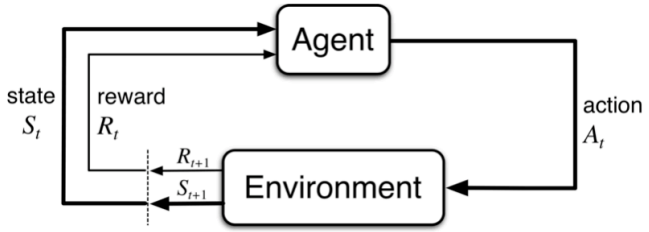
In **policy iteration**:

- 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



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

$R(s, a, s')$: 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 value (utility) of a state s:

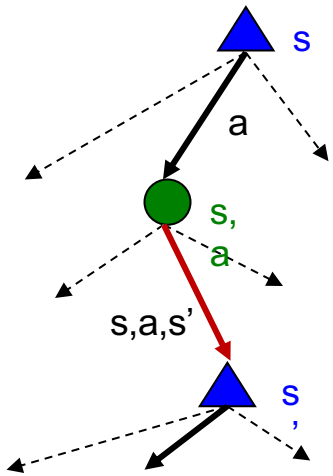
  $V^*(s)$ = expected utility starting in s and acting optimally

- The value (utility) 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



s is a *state*

(s, a) is a *q-state*

(s,a,s') is a *transition*

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

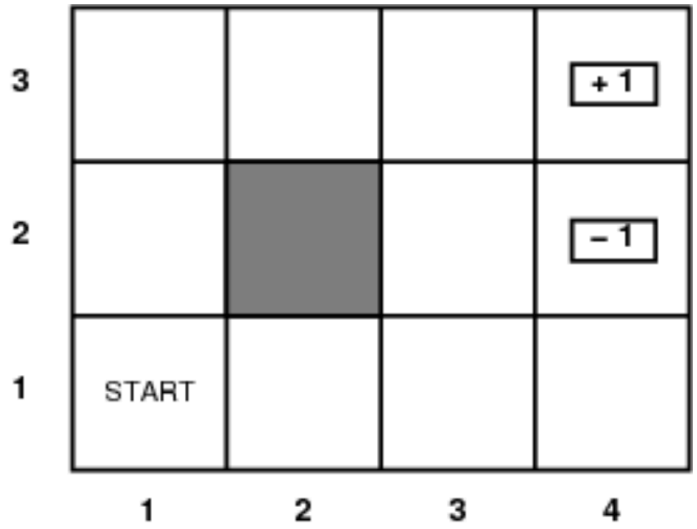
- What if the environment is not observable?
- Cannot execute policy since the state is unknown!
- Results in a Partially Observable MDP aka **POMDP**
- Sensors provide observations of the environment
- Observation model O(s,o) gives probability of making observation o in state s

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’) = some function of b(s), a, s and o….

Belief state update

- Need to update the belief state as we go along
- 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”)
- 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’}

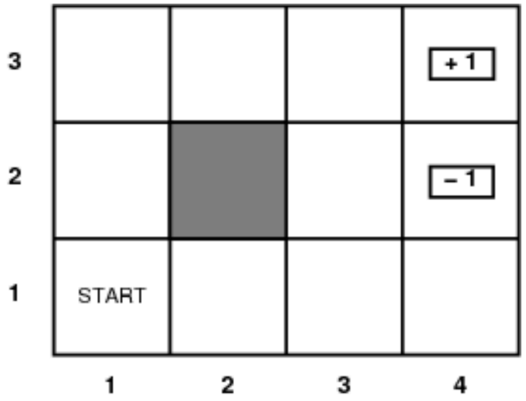
Solving POMDP

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 a MDP

Introduce

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

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

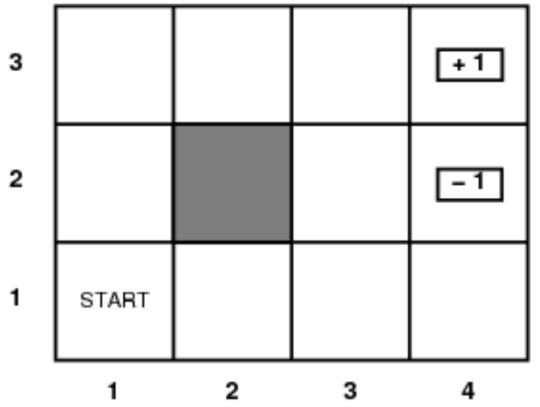
Optimal MDP strategy $\pi^*(b)$ is also optimal for the original POMDP

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!!!

