CS 581 – Advanced Artificial Intelligence

Topic: Decision Making Under Uncertainty

Mustafa Bilgic

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

Motivation

- 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, …)

What to do?

- Probability of a given email being spam is 0.6. Where should that email be delivered?
- Probability of rain is 0.6. I don't like carrying my umbrella around, but I also don't like getting wet. Should I take my umbrella?
- Probability of a given patient suffering from a heart disease is 0.6. Should more tests be conducted and if so, which ones and in which order? What should the treatment plan be?
- Probability of the product selling is 0.6. If we spend $100K in ads, the probability goes up to 0.7. Is $100K on ads justified?

Rational Agent

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

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

Uncertainty Representation

- Covered
  - Probability background (marginal, conditional, independence, Bayes rule, …)
  - Probabilistic classification (naïve Bayes, logistic regression, neural networks)
- Skipped
  - Bayesian networks (covered in CS 480 and CS 583)

UTILITY THEORY

- Lottery: n possible outcomes with probabilities
  - $[p_1, \mathrm{S}_1; p_2, \mathrm{S}_2; \ldots\ p_n, \mathrm{S}_n]$
  - 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 Theory

- Lottery: n possible outcomes with probabilities
  - $[p_1, \mathrm{S}_1; p_2, \mathrm{S}_2; \ldots\ p_n, \mathrm{S}_n]$ $\Sigma p_i = 1$
  - Each S_i can be an atomic state or another lottery
- **Expected utility of a lottery**
  - $EU([p_1, S_1;\ p_2, S_2;\ \ldots\ pn, Sn]) = \sum_{i=1}^{n} p_i U(S_i)$

$U(S_1) = 100$

$u(S_2) = 50$

$[0.6\ S_1;\ 0.4\ S_2]$

$0.6 \times 100 + 0.4 \times 50 = 60 + 20$
$= 80$

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 *prefer*
    - 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

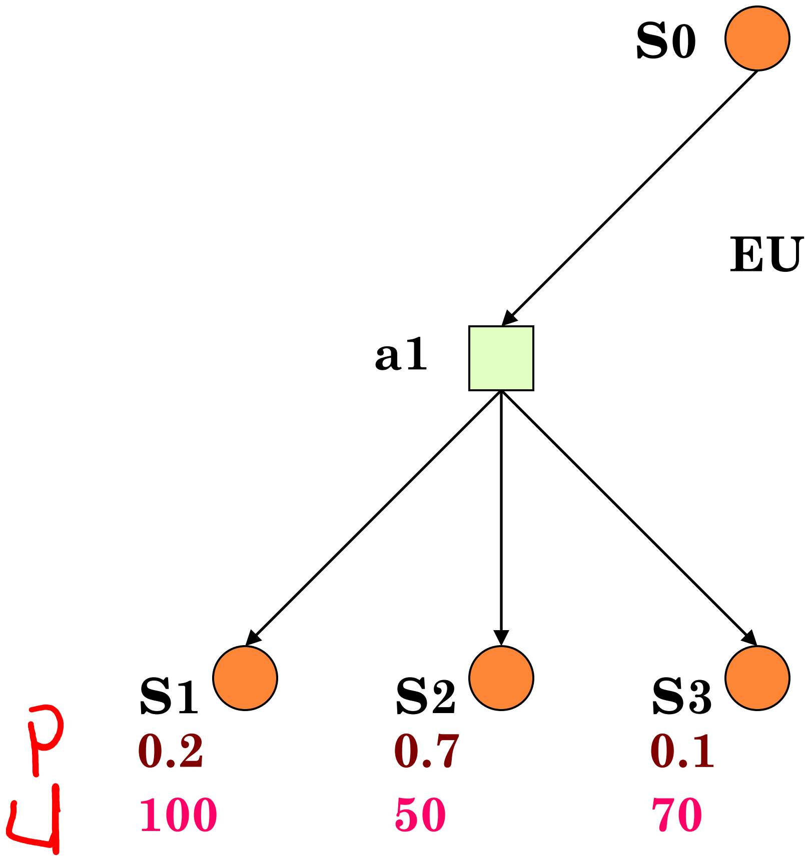
Utility ≠ Money

- 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

Which Action To Take?

