Introduction to Artificial Intelligence (AI)

Computer Science cpsc502, Lecture 12

Oct, 20, 2011

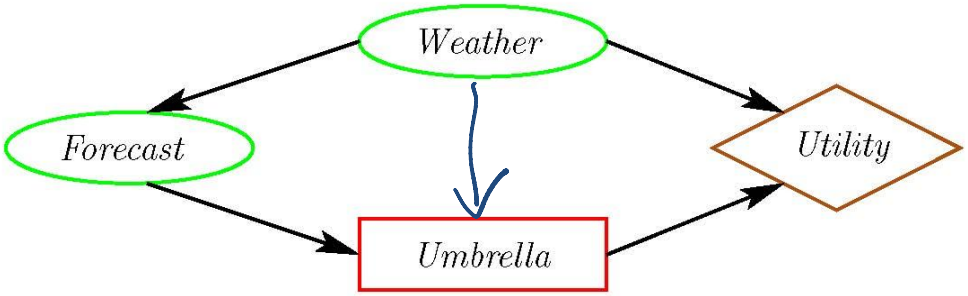
Today Oct 20

Value of Information and value of Control

Markov Decision Processes

- Formal Specification and example
- Policies and Optimal Policy
- Value Iteration
- Rewards and Optimal Policy

Value of Information



- What would help the agent make a better *Umbrella* decision?

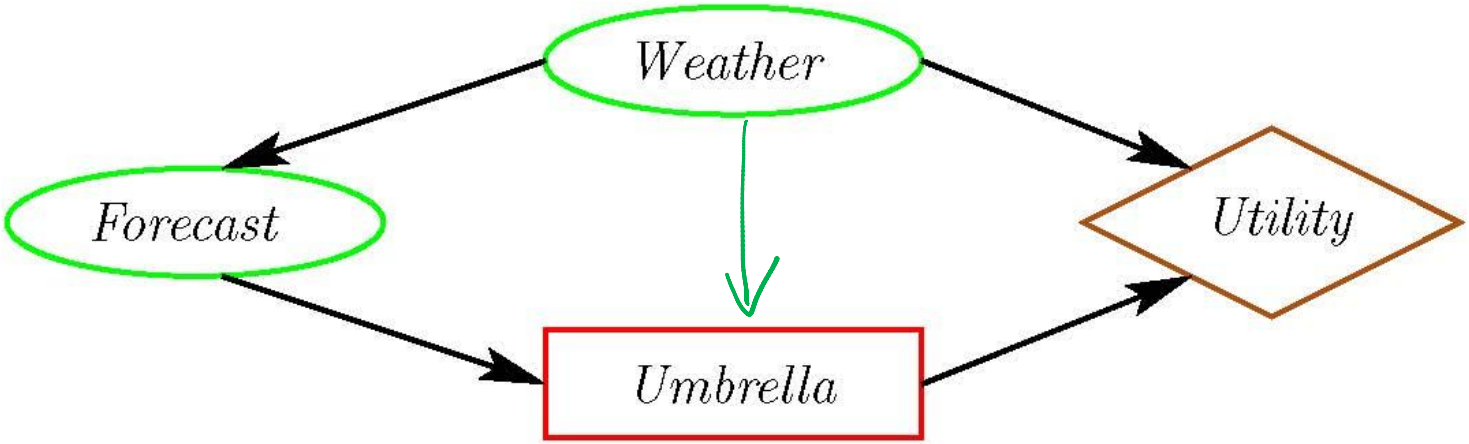
- The value of information of a random variable X for decision D is: $EU(\text{Knowing } X) - EU(\text{not Knowing } X)$

the utility of the network with an arc from X to D **minus** the utility of the network without the arc.

- Intuitively:
  - The value of information is always ≥ 0
  - It is positive only if the agent changes its policy

Value of Information (cont.)

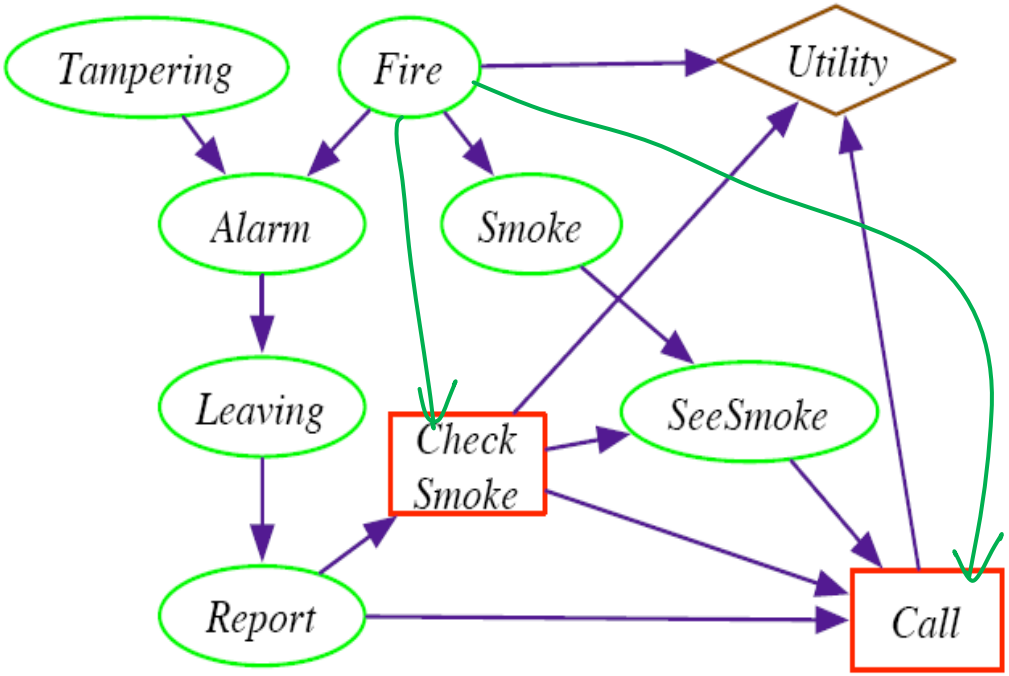
- The value of information provides a bound on how much you should be prepared to pay for a sensor. How much is a **perfect** weather forecast worth?



- Original maximum expected utility: 77
- Maximum expected utility when we know Weather: 91
- Better forecast is worth at most: 14

Value of Information

- The value of information provides a bound on how much you should be prepared to pay for a sensor. How much is a **perfect** fire sensor worth?

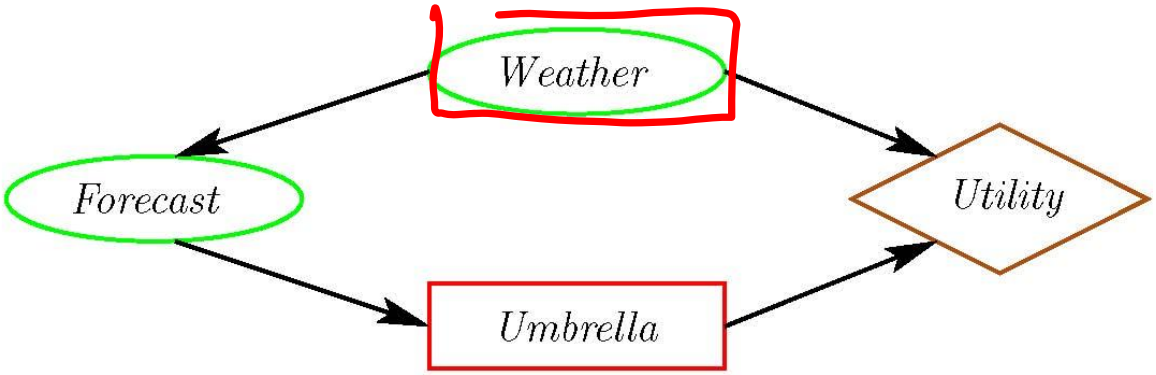


- Original maximum expected utility: -22.6
- Maximum expected utility when we know Fire: -2
- Perfect fire sensor is worth: 20.6

Value of control

- The value of control of a variable X is the utility of the network when you make X a decision variable minus the utility of the network when X is a random variable.

- What if we could control the weather?



- Original maximum expected utility: 77
- Maximum expected utility when we control the weather: 100
- Value of control of the weather: 23

Today Oct 20

Value of Information and value of Control

Markov Decision Processes

- Formal Specification and example
- Policies and Optimal Policy
- Value Iteration
- Rewards and Optimal Policy

Combining ideas for Stochastic planning

- What is a key limitation of decision networks?

