CS 581 – Advanced Artificial Intelligence

Topic: Decision Making Under Uncertainty

Mustafa Bilgic

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

https://twitter.com/bilgicm

UNCERTAINTY

- The agent needs reason in an uncertain world
- Uncertainty can be due to
  - Noisy sensors (e.g., temperature, GPS, camera, etc.)
  - Imperfect data (e.g., low resolution image)
  - Missing data (e.g., lab tests)
  - Imperfect knowledge (e.g., medical diagnosis)
  - Exceptions (e.g., all birds fly except ostriches, penguins, birds with injured wings, dead birds, …)
  - Changing data (e.g., flu seasons, traffic conditions during rush hour, etc.)
  - …
- The agent still must act (e.g., step on the breaks, diagnose a patient, order a lab test, …)

A Few Examples

- Spam filtering
- Medical diagnosis
- Loan approval
- Automated driving
- …

Rational Agent

- Given world states, a utility function, actions, transitions, evidence, and probabilities,
- A rational agent chooses the action that maximizes expected utility

$$action = \underset{a}{\operatorname{argmax}}\, EU(a|\mathrm{e})$$

UTILITY THEORY

- Lottery: n possible outcomes with probabilities
  - $[p_1, S_1; p_2, S_2; \ldots\ p_n, S_n]$
  - Each S_i can be an atomic state or another lottery
- **Expected utility of a lottery**
  - $EU([p_1, S_1;\ p_2, S_2;\ \ldots\ pn, Sn]) = \sum_{i=1}^{n} p_i U(S_i)$

Utility ≠ Money

- Most agents prefer more money to less money,
  - But this does not mean money behaves as a utility function
- For example, which lottery would you prefer
  - L_1: [1, \$1 Million]
  - L_2: [0.5, \$0; 0.5, \$2.5 Million]
- If money served as a utility function, then you'd prefer L_2 no matter what, but the answer *often* depends on how much money you currently have
  - The utility of money depends on what you *<u>prefer</u>*
    - If you are short on cash, a little more *certain* money can help
    - If you are already billionaire, you might take the risk
    - Or if you are swimming in debt, you might like to gamble

UTILITY ≠ MONEY

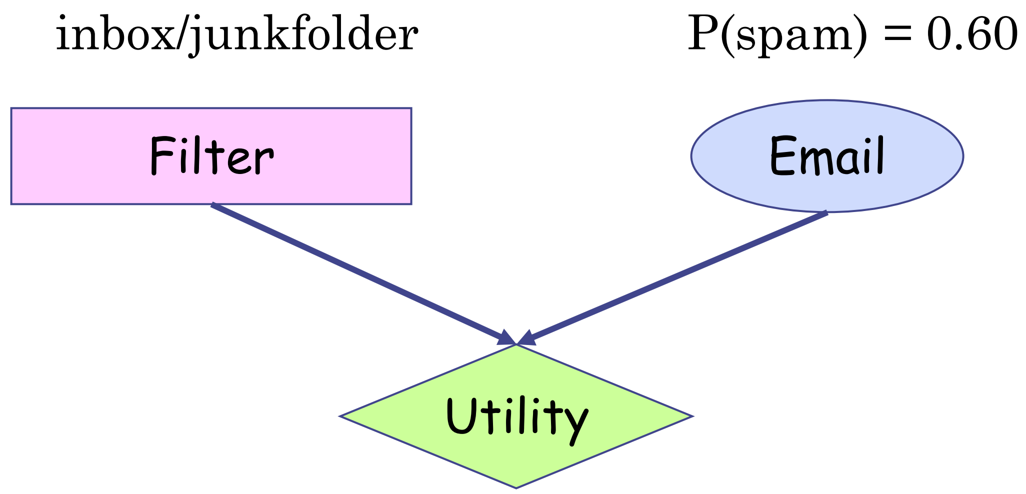
- Let's say you currently have \$k and let S_k represent the state of having \$k
- $EU(L_1) = U(S_{k+1M})$
- $EU(L_2) = 0.5*U(S_k) + 0.5*U(S_{k+2.5M})$
- The rational choice depends on your preferences for S_k, S_{k+1M}, and $S_{k+2.5M}$
  - i.e, it depends on the values of $U(S_k)$, $U(S_{k+1M})$, and $U(S_{k+2.5M})$
- $U(S_i)$ does not have to be a linear function of i, and for people it often is not