- Given
  - A probability distribution
  - Choice of actions and their effects on the distribution
  - Evidence
  - A utility function (as a function of states and possibly actions)
- Find
  - Which action maximizes the expected utility?

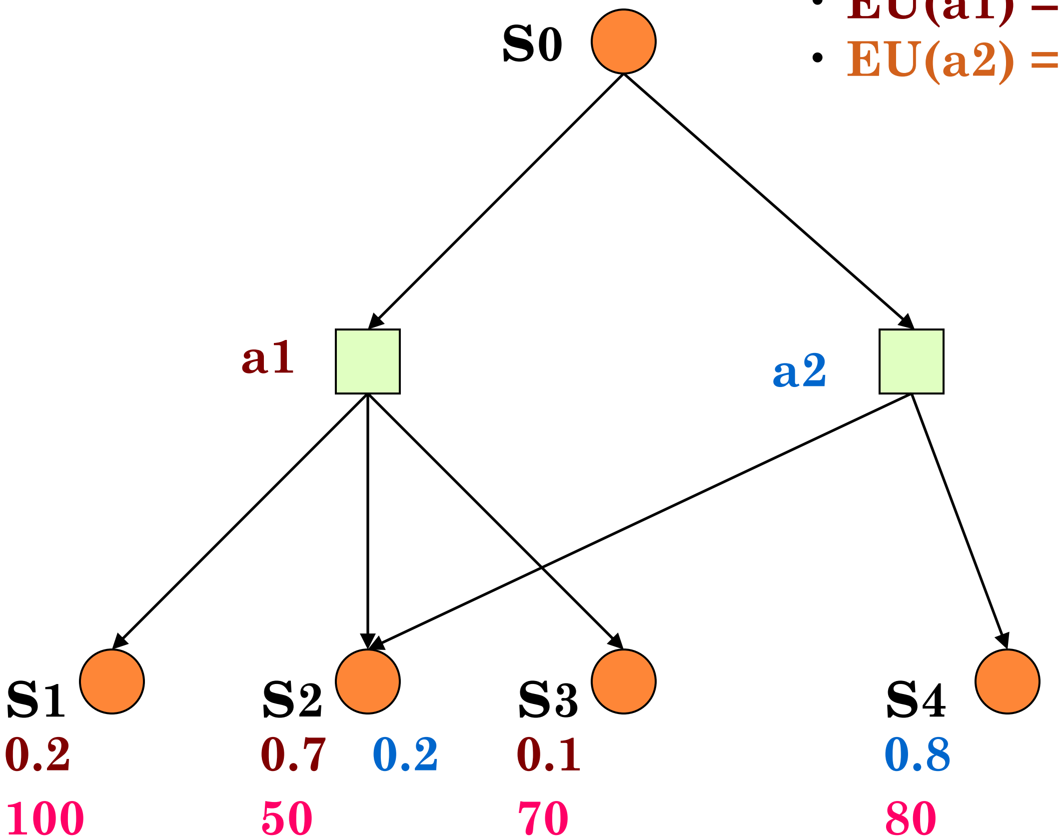
One Action Example



EU(a1) = 100 x 0.2 + 50 x 0.7 + 70 x 0.1
= 20 + 35 + 7
= 62

Two Actions Example

- EU(a1) = 62
- EU(a2) = 0.2*50+0.8*80 = 74



A simple Spam Filtering Example

- Given an email:
  - P(s | e) = 0.6
  - P(~s | e) = 0.4
- Actions:
  - deliverIntoSpamFolder (dS)
  - deliverIntoInboxFolder (dI)
- Utility function:
  - U(dS, s) = 200
  - U(dI, ~s) = 100
  - U(dI, s) = -100
  - U(dS, ~s) = -500
- Where should this email be delivered and why?

A SIMPLE SPAM FILTERING EXAMPLE

- Given an email:
  - P(s | e) = 0.6
  - P(~s | e) = 0.4
- Actions:
  - deliverIntoSpamFolder (dS)
  - deliverIntoInboxFolder (dI)
- Utility function:
  - U(dS, s) = 200
  - U(dI, ~s) = 100
  - U(dI, s) = -100
  - U(dS, ~s) = -500
- Where should this email be delivered and why?