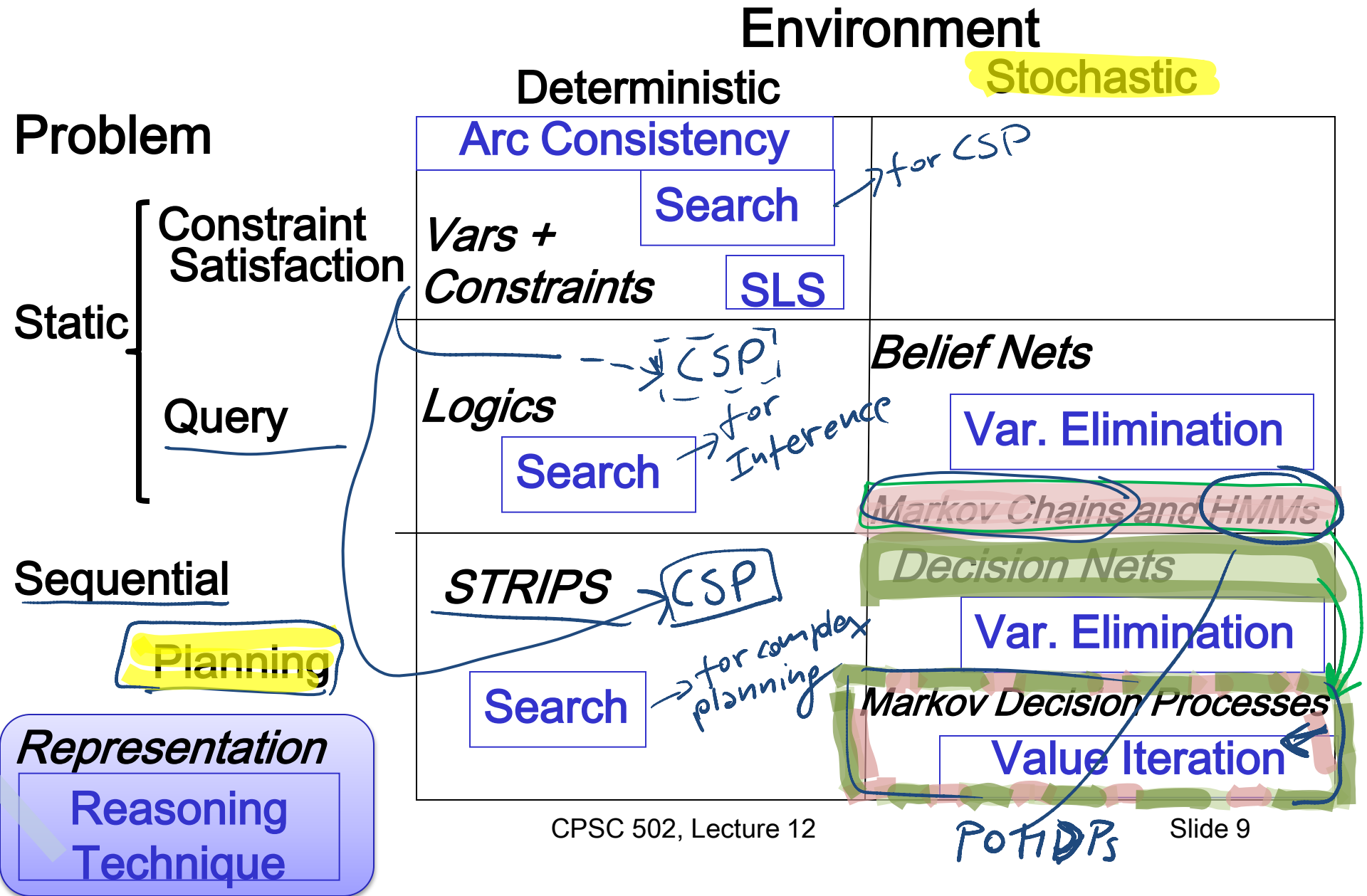
  Represent (and optimize) only a fixed number of decisions

- What is an advantage of Markov models?

  The network can extend indefinitely

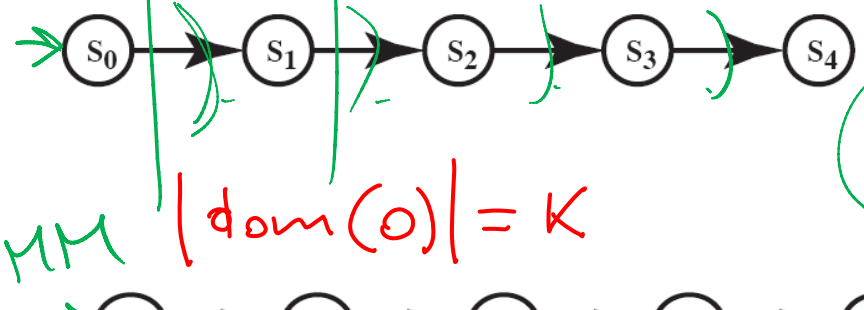
Goal: represent (and optimize) an indefinite sequence of decisions

Planning in Stochastic Environments



Recap: Markov Models

Tables



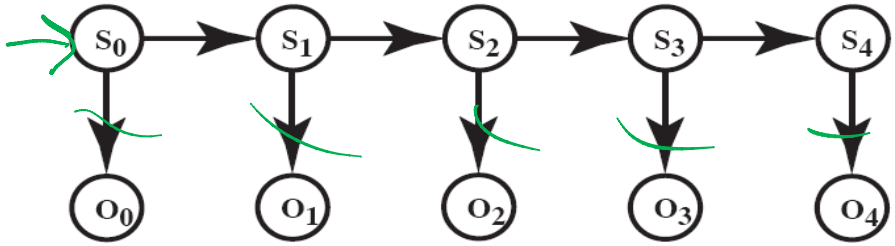
$|dom(S)| = n$

$P(S_0)$ 1:n

$P(S_{t+1} | S_t)$ n:n

HMM

$|dom(O)| = K$

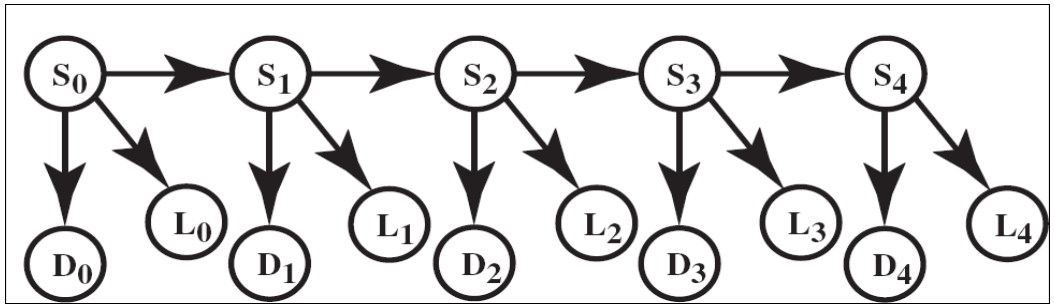


$P(O_t | S_t)$ n:K

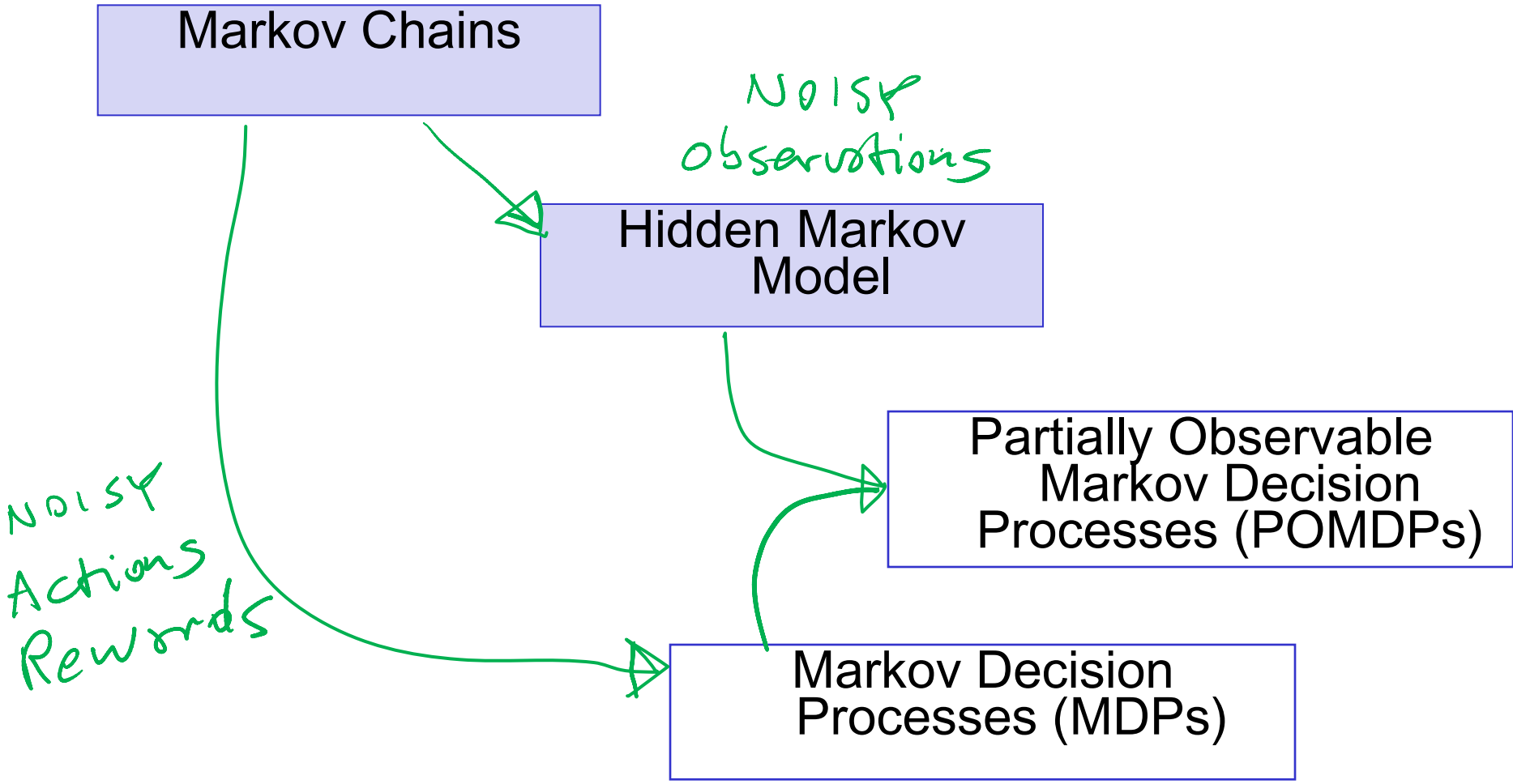
extended HMM

multiple sensors

"sensor fusion"



Markov Models



Decision Processes

Often an agent needs to go beyond a fixed set of decisions – Examples?

