

CS480 – INTRODUCTION TO ARTIFICIAL INTELLIGENCE

TOPIC: COMPLEX DECISIONS
CHAPTER: 17



Mustafa Bilgic



<http://www.cs.iit.edu/~mbilgic>



<https://twitter.com/bilgicm>

COMPARISON WITH CH16

○ Similar

- The outcomes of each action are stochastic
- The agent is trying to maximize expected utility

○ Different

- Instead of an episodic decision (chapter 16), the agent's utility depends on a sequence of actions (chapter 17)

WE

- Will cover

- 17.1 – Intro to sequential decision making
- 17.2 – Value iteration
- 17.3 – Policy iteration

- Will not cover

- 17.4 – Partially-observable environments
- 17.5 – Multiple agents
- 17.6 – Mechanism design

EXAMPLE

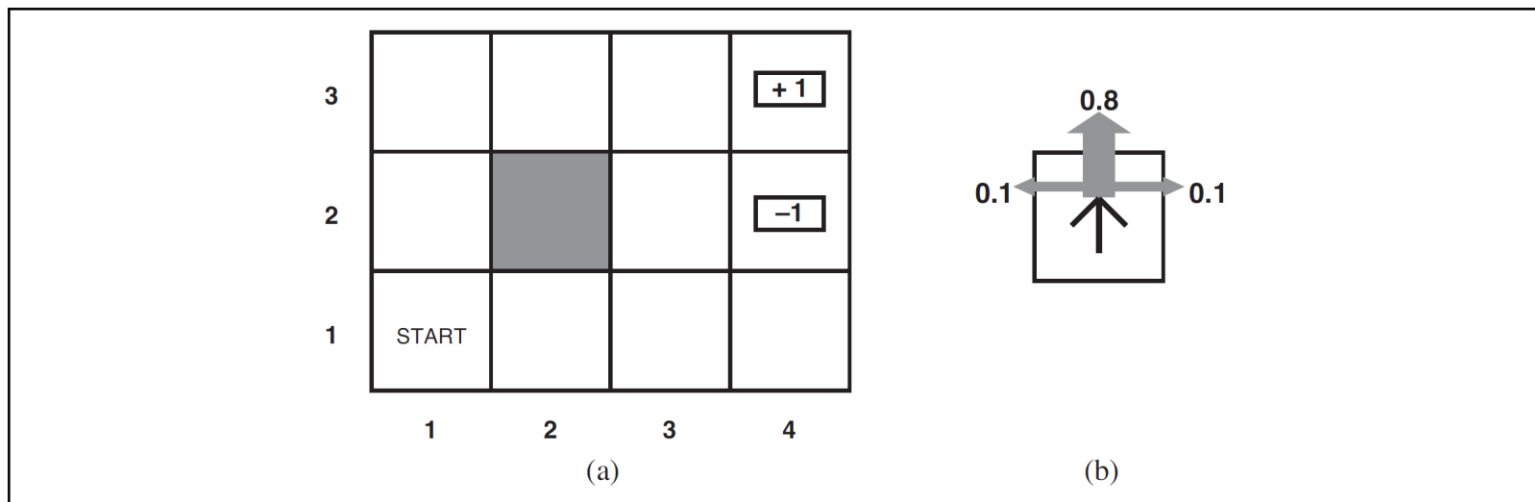


Figure 17.1 FILES: figures/sequential-decision-world.eps (Tue Nov 3 16:23:43 2009). (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the “intended” outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and -1, respectively, and all other states have a reward of -0.04.