$dS \Rightarrow 0.6 \times 200 + 0.4 \times (-500)$

$120 - 200 = -80$

$dI \Rightarrow 0.4 \times 100 + 0.6(-100)$

$40 - 60 = -20$

Umbrella Example

- Check the weather:
  - P(r) = p
  - P(~r) = 1-p
- Actions:
  - takeUmbrella (t)
  - ~takeUmbrella (~t)
- Utility function:
  - U(~r, ~t) = a
  - U(~r, t) = b
  - U(r, t) = c
  - U(r, ~t) = d
- When should you take the umbrella?

UMBRELLA EXAMPLE

$t \Rightarrow p \cdot c + (1-p) \cdot b$

$\neg t \Rightarrow p \cdot d + (1-p) \cdot a$

- Check the weather:
  - P(r) = p
  - P(~r) = 1-p
- Actions:
  - takeUmbrella (t)
  - ~takeUmbrella (~t)
- Utility function:
  - U(~r, ~t) = a
  - U(~r, t) = b
  - U(r, t) = c
  - U(r, ~t) = d
- When should you take the umbrella?

Value of Information

- If I can buy more information to help with my decision, up to how much should I pay for that information?
  - Currency here is in “utility” units
- 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

Value of Information – All Binary

- Probability of state:
  - $P(s) = p$, $P(\sim s) = 1-p$
- Actions
  - a, ~a
- Additional info, E
  - $P(s \mid e) = q$, $P(\sim s \mid e) = 1-q$
  - $P(s \mid \sim e) = r$, $P(\sim s \mid \sim e) = 1-r$
  - $P(e) = v$, $P(\sim e) = 1-v$
- Utilities
  - $U(s, a) = u1$
  - $U(s, \sim a) = u2$
  - $U(\sim s, a) = u3$
  - $U(\sim s, \sim a) = u4$
- What is the value of E? VOI(E)?

NEXT

- Multi-armed bandit
- Markov decision processes
- Reading
  - Chapter 17 of the AI book by Stuart & Russel (http://aima.cs.berkeley.edu/)
  - Chapters 2 and 3 of the RL book by Sutton & Barto (http://www.incompleteideas.net/book/the-book-2nd.html)

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/negative 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)?

NOTATION

- A_t: action at time t
- R_t: reward at time t
- Sequence of actions and rewards:
  $A_1, R_1, A_2, R_2, \cdots, A_T, R_T$
- $q_*(a) \ = E[R_t \mid A_t = a]$
  - Not given; otherwise, the solution is trivial
- $Q_t(a)$: Average reward for action a, up to, excluding, time t
  - Estimated using prior experiences
  - If action a has never been taken before time t, $Q_t(a)$ is initialized to a default value

Optimal Strategy

- $\underset{a}{\operatorname{argmax}}\, q_*(a)$
- At time t, choose the action that has the highest expected reward
- Challenge
  - We do not know $q_*(a)$

Greedy Strategy (Exploit only)

- $\underset{a}{\operatorname{argmax}} Q_t(a)$
- Calculate the average reward for each action, up to time t, and choose the maximum one
- Problems
  - Initial values of $Q_t(a)$ plays a big role
  - It simply tries the best action it has found; purely exploitation
  - Could easily miss other actions that are better but not tried

Random Strategy (Explore Only)

- Choose an action at random
- Good
  - Explores all the time
- Bad
  - Does not exploit; does not learn

Epsilon Greedy

- Given an exploration parameter ϵ
- At each step t:
  - With probability ϵ, choose a random action (explore)
  - With probability $1 - \epsilon$, choose the current best action: $\underset{a}{\operatorname{argmax}} Q_t(a)$ (exploit)
- $\epsilon = 0$ is fully greedy, and $\epsilon = 1$ is fully random