Influence Diagrams

Influence Diagrams / Decision Networks

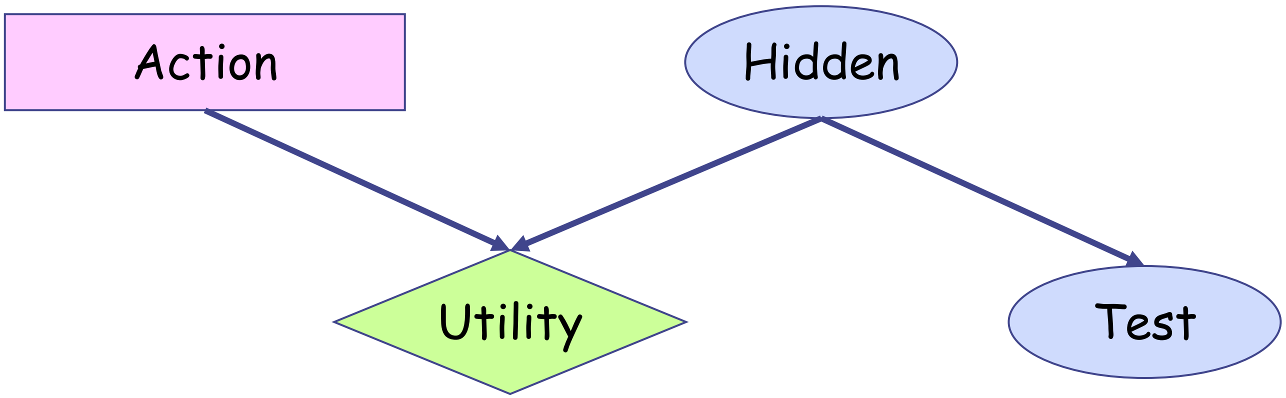
- Builds on Bayesian networks
- In addition to the chance nodes (ovals), decision networks have
  - Decision nodes – square
    - Represents actions
  - Utility nodes – diamond
    - Represents utilities for possible states and actions

Spam Filtering



U(inbox, spam) = 0
U(inbox, ~spam) = 100
U(junkfolder, spam) = 200
U(junkfolder, ~spam) = -500

A Common Pattern



Decision networks - applications

- Used for
  - What action to take
  - What information to gather
  - How much to pay for a piece of information
- For example:
  - Medical diagnosis: which test to perform, which treatment to prescribe, …
  - Marketing: which project to invest in, how much to spend on marketing, how much to spend on user surveys, …

Evaluating decision networks

- Set evidence nodes **E** to their values **e**
- For each choice **a** of action **A**
  - Set **A=a**
  - Compute the posterior probability of the parent chance nodes of the utility node; i.e., compute P(Pa(Utility) | **e, a**)
  - Compute expected utility using the utility node and the probability distribution P(Pa(Utility) | **e, a**)
- Choose action **a** with the maximum expected utility

EXAMPLES

- See OneNote

Value of Information

- If I am allowed to observe the value of a chance node, how much valuable is that information to me?
- Value of information
  - Expected utility after the information is acquired
    - Minus
  - Expected utility before the information is acquired
- There is one catch: we do not know the content of the information before we acquire it
  - Solution: take an expectation over the possible outcomes

Markov Decision Processes

Problem Setting

- An agent in a stochastic environment, looking for a sequence of actions that will maximize its expected utility
- *Reading: Chapter 17 of the AI book by Stuart and Russel*