- Would like to have an **ongoing decision process**

Infinite horizon problems: process does not stop

Robot surviving on planet, Monitoring Nuc. Plant, ...

Indefinite horizon problem: the agent does not know when the process may stop

reaching location

Finite horizon: the process must end at a give time N

in N steps

How can we deal with indefinite/infinite processes?

We make the same two assumptions we made for….

The action outcome depends only on the current state

Markov

Let S_t be the state at time t …

$P(S_{t+1} | S_t, A_t, S_{t-1}, A_{t-1} \ldots)$

$P(S_{t+1} | S_t, A_t)$

the same for all t

The process is *stationary*…

We also need a more flexible specification for the utility. How?

- Defined based on a reward/punishment *R(s)* that the agent receives in each state *s*

$S_0 \; S_1 \; \ldots\ldots \; S_n$

eg. $\sum$ $r_0 \; r_1 \; \ldots\ldots \; r_n$

MDP: formal specification

For an MDP you specify:

- set S of states and set A of actions
- the process' dynamics (or ***transition model***)

  $P(S_{t+1}/S_t, A_t)$
- The **reward function**

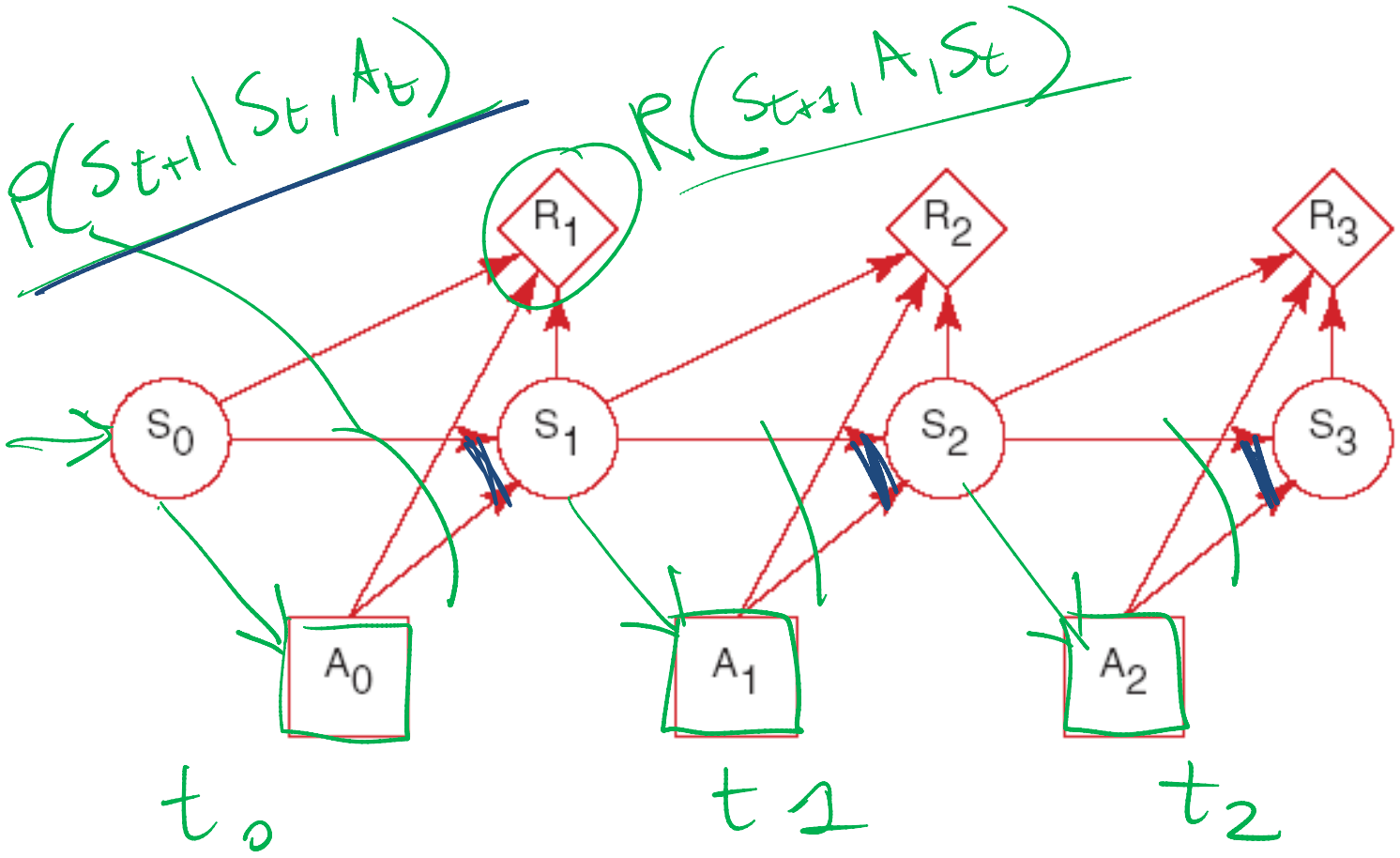
  $R(s, a, s')$

  describing the reward that the agent receives when it performs action *a* in state *s* and ends up in state *s'*
- $R(s)$ is used when the reward depends only on the state s and not on how the agent got there
- **Absorbing/stopping/terminal state** $\leftarrow s_{ab}$

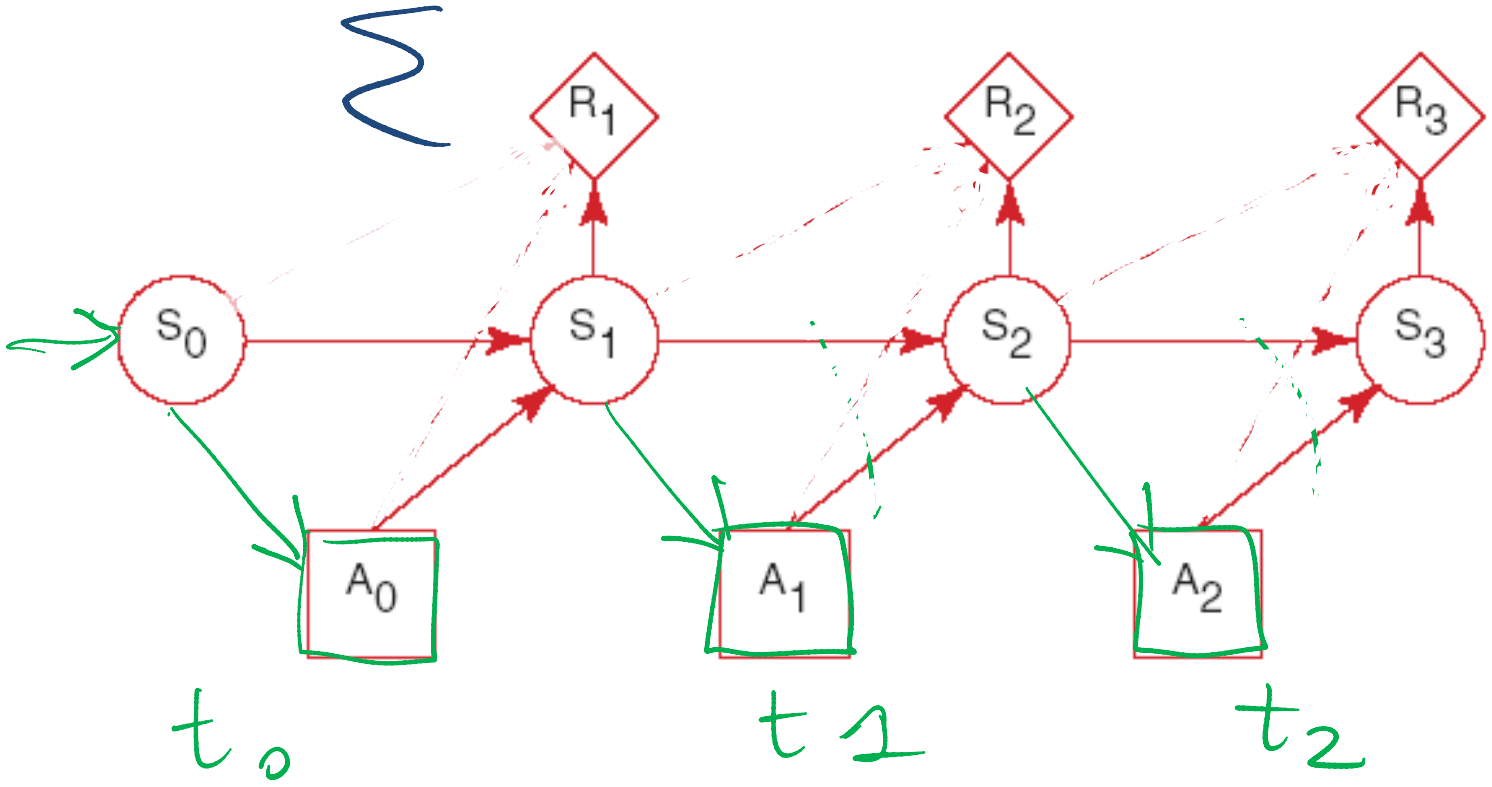
for all action $P(s_{ab} \mid a, s_{ab}) = 1$ $R(s_{ab}, a, s_{ab}) = 0$

MDP graphical specification

Basically a MDP augments a Markov Chain augmented with actions and rewards/values