REPRESENTATION

- **Fully observable:** the agent knows where it is
- **Stochastic:** the agent's actions have probabilistic outcomes
 - $P(s' | s, a)$ Probability of arriving at state s' given we are at state s and take action a
- Transition model is **Markovian**
 - $P(s' | s, a)$ depends on only a and s and not the entire history of actions and states
- **Sequential:** the utility depends on the sequence of states
- Utility is **additive:** the utility is the sum (with a potential discounting factor) of the reward, $R(s)$, received at each state s that the agent visits

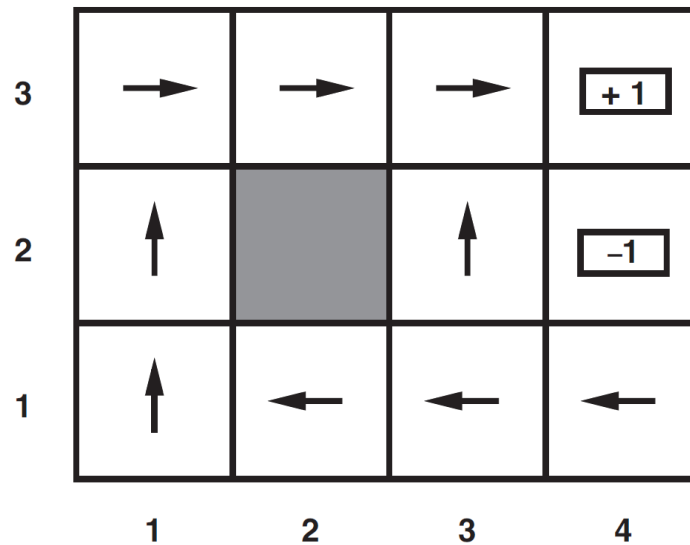
MARKOV DECISION PROCESS

- “*A sequential decision problem for a fully-observable stochastic environment with a Markov transition model and additive rewards is called a **Markov Decision Process (MDP)***” – Textbook page 647

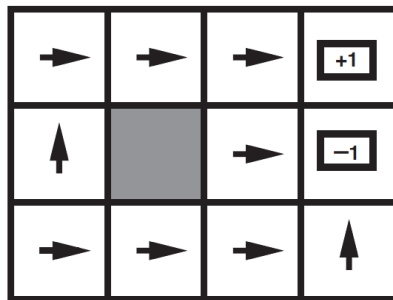
SOLUTION?

- A fixed action sequence is not the answer due to stochasticity
 - For example, [Up, Up, Right, Right, Right] is not a solution
 - It would be a solution if the environment was deterministic
- A solution must specify the agent should do in any state that the agent might reach
 - This is called a **policy**
- Policy notation: π
 - $\pi(s)$ specifies what action the agent should take at state s
- An **optimal policy** is the one that maximizes the expected utility
 - π^*

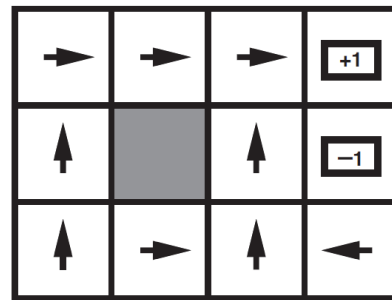
SOLUTION TO EARLIER EXAMPLE



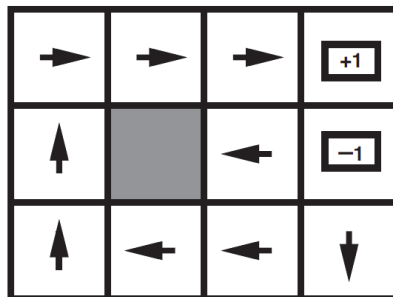
SOLUTIONS FOR DIFFERENT $R(s)$



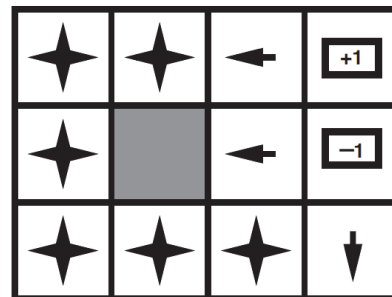
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



$$-0.0221 < R(s) < 0$$



$$R(s) > 0$$

UTILITIES OVER TIME

- How should we calculate $U([s_0, s_1, s_2, \dots])$?
- Additive rewards
 - $U([s_0, s_1, s_2, \dots]) = R(s_0) + R(s_1) + R(s_2) + \dots$
- Discounted rewards
 - $U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$
 - The discounting factor γ is a number between 0 and 1
 - The agent prefers current rewards to future rewards
 - When γ is close to 0, distant future is insignificant
 - When $\gamma = 1$, it is equivalent to additive rewards