Simulation Setup

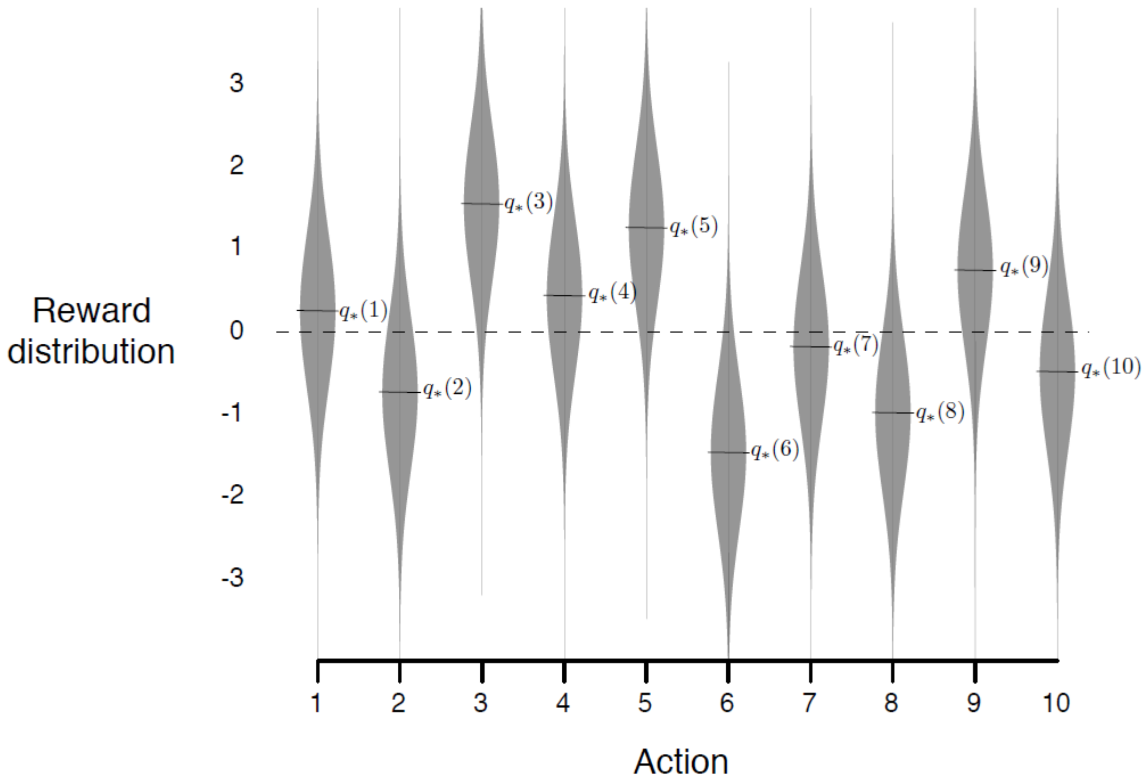


Figure 2.1: An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$, unit-variance normal distribution, as suggested by these gray distributions.