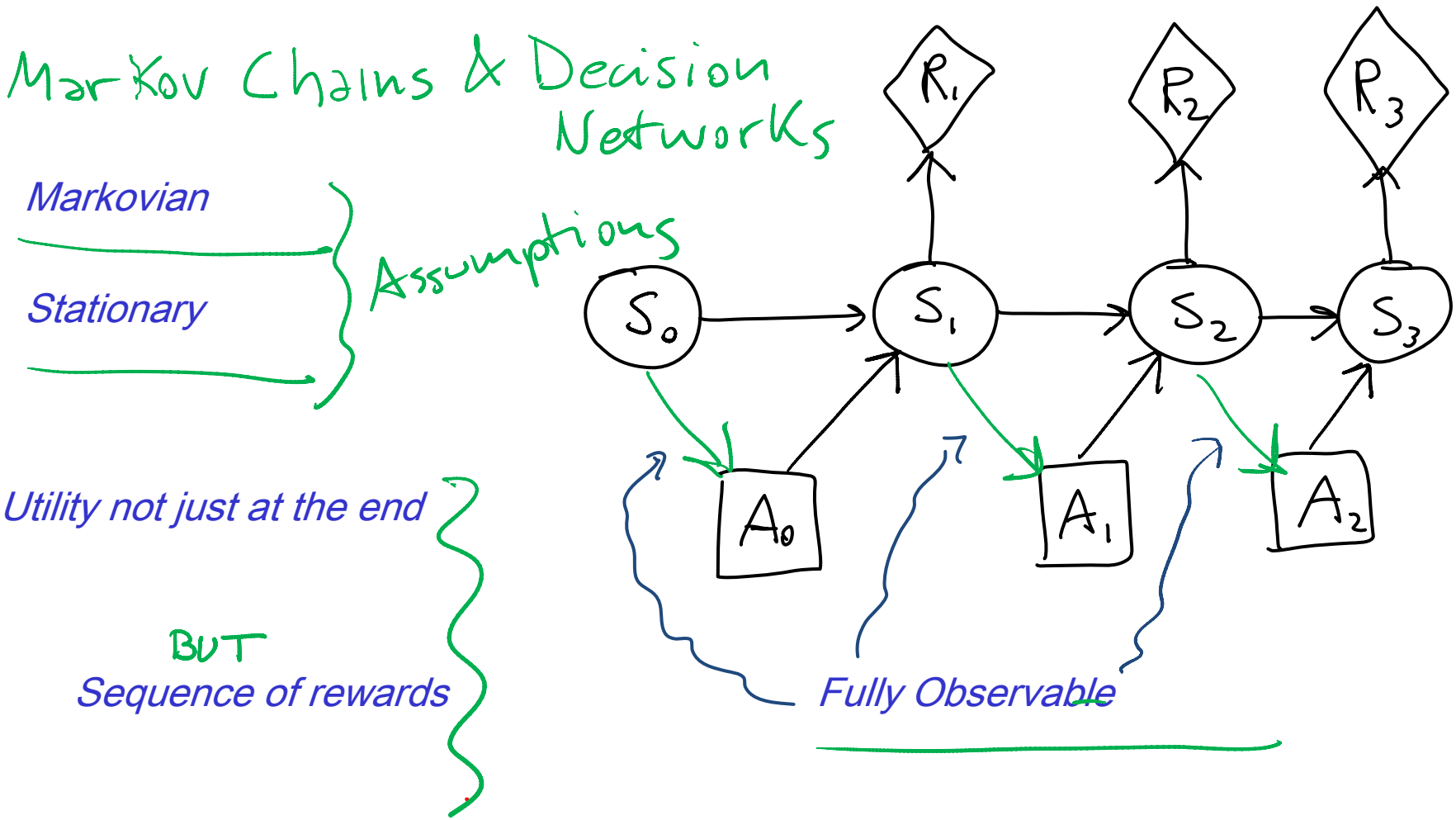


When Rewards only depend on the state

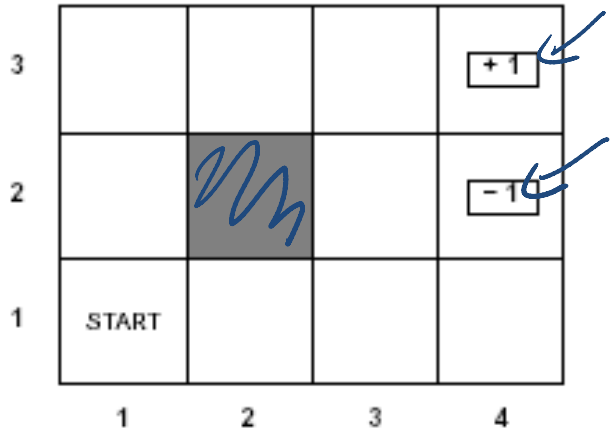


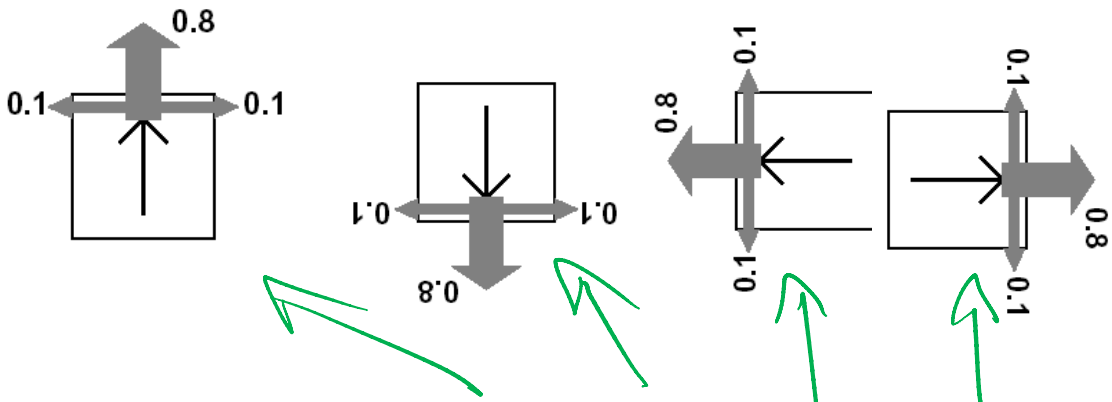
Decision Processes: MDPs

To manage an ongoing (indefinite… infinite) decision process, we combine….



Example MDP: Scenario and Actions





Agent moves in the above grid via actions *Up, Down, Left, Right*

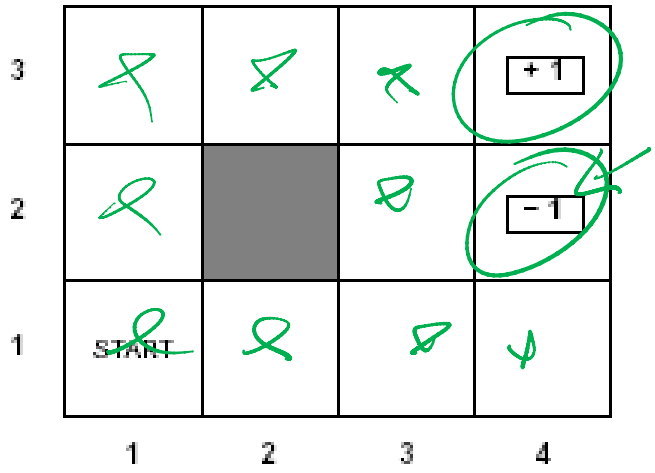
Each action has:

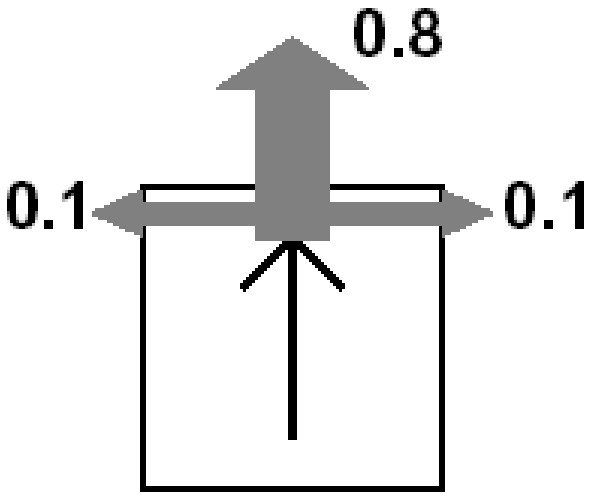
- 0.8 probability to reach its intended effect
- 0.1 probability to move at right angles of the intended direction
- If the agents bumps into a wall, it says there

How many states?

There are two terminal states (3,4) and (2,4)

Example MDP: Rewards



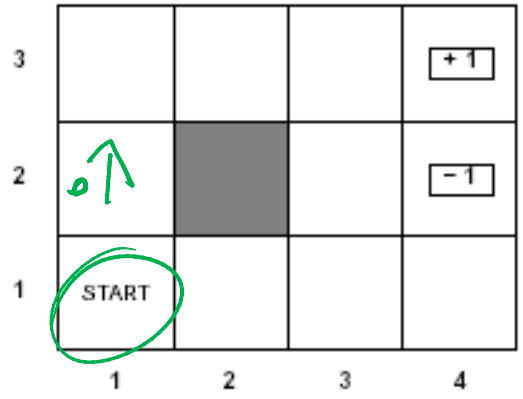


$$R(s) = \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Example MDP: Underlying info structures

Four actions *Up, Down, Left, Right*

Eleven States: {(1,1), (1,2)…… (3,4)}



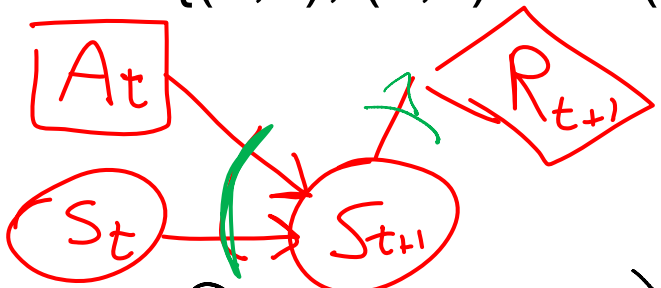


Table 4×11×11 $P(S_{t+1} \mid S_t, A_t)$

Up

	1,1	2,1	1,2	3,1
1,1	.1	.8	.1	0
1,2	0	.2	0	.8
⋮				

Down L R

terminal states ?