UTILITY OF STATES

- The agent receives a reward at each state
- Utility of a state s given a policy π is the expected reward that the agent will get starting from state s and following policy π
- Let S_t denote the state that the agent reaches at time t
- $U^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t)]$
- The expectation is with respect to the transition probabilities

THE OPTIMAL POLICY

- The optimal policy is the one that maximizes the expected utility
 - $\pi_s^* = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$
- Remember that π_s^* is a policy; that is, it recommends an action for each state, regardless of whether it is the starting state or not
- It is optimal when the starting state is s
- When the rewards are discounted, the optimal policy is independent of the start state
 - Wait, what?
- The optimality of the policy does not depend on the starting state but of course the action sequence depends on the starting state
- True utility of each state is defined as $U^{\pi^*}(s)$ -- the expected rewards the agent will receive if it executes the optimal policy starting at s

$U(s)$ vs $R(s)$

- $R(s)$ is the short-term immediate reward at s
- $U(s)$ is the long-term cumulative reward from s and onward

U(s) FOR THE EXAMPLE

3	0.812	0.868	0.918	+ 1
2	0.762		0.660	- 1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Figure 17.3 FILES: figures/sequential-decision-values.eps (Tue Nov 3 16:23:42 2009). The utilities of the states in the 4×3 world, calculated with $\gamma = 1$ and $R(s) = -0.04$ for nonterminal states.

IF WE WERE GIVEN $U(s)$

- $\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$
- However, we are not given $U(s)$
- Two algorithms for finding optimal policies
 1. Value iteration
 2. Policy iteration

VALUE ITERATION

- $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$
- This is called the **Bellman equation**
- Exercise: write the Bellman equation for $U(x,y)$ for our example
- n possible states, n Bellman equations, one for each state
- However, these are non-linear equations, due to the max operator
- One approach: iterative
 - Start with an initial guess (could be random)
 - Iterate until convergence

VALUE ITERATION FUNCTION

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function

inputs: mdp , an MDP with states S , actions $A(s)$, transition model $P(s' | s, a)$,
rewards $R(s)$, discount γ

ϵ , the maximum error allowed in the utility of any state

local variables: U, U' , vectors of utilities for states in S , initially zero

δ , the maximum change in the utility of any state in an iteration

repeat

$U \leftarrow U'; \delta \leftarrow 0$

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

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

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

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

return U

OPTIMAL POLICY

- Once the value iteration algorithm terminates, use $\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$ to determine the optimal policy, i.e., the optimal action for each s

SKIPPED DETAILS

- Is the value iteration algorithm guaranteed to converge to the true values, and under what conditions
- How long does it take to converge?
- Do we need the algorithm fully converge to find the optimal policy or is an approximate computation of the utilities often enough?

POLICY ITERATION

- Start with an initial policy π_0
- Alternate between
 1. Policy iteration: given policy π_i , calculate U^{π_i}
 2. Policy improvement: Calculate a new MEU policy π_{i+1} , using the utilities calculated in the previous step
- Stop when utilities no longer change

SUMMARY – IMPORTANT EQUATIONS

- $U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$
 - Discounted rewards
- $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$
 - Bellman equation; utility under optimal policy
- $\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$
 - Optimal policy when utilities are given
- $U^{\pi_i}(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi_i(s)) U^{\pi_i}(s')$
 - Utility under fixed policy π_i

EXERCISE

- Define a simple MDP
- Solve it manually
- Solve it using Python
 - See examples at <https://github.com/aimacode/aima-python/blob/master/mdp.ipynb>

NEXT

- Part V – Learning