Figure from http://www.incompleteideas.net/book/the-book-2nd.html

Simulation Results

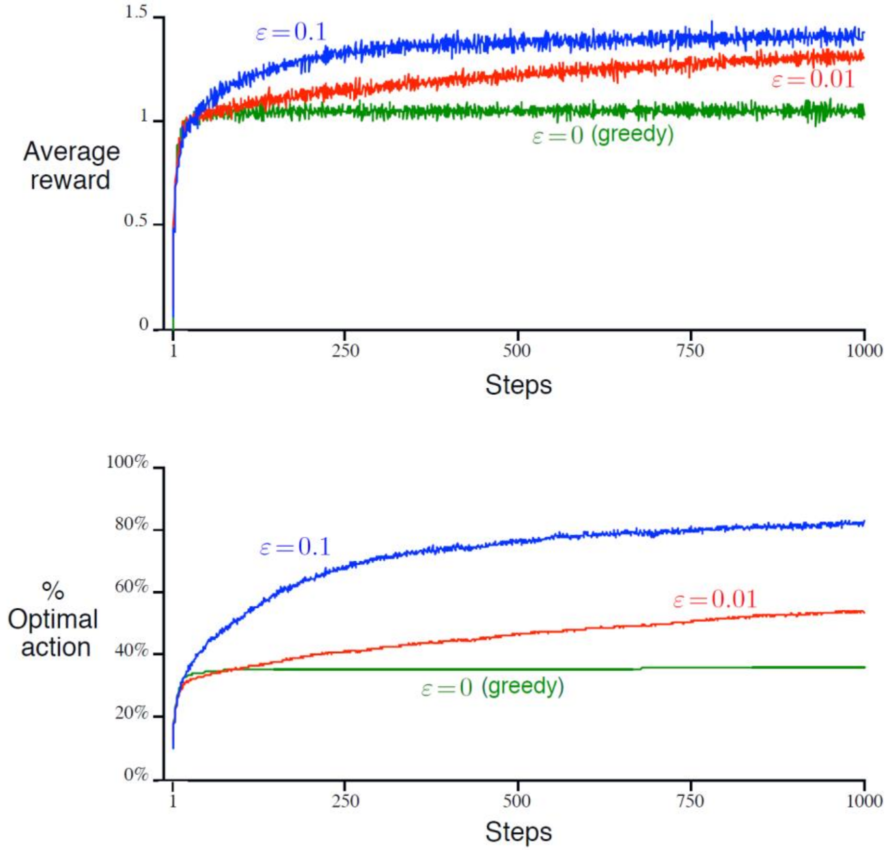
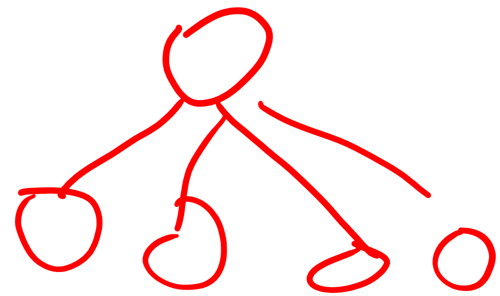


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

Figure from http://www.incompleteideas.net/book/the-book-2nd.html

UCB



- The upper-confidence bound (UCB) method calculates the upper bound on the mean for each slot and chooses the machine with max value
  - $Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}}$, where $N_t(a)$ is the number of times the action a is tried and c is the trade-off parameter
- The term in the square root is a measure of uncertainty in $Q_t(a)$; and hence the name upper confidence bound
  - The upper confidence is derived using the Chernoff-Hoeffding bound
- The exploration grows with *ln(t)*, shrinks with $N_t(a)$
- We've seen this before; where?

Simulation Results

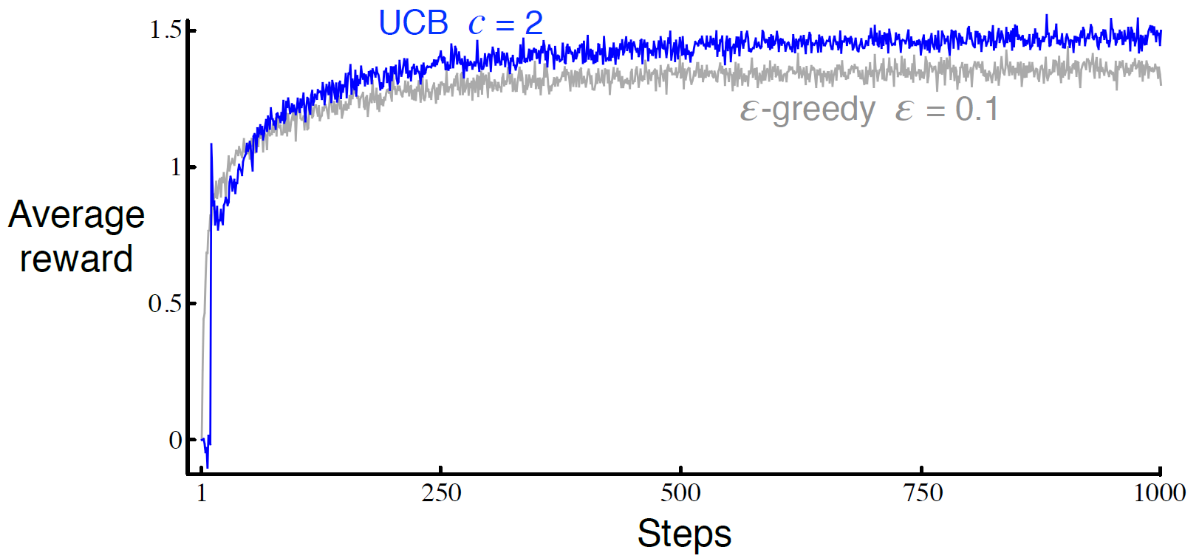


Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

Figure from http://www.incompleteideas.net/book/the-book-2nd.html

A Running Average Computation

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
Q_{n+1} &= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}
$$