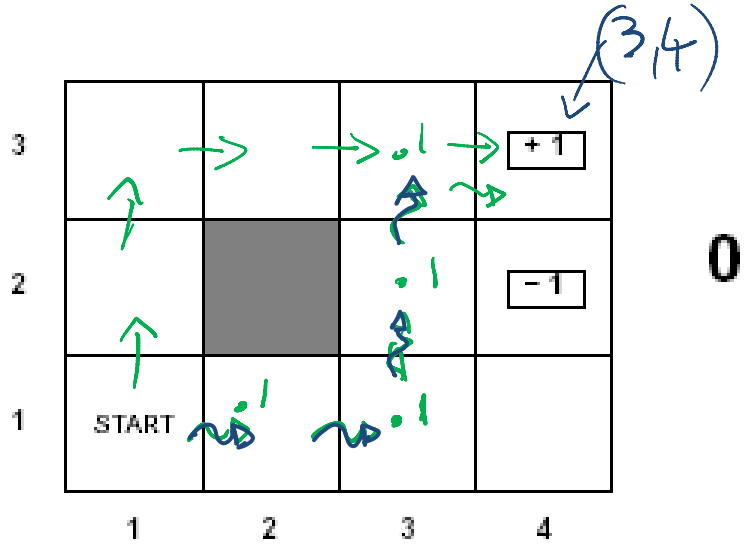
$P(S_0)$

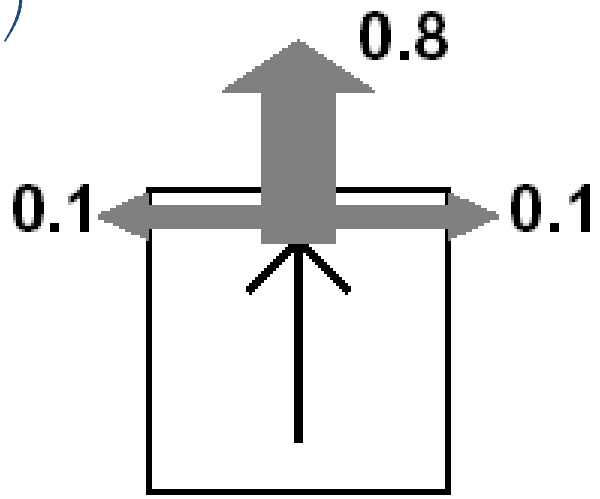
1,1	1
⋮	0
	0
	0
	0
	0

$R(S)$

1,1	-.04
⋮	
(2,4)	-1
(3,4)	+1

Example MDP: Sequence of actions





Can the sequence [*Up, Up, Right, Right, Right*] take the agent in terminal state (3,4)?

Can the sequence reach the goal in any other way?

Today Oct 20

Value of Information and value of Control

Markov Decision Processes

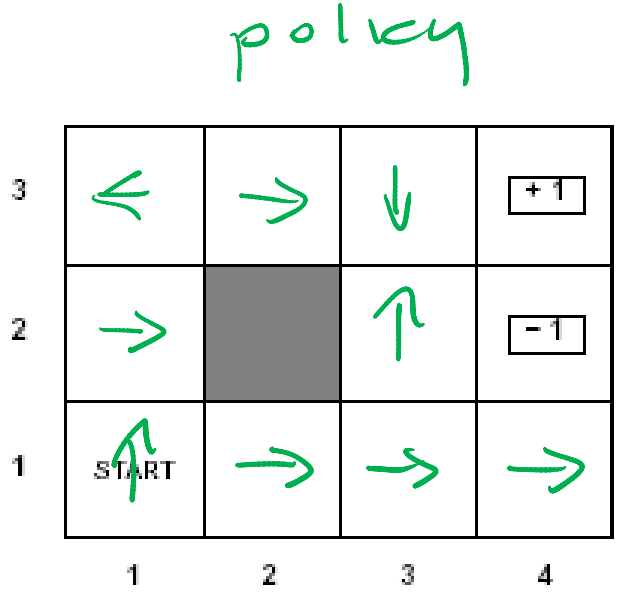
- Formal Specification and example
- **Policies and Optimal Policy**
- Value Iteration
- Rewards and Optimal Policy

MDPs: Policy

- The robot needs to know what to do as the decision process unfolds…
- It starts in a state, selects an action, ends up in another state selects another action….

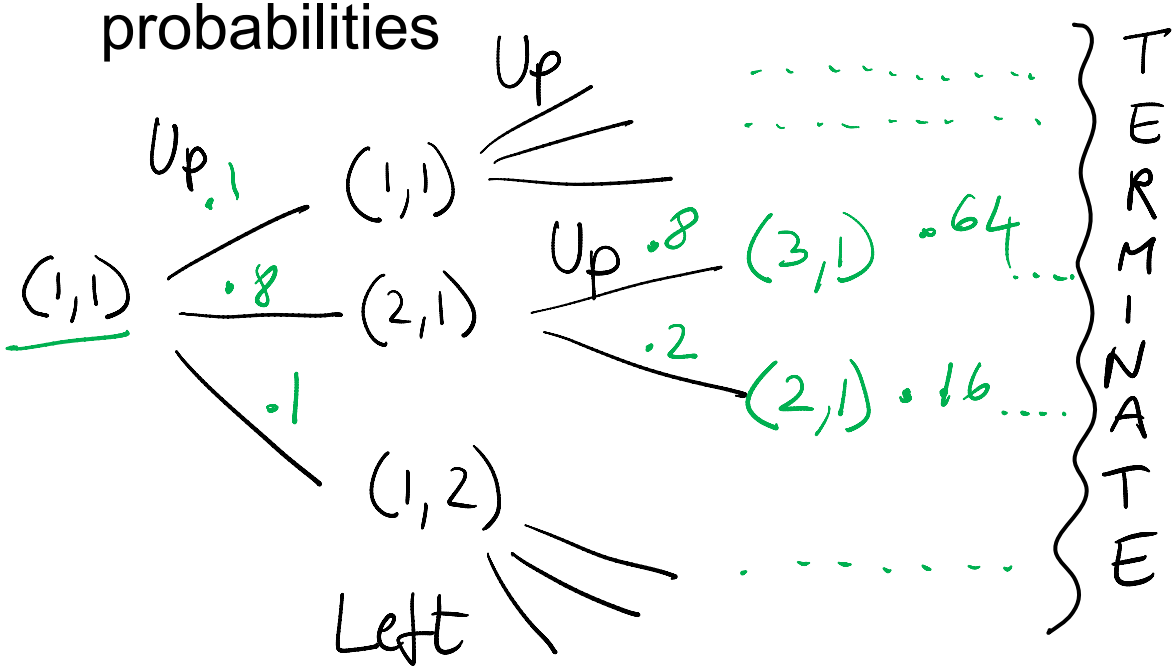
- Needs to make the same decision over and over: Given the current state what should I do?

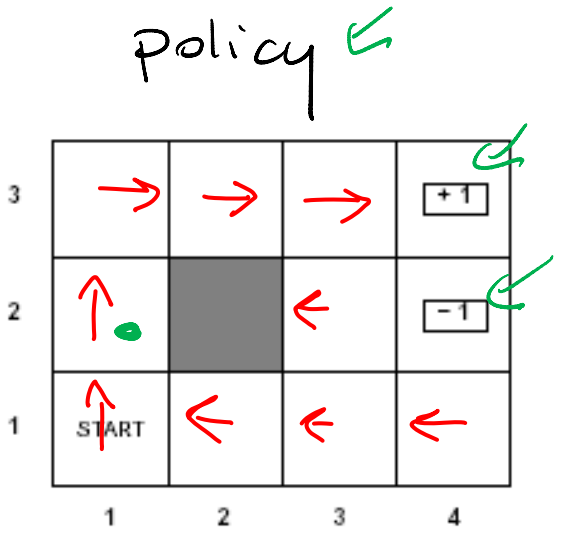
- So a policy for an MDP is a single decision function $\pi(s)$ that specifies what the agent should do for each state s



How to evaluate a policy

A policy can generate a set of state sequences with different probabilities





4^9 policies

Each state sequence has a corresponding reward. Typically the sum of the rewards for each state in the sequence