Running Example

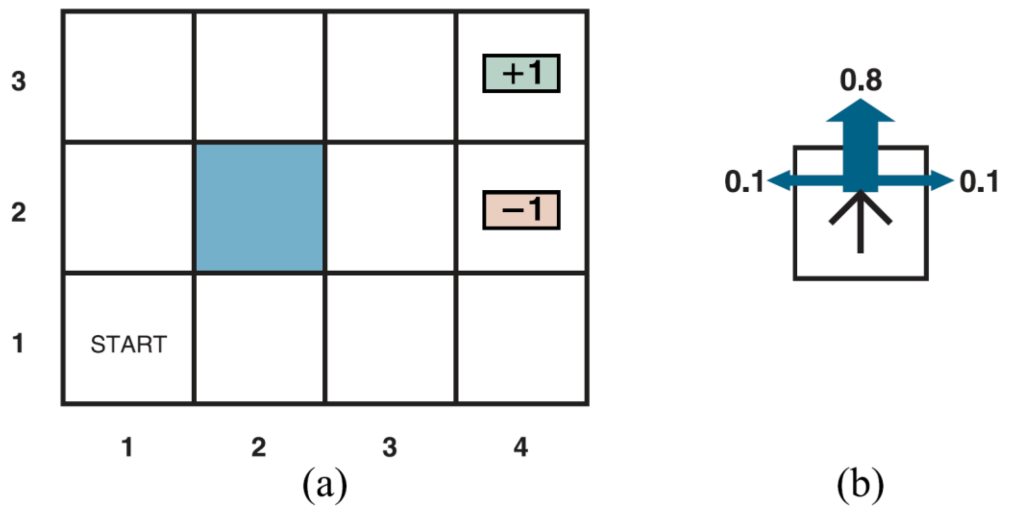


Figure 17.1 (a) A simple, stochastic 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and –1, respectively, and all other transitions have a reward of –0.04.

Figure from http://aima.cs.berkeley.edu/figures.pdf

NOTATION

- $P(s' \mid s, a)$ Probability of arriving at state s' given we are at state s and take action a
- $R(s, a, s')$ The reward the agent receives when it transitions from state s to state s' via action a
- $\pi(s)$ The action recommended by policy π at state s
- π^* Optimal policy
- $U^\pi(s)$ The expected utility obtained via executing policy π starting at state s
- $U^{\pi^*}(s)$ is abbreviated as $U(s)$
- $Q(s, a)$ The expected utility of taking action a at state s
- γ Discount factor

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)***" – AI textbook by Russel and Norvig

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: $\boldsymbol{\pi}$
  - $\pi(s)$ specifies what action the agent should take at state s
- An **optimal policy** is the one that maximizes the expected utility
  - $\boldsymbol{\pi^*}$