Figure from http://www.incompleteideas.net/book/the-book-2nd.html

Update Rule

- $Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$
- new = old + stepSize*(target – old)
- For computing the exact average, stepSize is a function of n

Non-stationary Rewards

- What if the reward distribution changes over time?
- We'd like to give recent rewards more weight
- A simple approach
  - $Q_{n+1} = Q_n + \alpha(R_n - Q_n)$
- Earlier rewards have lower weights
  - $$\begin{aligned} Q_{n+1} &= Q_n + \alpha(R_n - Q_n) \\ &= \alpha R_n + (1-\alpha)Q_n \\ &= \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1-\alpha)Q_{n-1}] \end{aligned}$$

$\underline{\alpha} R_n + \underline{(1-\alpha)\alpha} R_{n-1} + \cdots$

Markov Decision Processes

Problem Setting

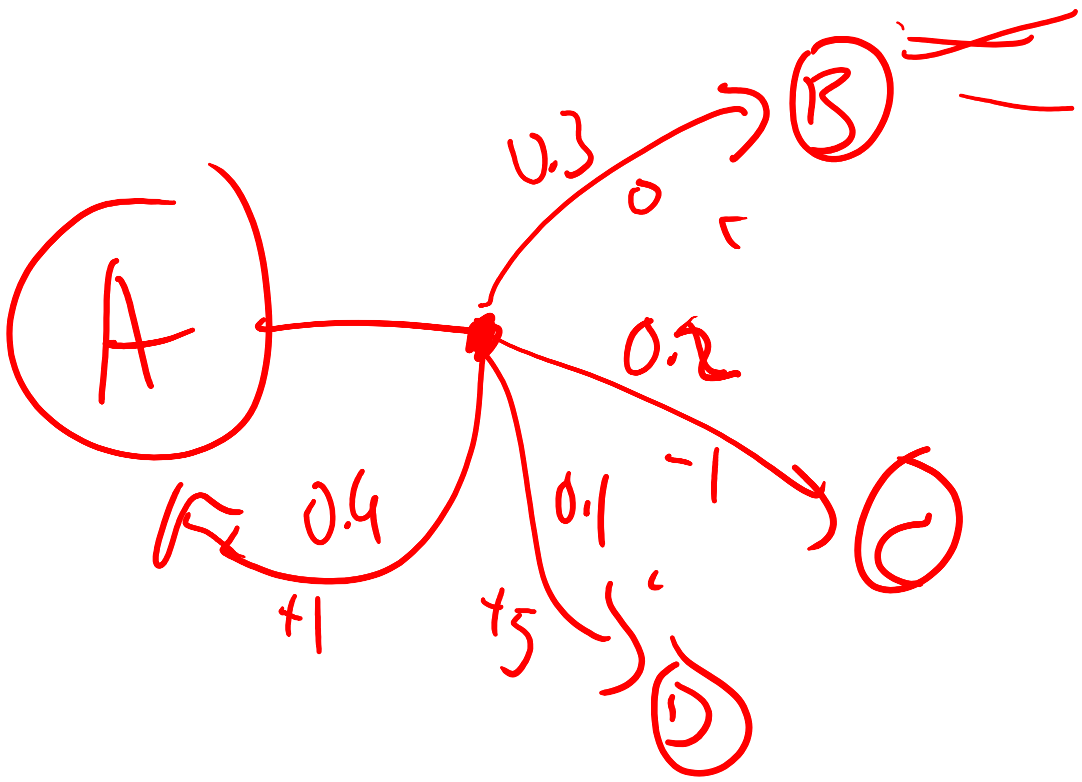
- The world is represented through states
- At each state, an agent is given 0 (terminal states) or more actions to choose from
- Each action moves the agent, probabilistically, to a state (could be the current state) and results, probabilistically, in a reward (could be zero, negative, positive)
- The agent needs to maximize the sum of the rewards it accumulates over time
- Greedy strategy with respect to immediate rewards often do not work; the agent needs to consider the long-term consequences of its actions