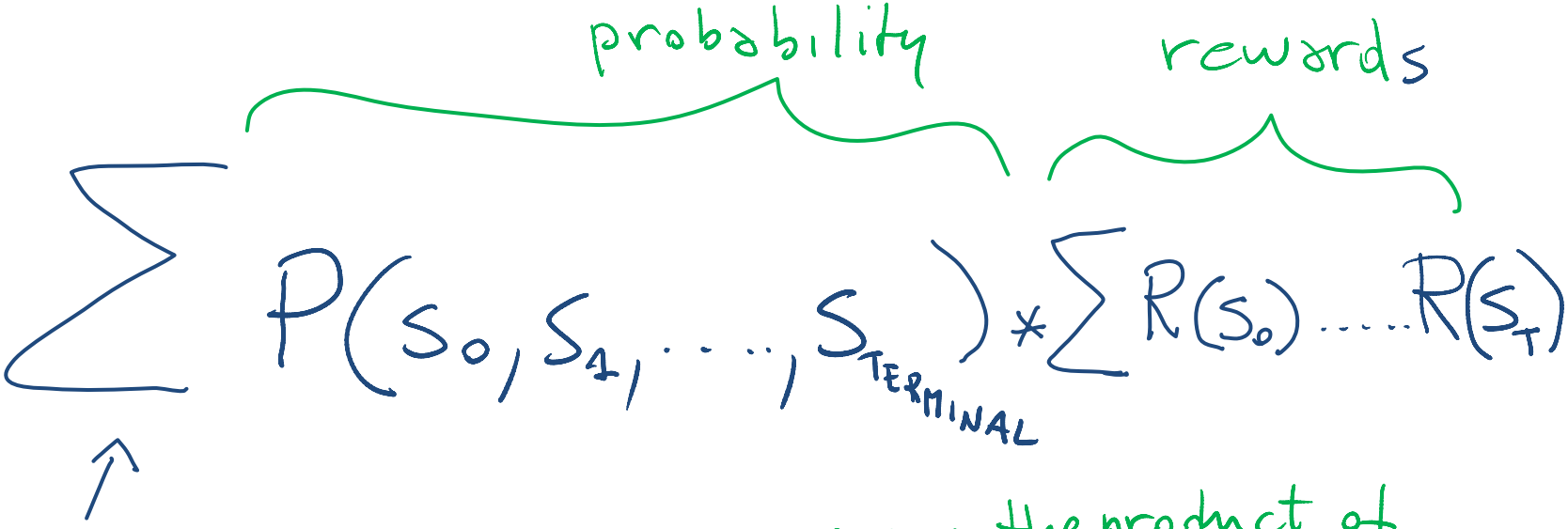
$\sum$ $(1,1) \xrightarrow{-.04} (1,1) \xrightarrow{-.04} (2,1) \rightarrow (3,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4)$ $+1$

$+.72$

MDPs: optimal policy

Optimal policy maximizes *expected total reward,* where

- Each environment history associated with that policy has a certain probability of accuriing and a given amount of total reward
- Total reward is a function of the rewards of its individual states



For all the sequences of states generated by the policy

we sum the product of its probability times its reward

Today Oct 20

Value of Information and value of Control

Markov Decision Processes

- Formal Specification and example
- Policies and Optimal Policy
- **Value Iteration**
- Rewards and Optimal Policy

Sketch of ideas to find the optimal policy for a MDP (Value Iteration)

We first need a couple of definitions

- $V^{\pi}(s)$: the expected value of following policy π in state s
- $Q^{\pi}(s, a)$, where a is an action: expected value of performing *a* in *s*, and then following policy π.

We have, by definition

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s'} P(s'|s,a) V^{\pi}(s')$$

reward obtained in *s*

Discount factor

states reachable from *s* by doing *a*

Probability of getting to *s'* from *s* via *a*

expected value of following policy π in *s'*

Value of a policy and Optimal policy

We can then compute $V^{\pi}(s)$ in terms of $Q^{\pi}(s, a)$

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$$

Expected value of following π in s

Expected value of performing the action indicated by π in s and following π after that

action indicated by π in s

For the optimal policy π^* we also have

$$V^{\pi^*}(s) = Q^{\pi^*}(s, \pi^*(s))$$

Value of Optimal policy

$$V^{\pi^*}(s) = Q^{\pi^*}(s, \pi^*(s))$$

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s'} P(s'|s,a) V^{\pi}(s')$$

Optimal policy π^* is one that gives the action that maximizes Q^{π^*} for each state

$$V^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) \times V^{\pi^*}(s'))$$

Value Iteration Rationale

- Given *N* states, we can write an equation like the one below for each of them

$$V(s_1) = R(s_1) + \gamma \max_a \sum_{s'} P(s' | s_1, a) V(s')$$

$$V(s_2) = R(s_2) + \gamma \max_a \sum_{s'} P(s' | s_2, a) V(s')$$

- Each equation contains *N* unknowns – the V values for the *N* states
- N equations in N variables (Bellman equations): It can be shown that they have a unique solution: the values for the optimal policy
- Unfortunately the N equations are non-linear, because of the max operator: Cannot be easily solved by using techniques from linear algebra
- Value Iteration Algorithm: Iterative approach to find the optimal policy and corresponding values

Value Iteration in Practice

- Let $V^{(i)}(s)$ be the utility of state s at the i^{th} iteration of the algorithm
- Start with arbitrary utilities on each state s: $V^{(0)}(s)$
- Repeat simultaneously for every s until there is "no change"

$$V^{(k+1)}(s) = R(s) + \gamma \max_a \sum_{s'} P(s' | s, a) V^{(k)}(s')$$

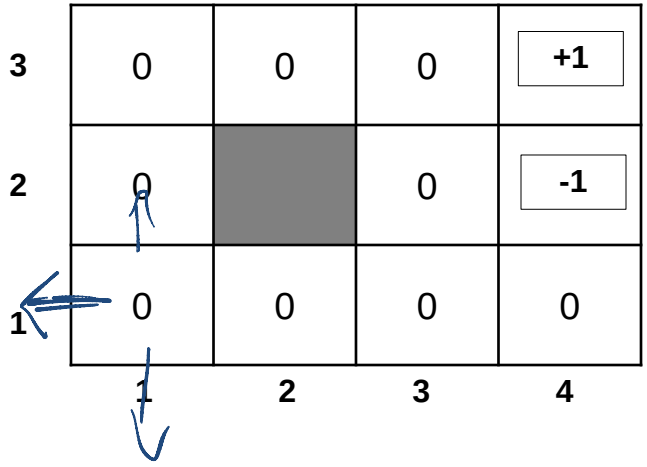
- True "no change" in the values of V(s) from one iteration to the next are guaranteed only if run for infinitely long.
  - In the limit, this process converges to a unique set of solutions for the Bellman equations
  - They are the total expected rewards (utilities) for the optimal policy

Example

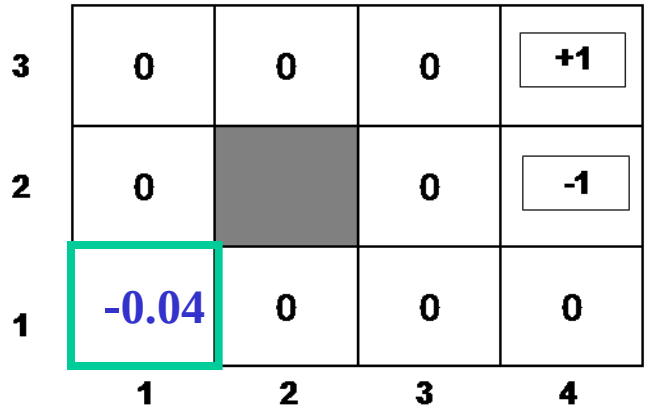
(sorry (column, row) to indicate state)

- Suppose, for instance, that we start with values $V^{(0)}(s)$ that are all 0

Iteration 0



Iteration 1



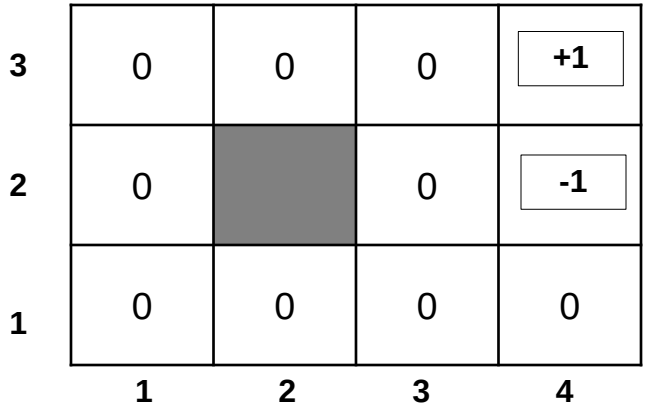
$$V^{(1)}(1,1) = -0.04 + 1 * \max \begin{bmatrix} 0.8V^{(0)}(1,2) + 0.1V^{(0)}(2,1) + 0.1V^{(0)}(1,1) & UP \\ 0.9V^{(0)}(1,1) + 0.1V^{(0)}(1,2) & LEFT \\ 0.9V^{(0)}(1,1) + 0.1V^{(0)}(2,1) & DOWN \\ 0.8V^{(0)}(2,1) + 0.1V^{(0)}(1,2) + 0.1V^{(0)}(1,1) & RIGHT \end{bmatrix}$$

$$V^{(1)}(1,1) = -0.04 + \max \begin{bmatrix} 0 & UP \\ 0 & LEFT \\ 0 & DOWN \\ 0 & RIGHT \end{bmatrix}$$

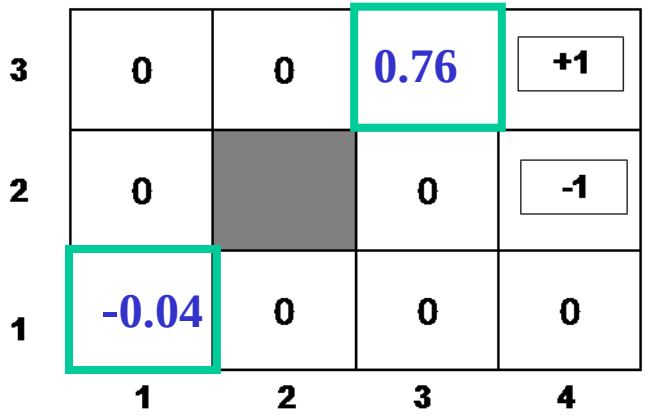
Example (cont'd)