Running Example

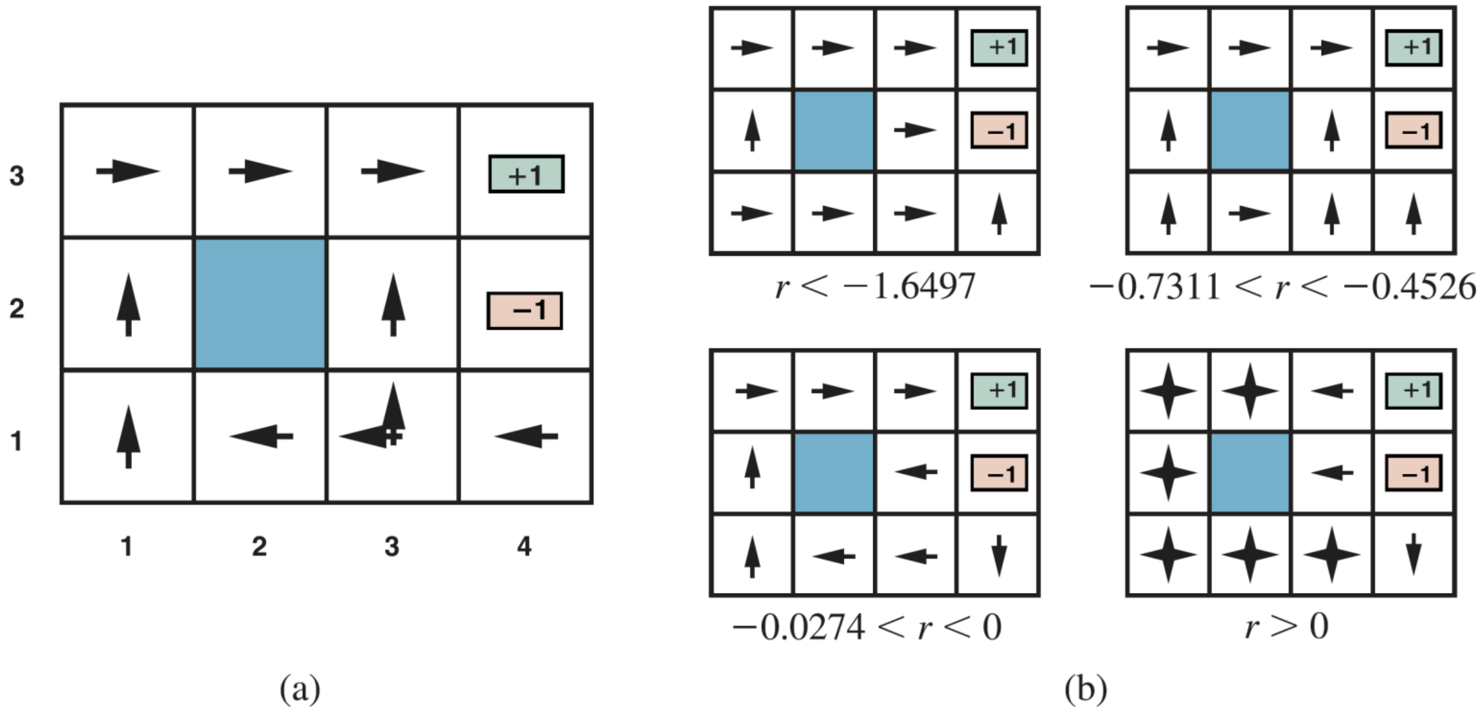


Figure 17.2 (a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. (b) Optimal policies for four different ranges of r.

Figure from http://aima.cs.berkeley.edu/figures.pdf

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 taking actions according to policy π
- Let S_t denote the state that the agent reaches at time t
- $U^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})]$
- 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
  - 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, a, s')

- *R(s, a, s')* is the short-term immediate reward the agent receives when it transitions from state *s* to state *s'* via action *a*
- *U(s)* is the long-term cumulative reward from *s* onward
- $U^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})]$

BELLMAN EQUATION

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

Action-Utility Function $Q(s, a)$

- $Q(s, a)$ The expected utility of taking action a at state s
- $Q(s, a) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]$
- $U(s) = \max_{a \in A(s)} Q(s, a)$
- $Q(s, a) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \max_{a' \in A(s')} Q(s', a')]$
  - Bellman equation for the Q function

Running Example

	1	2	3	4
3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	–1
1	0.7453	0.6953	0.6514	0.4279

Figure 17.3 The utilities of the states in the 4×3 world with $\gamma = 1$ and $r = -0.04$ for transitions to nonterminal states.

Figure from http://aima.cs.berkeley.edu/figures.pdf

EXERCISE

- Confirm that the utilities given in the previous slide satisfy the Bellman equations

How to Find π^*

- $\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]$
- However, we are not given $U(s)$
- Two algorithms for finding optimal policies

1. Value iteration
2. Policy iteration

Value Iteration

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

Policy Iteration

- Start with an initial policy π_0
- Alternate between

1. Policy evaluation: 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

Multi-armed Bandit

Setup

- K slot machines
  - Each one is a one-armed bandit
- Unknown reward functions
  - The distribution of rewards for each machine are unknown
- Limited resources
  - Have a limited number of tries (or alternatively, future rewards are discounted)
- Can gather information
  - Each try gives a (potentially) zero reward, but also is useful for information gathering purposes
- Main objective
  - Maximize rewards
- Exploration vs exploitation trade-off
  - How should you balance exploitation (sticking with a machine that looks good) versus exploration (trying new machines)?

SOME STRATEGIES

1. Fully random (explore only)
2. Epsilon-greedy: With ϵ probability, choose a random lever; otherwise, choose the best lever so far
3. Epsilon-first: Explore at random the first $\epsilon \times M$ tries; exploit the best after that
4. Upper-confidence bound (UCB): next slide

- Evaluation: regret (ρ): the difference to the optimal strategy at time T
  - $\rho = T \times \mu^* - \sum_{t=1}^{T} r_t$
- A zero-regret strategy: a strategy where ρ/T goes to zero in the limit

UCB1

- Assume K machines; each with a binary reward of 0 or 1; the probability of reward 1 is p_i
  - Remember that the user does not know p_i
- If you knew p_i, what is the mean reward for machine i?
- The upper-confidence bound (UCB) method calculates the upper bound on the mean for each slot and chooses the machine with max value
  - $\overline{r_i} + \sqrt{\frac{2\ln(n)}{n_i}}$, where $\overline{r_i}$ is the average reward for the i^{th} machine, n is the total number of tries so far, and n_i is the number of times the machine i is tried so far
- The upper confidence grows with n, shrinks with n_i; in essence, UCB balances between exploration and exploitation
- The upper confidence is derived using the Chernoff-Hoeffding bound

Variations

- The reward distributions can be more complicated than simple random distributions
- Contextual bandit
  - The state and the rewards depend on the context
- Adversarial bandit
  - At each iteration, the agent chooses an arm while at the same time an adversary chooses the reward functions

NEXT

- Reinforcement learning
  - In fact, we already covered many of the fundamentals of RL
    - Value iteration, policy iteration, exploration vs exploitation trade-off
  - We are now ready to make the leap from MDPs to RL
  - RL can be considered as solving an MDP where the transition and reward dynamics are unknown