Markov Decision Process

- A sequential decision-making process
- Stochastic environment
- Markov transition model
- Additive rewards

MDP Diagram Example

MDP Diagram Example



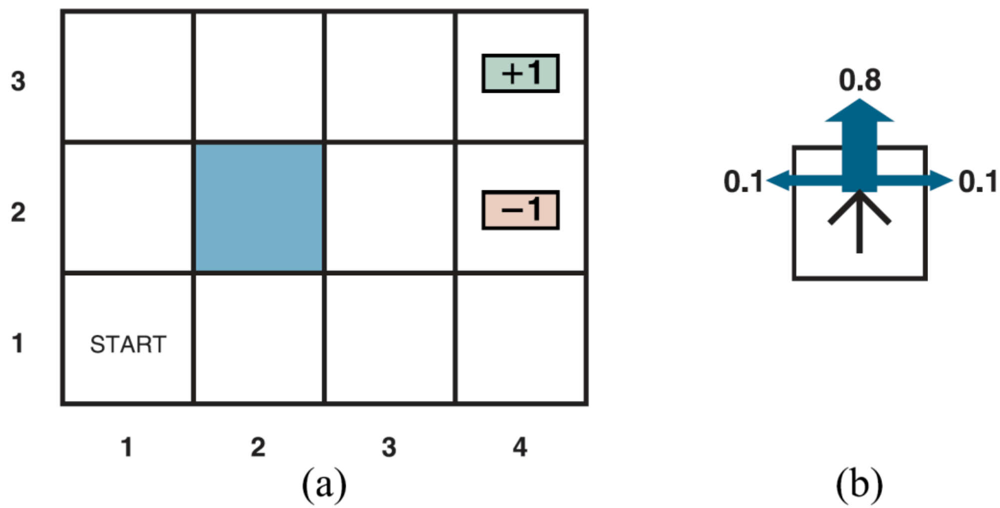
NOTATION

- $P(s' \mid s, a)$ 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 often abbreviated as $U(s)$
- $Q(s, a)$ The expected utility of taking action a at state s
- γ Discount factor [0, 1]