- Let's compute $V^{(1)}(3,3)$

Iteration 0



Iteration 1



$$V^{(1)}(3,3) = -0.04 + 1 * \max \begin{bmatrix} 0.8V^{(0)}(3,3) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(4,3) & UP \\ 0.8V^{(0)}(2,3) + 0.1V^{(0)}(3,3) + 0.1V^{(0)}(3,2) & LEFT \\ 0.8V^{(0)}(3,2) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(4,3) & DOWN \\ 0.8V^{(0)}(4,3) + 0.1V^{(0)}(3,3) + 0.1V^{(0)}(3,2) & RIGHT \end{bmatrix}$$

$$V^{(1)}(3,3) = -0.04 + \max \begin{bmatrix} 0.1 & UP \\ 0 & LEFT \\ 0.1 & DOWN \\ 0.8 & RIGHT \end{bmatrix}$$

Example (cont'd)

- Let's compute $V^{(1)}(4,1)$

Iteration 0

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

Iteration 1

3	0	0	.76	+1
2	0		0	-1
1	-0.04	0	0	-0.04
	1	2	3	4

$$V^{(1)}(4,1) = -0.04 + \max \begin{bmatrix} 0.8V^{(0)}(4,2) + 0.1V^{(0)}(3,1) + 0.1V^{(0)}(4,1) & UP \\ 0.8V^{(0)}(3,1) + 0.1V^{(0)}(4,2) + 0.1V^{(0)}(4,1) & LEFT \\ 0.8V^{(0)}(4,1) + 0.1V^{(0)}(3,2) + 0.1V^{(0)}(4,1) & DOWN \\ 0.8V^{(0)}(4,1) + 0.1V^{(0)}(4,2) + 0.1V^{(0)}(4,1) & RIGHT \end{bmatrix}$$

$$V^{(1)}(4,1) = -0.04 + \max \begin{bmatrix} -0.8 & UP \\ -0.1 & LEFT \\ 0 & DOWN \\ -0.1 & RIGHT \end{bmatrix}$$

After a Full Iteration

Iteration 1

3	-.04	-.04	0.76	+1
2	-.04		-.04	-1
1	-.04	-.04	-.04	-.04
	1	2	3	4

- Only the state one step away from a positive reward (3,3) has gained value, all the others are losing value because of the cost of moving

Some steps in the second iteration

Iteration 1

3	-.04	-.04	0.76	+1
2	-.04		-.04	-1
1	-.04	-.04	-.04	-.04
	1	2	3	4

Iteration 2

3	-.04	-.04	0.76	+1
2	-.04		-.04	-1
1	-0.08	-.04	-.04	-.04
	1	2	3	4

$$V^{(2)}(1,1) = -0.04 + 1 * \max \begin{bmatrix} 0.8V^{(1)}(1,2) + 0.1V^{(1)}(2,1) + 0.1V^{(1)}(1,1) & UP \\ 0.9V^{(1)}(1,1) + 0.1V^{(1)}(1,2) & LEFT \\ 0.9V^{(1)}(1,1) + 0.1V^{(1)}(2,1) & DOWN \\ 0.8V^{(1)}(2,1) + 0.1V^{(1)}(1,2) + 0.1V^{(1)}(1,1) & RIGHT \end{bmatrix}$$

$$V^{(2)}(1,1) = -0.04 + \max \begin{bmatrix} -.04 & UP \\ -.04 & LEFT \\ -.04 & DOWN \\ -.04 & RIGHT \end{bmatrix} = -0.08$$

Example (cont'd)

- Let's compute $V^{(1)}(2,3)$

Iteration 1

3	-.04	-.04	0.76	+1
2	-.04		-.04	-1
1	-.04	-.04	-.04	-.04
	1	2	3	4

Iteration 2

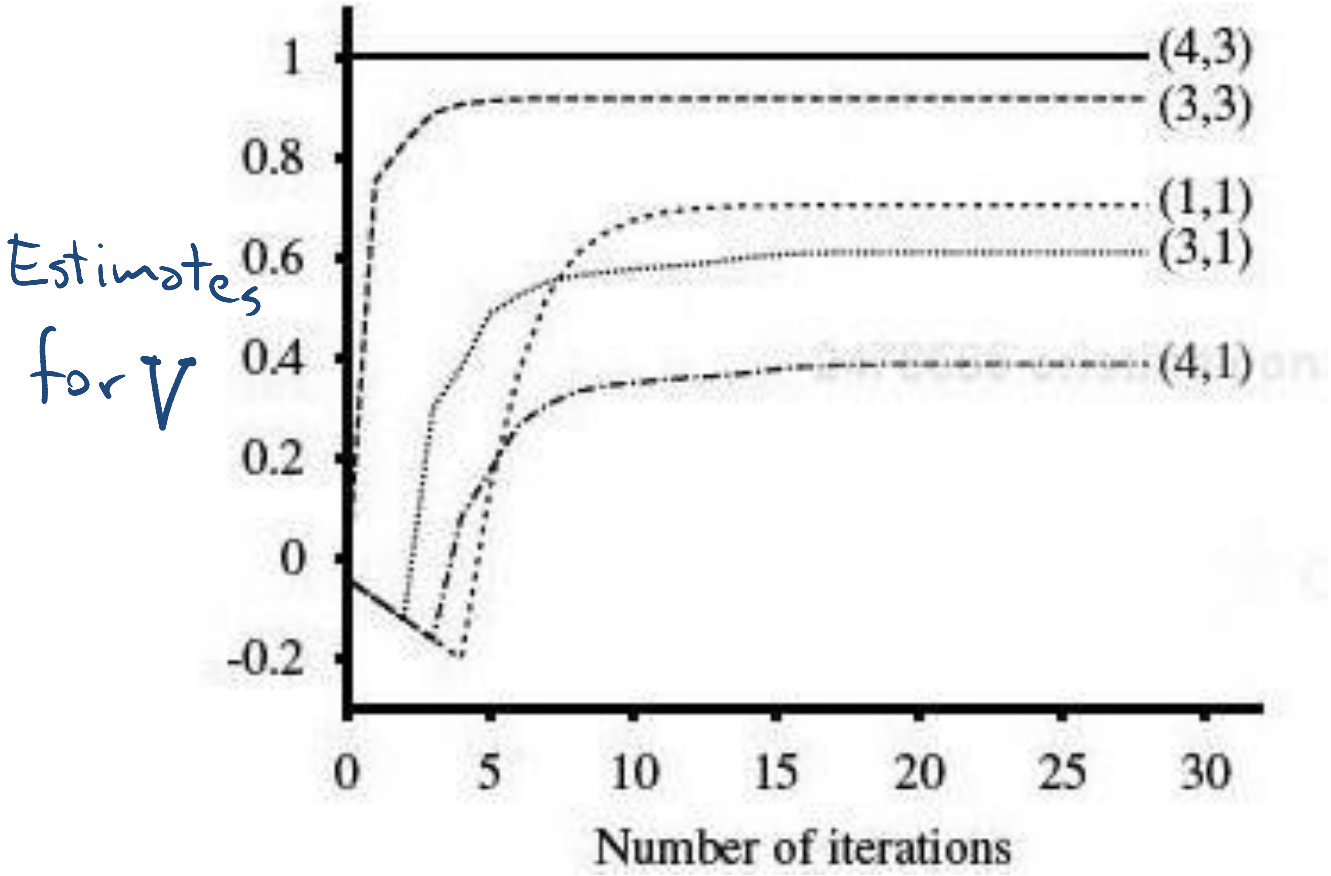
	-.04	0.56	0.76	+1
	-.04		-.04	-1
	-0.08	-.04	-.04	-.04
	1	2	3	4

$$V^{(1)}(2,3) = -0.04 + 1 * \max \begin{bmatrix} 0.8V^{(0)}(2,3) + 0.1V^{(0)}(1,3) + 0.1V^{(0)}(3,3) & UP \\ 0.8V^{(0)}(1,3) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(2,3) & LEFT \\ 0.8V^{(0)}(2,3) + 0.1V^{(0)}(1,3) + 0.1V^{(0)}(3,3) & DOWN \\ 0.8V^{(0)}(3,3) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(2,3) & RIGHT \end{bmatrix}$$

$$V^{(1)}(2,3) = -0.04 + (0.8 * 0.76 + 0.2 * -0.04) = 0.56$$

- Steps two moves away from positive rewards start increasing their value

State Utilities as Function of Iteration



- Note that values of states at different distances from (4,3) accumulate negative rewards until a path to (4,3) is found

Value Iteration: Computational Complexity

Value iteration works by producing successive approximations of the optimal value function.

- Each iteration can be performed in $O(|A||S|^2)$ steps,
- or faster if there is sparsity in the transition function.