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

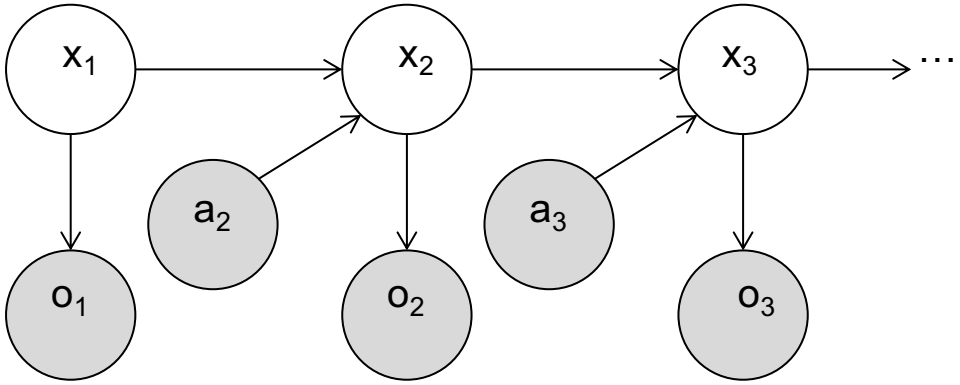


POMDP Connection to HMM?

How does this relate to a Hidden Markov Model?

POMDP Connection to HMM?

- The next state (position) depends only on the previous state (and the action)
- Our transition matrix is a function of action!
- Now also uncertain observation like in HMM case but unlike MDP



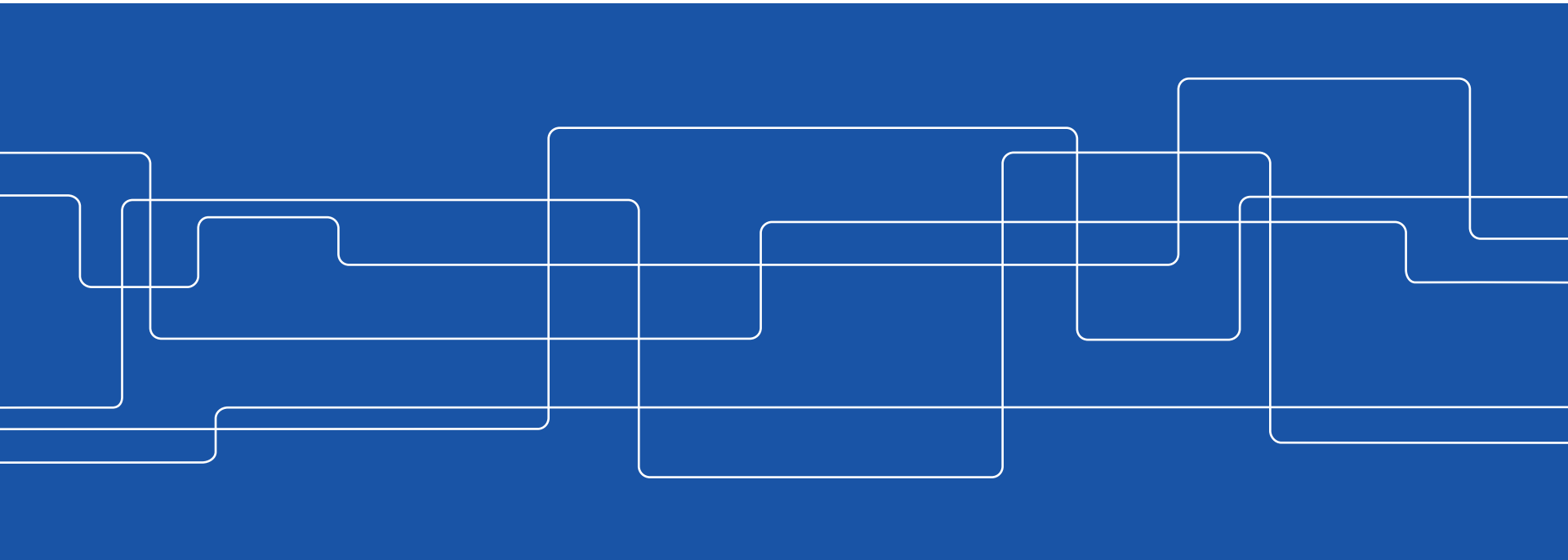
The big picture

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

	full observability	*partial observability*
no actions	**Markov process**	**HMM**
actions	**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 not only to make money at gambling
    - 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:

“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)

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 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: One person 10 years, other goes 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
- Each player has utility value $\mathbf{v_i}$ for the good
- Utility value known only to the bidder
- Bidders make bids $\mathbf{b_i}$, highest bid wins

Example: English auction (ascending bid)

- Bids are incremented
- Continues until only one bidder
- Bidder with the highest v_i wins
- 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
- 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
- Strategy: bid your own value $\mathbf{v_i}$
- 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

Credits

Original slides:

- Patric Jensfelt, KTH

Based partly on materials from:

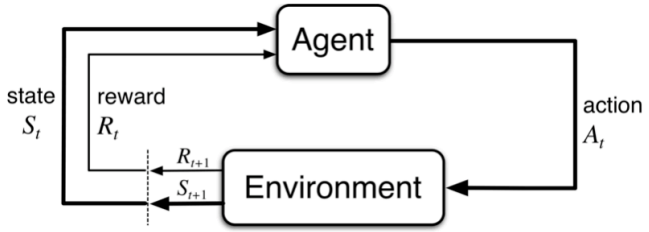
- Berkeley AI
- Duke University AI (CPS 270)

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



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

$R(s, a, s')$: 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 $R(s)$