UP, DOWN, LEFT, RIGHT

Running Example

0.1 .8 0.1



0.8 UP
0.1 0.1

0.8

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

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
  - $\boldsymbol{\pi^*}$

$\gamma = 1$

Running Example

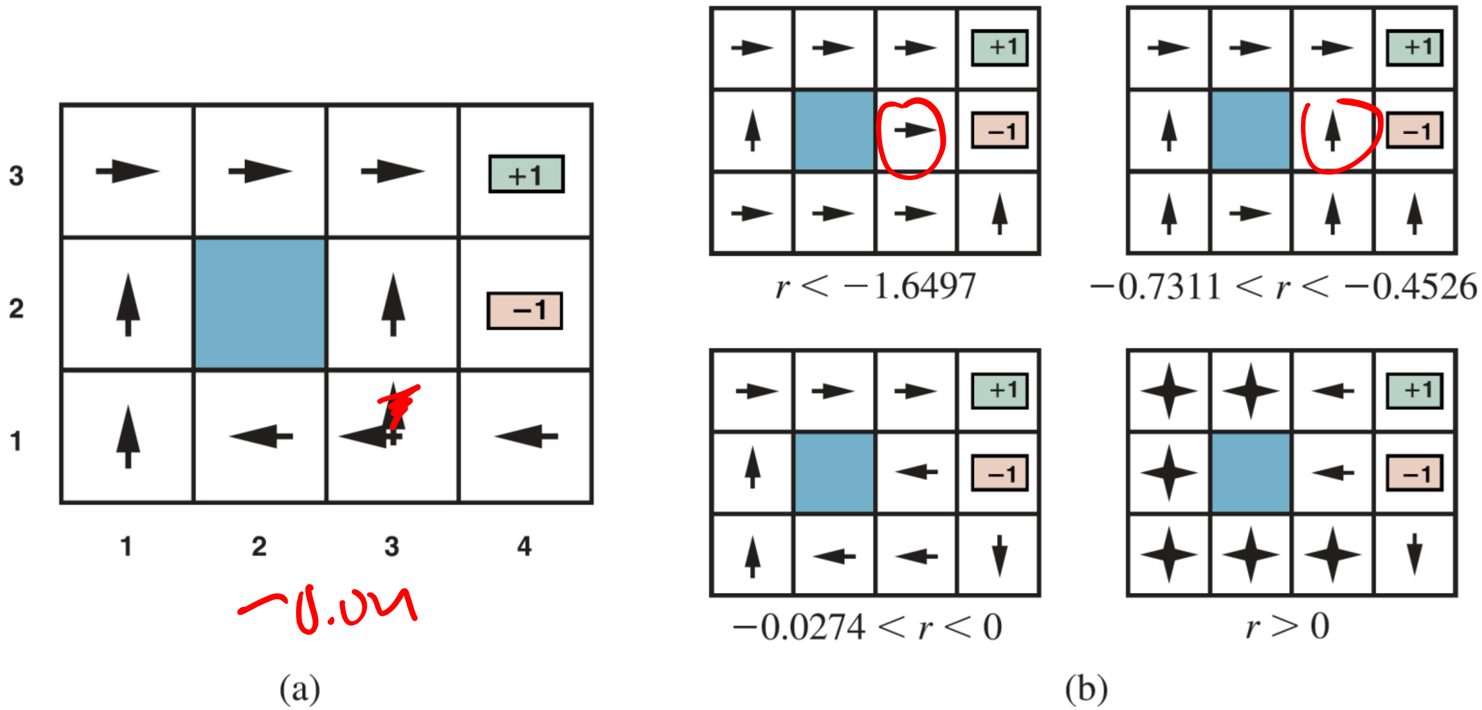


Figure 17.2 (a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. (b) Optimal policies for four different ranges of r.

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

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

$$U^{\pi}(s) = \sum_{s'} P(s' | s, \pi(s))[R(s, \pi(s), s') + \gamma U^{\pi}(s')]$$

Bellman Optimality Equation

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

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*

ACTION-UTILITY FUNCTION $Q(s, a)$

- $Q^{\pi}(s, a)$ The expected utility of taking action a at state s and then following policy π
- Bellman equation
  - $Q^{\pi}(s, a) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U^{\pi}(s')]$

$\gamma Q^{\pi}(s', \pi(s'))$

Bellman Optimality Equation For $Q(s, a)$

- $Q^{\pi^*}(s,a) = \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma \underline{\max_{a' \in A(s')}} Q^{\pi^*}(s',a')]$
- $U^{\pi^*}(s) = \max_{a \in A(s)} Q^{\pi^*}(s,a)$
- $\pi^*(s) = \operatorname{argmax}_{a \in A(s)} Q^{\pi^*}(s,a)$

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

Running Example

3	0.8516	0.9078	0.9578	+1 A
2	0.8016		0.7003	−1 B
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

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

Running Example

$U(T_s) = 0$

	1	2	3	4
3	0.8516	0.9078	0.9578	+1 A
2	0.8016		0.7003	−1 B
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^*

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

1. Policy iteration
2. Value iteration

Policy Iteration

- Start with an initial policy π_0
- Alternate between

1. Policy evaluation: given policy π_i, calculate U^{π_i}
    - Can be calculated exactly (linear equations) or iteratively
2. Policy improvement: Calculate a new MEU policy π_{i+1}, using the utilities calculated in the previous step

- Stop when policy no longer changes

Iterative Policy Evaluation

- Given a policy π
- Initialize U
- loop until convergence
  - for each state s
    - Update U using the Bellman equation
    - $U^{\pi}(s) = \sum_{s'} P(s' \mid s, \pi(s))[R(s, \pi(s), s') + \gamma U^{\pi}(s')]$

Policy Improvement

- Given U
- $\pi(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]$

POLICY ITERATION

- Loop
  - Given π, run iterative policy evaluation to calculate U
  - Given U, run policy improvement to update π

Value Iteration

- Intuitively, perform policy evaluation and policy iteration at the same time
- $U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]$

Finding Optimal Policies

- The previous two slides computed U
  - Many people find the concept of U more intuitive than the concept of Q
- Using Q to find the optimal policy makes more sense
  - $\pi^*(s) = argmax_a Q^{\pi^*}(s, a)$

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