Today Oct 20

Value of Information and value of Control

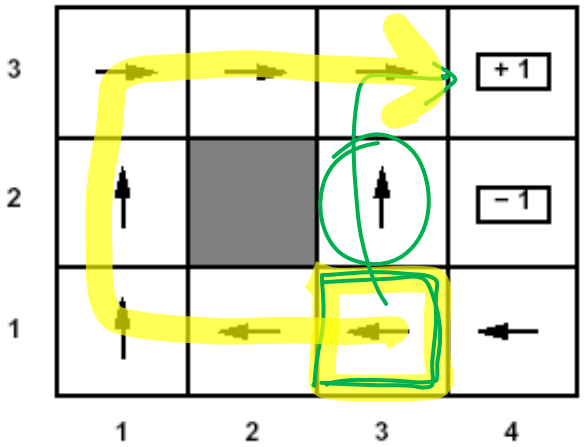
Markov Decision Processes

- Formal Specification and example
- Policies and Optimal Policy
- Value Iteration
- **Rewards and Optimal Policy**

Rewards and Optimal Policy

Optimal Policy when penalty in non-terminal states is -0.04

computed by Value Iteration



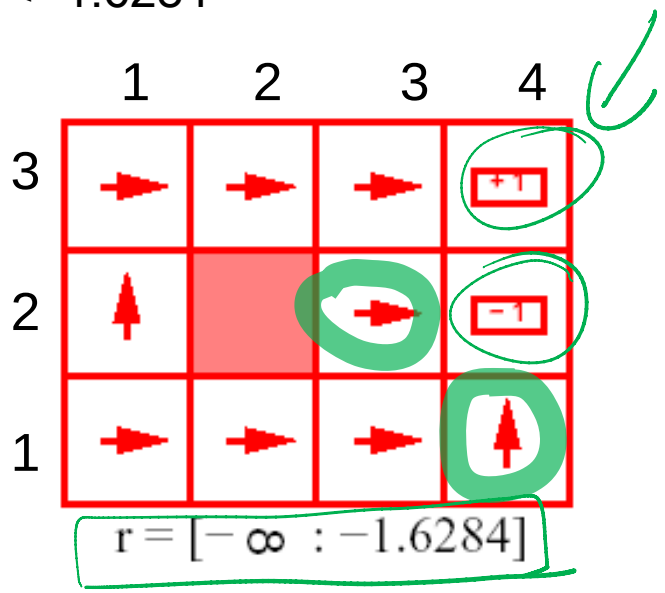
Note that here the cost of taking steps is small compared to the cost of ending into (2,4)

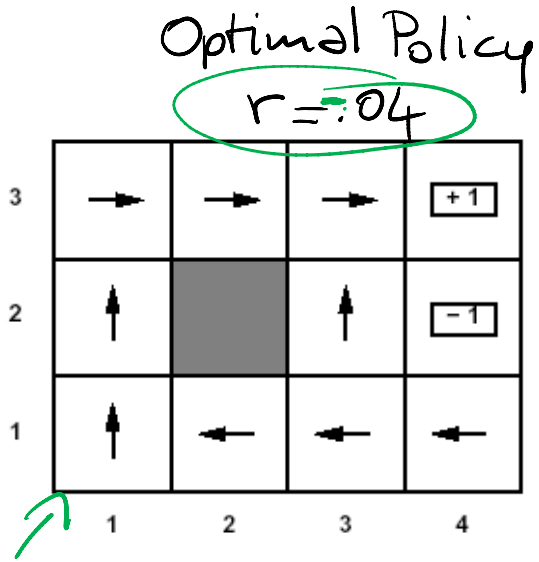
- Thus, the optimal policy for state (1,3) is to take the long way around the obstacle rather then risking to fall into (2,4) by taking the shorter way that passes next to it

May the optimal policy change if the reward in the non-terminal states (let's call it *r*) changes?

Rewards and Optimal Policy

Optimal Policy when r < -1.6284

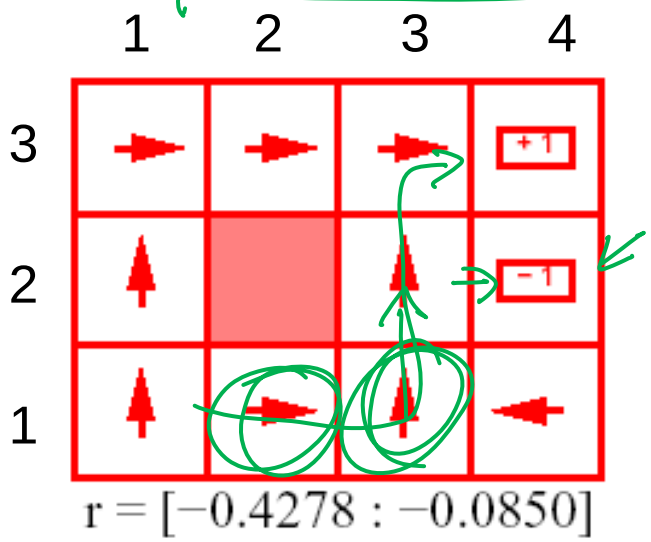




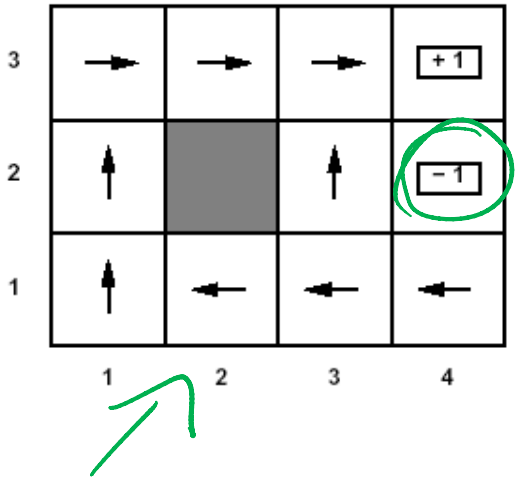
Why is the agent heading straight into (2,4) from its surrounding states?

Rewards and Optimal Policy

Optimal Policy when -0.427 < r < -0.085



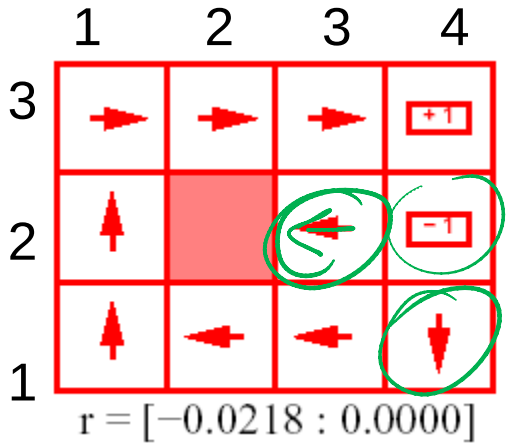
Optimal Policy
r = -.04



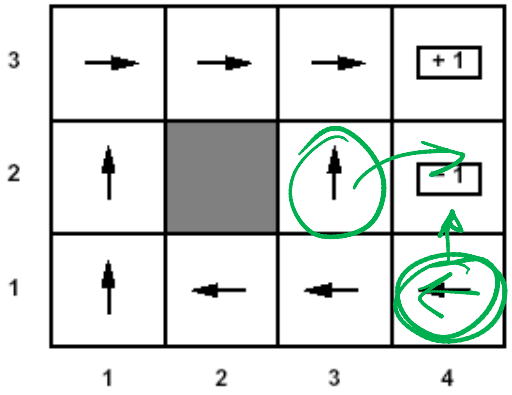
The cost of taking a step is high enough to make the agent take the shortcut to (3,4) from (1,3)

Rewards and Optimal Policy

Optimal Policy when -0.0218 < r < 0



Optimal Policy
r = -.04

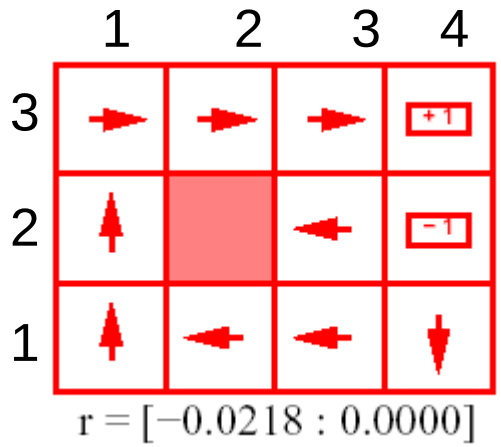


Why is the agent heading straight into the obstacle from (2,3)? And into the wall in (1,4)?

see next slide .....

Rewards and Optimal Policy

Optimal Policy when -0.0218 < r < 0



Stay longer in the grid is not penalized as much as before. The agent is willing to take longer routes to avoid (2,4)

- This is true even when it means banging against the obstacle a few times when moving from (2,3)

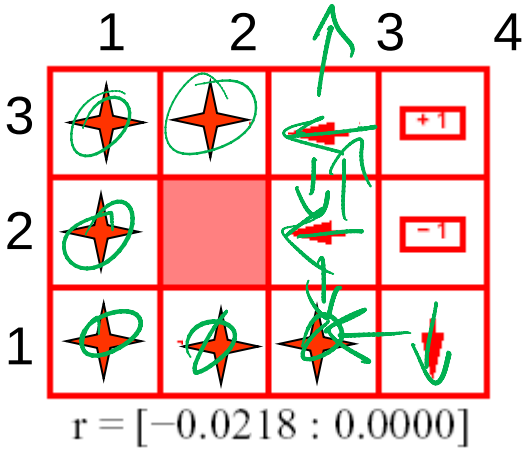
Rewards and Optimal Policy

Optimal Policy when r > 0

Which means the agent is rewarded for every step it takes

it avoids terminal states completely

state where every action belong to an optimal policy



r = [−0.0218 : 0.0000]

AI talk today: Lots of concepts covered in 502

Speaker: Thomas G. Dietterich, Professor Oregon State University
http://web.engr.oregonstate.edu/~tgd/

Title: Challenges for Machine Learning in Ecological Science and Ecosystem Management
Time: 3:30 - 4:50 p.m
Location: Hugh Dempster Pavilion (DMP)
Room 110, 6245 Agronomy Rd.
Abstract:
Just as machine learning has played a huge role in genomics, there are many problems in ecological science and ecosystem management that could be transformed by machine learning. ….. These include (a) .., (b) automated classification of images of arthropod specimens, (c) species distribution modeling …. (d) design of optimal policies for managing wildfires and invasive species. …. combining probabilistic graphical models with non-parametric learning methods, and optimization of complex spatio-temporal Markov processes.

TODO for next Tue

- **Read Textbook 9.5**

- Also Do exercises 9.C
http://www.aispace.org/exercises.shtml