Decision Making

Dr. Fayyaz ul Amir Afsar Minhas
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

Agenda

- Making Simple Decisions
  - 16.1 Maximum Expected Utility Principle
  - Human Decision Making (16.3.4)*
- Making complex decisions
  - 17.1 Sequential Decisions and Markov Decision Processes (MDPs)
  - 17.2 Value Iteration
  - 17.3 Policy Iteration
- Reinforcement learning (chapter 21)
  - Passive: TD
  - Active: Q-Learning
  - Policy Search
  - Applications
- See the "Reinforcement Learning Folder"

Environments of Agents

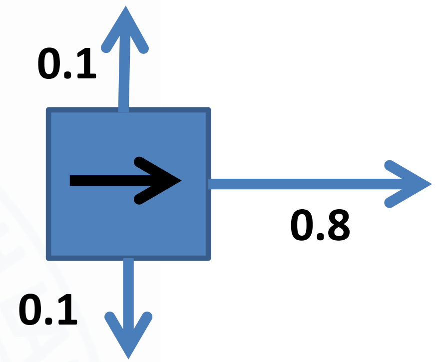
- Up till now, all our discussion has been about deterministic and observable environments
  - Fully vs. partially observable: an environment is fully observable when the sensors can detect all aspects that are *relevant* to the choice of action.
  - Deterministic vs. Stochastic: If the next state of the environment is completely determined by the current state and the action executed by the agent, then the environment is deterministic otherwise it is stochastic

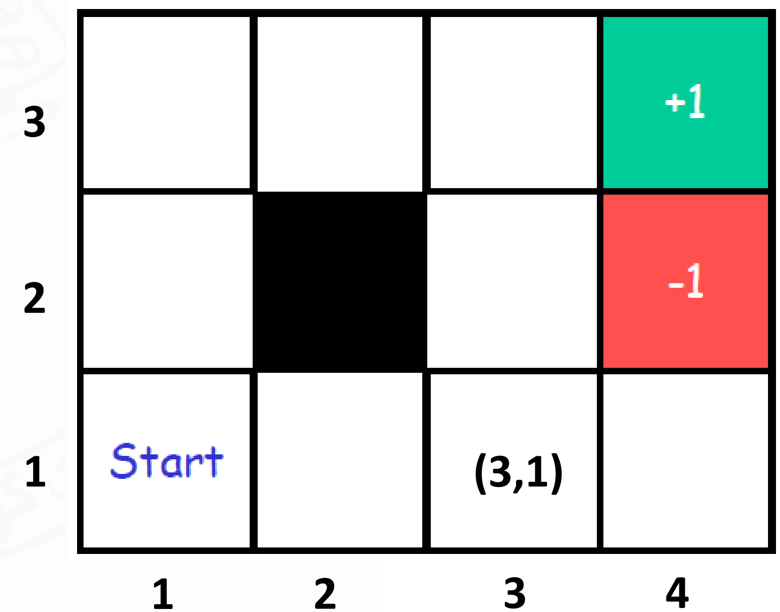
Environment /Property	Crossword Puzzle	Taxi Driving	Medical Diagnosis	Chess (with Clock)	Part Picking Robot
Observable	***Fully***	***Partially***	***Partially***	***Fully***	***Partially***
Deterministic	***Deterministic***	***Stochastic***	***Stochastic***	***Strategic***	***Stochastic***
Episodic	***Sequential***	***Sequential***	***Sequential***	***Sequential***	***Episodic***
Static	***Static***	***Dynamic***	***Dynamic***	***Semi***	***Dynamic***
Discrete	***Discrete***	***Continuous***	***Continuous***	***Discrete***	***Continuous***
Agents	***Single***	***Multi***	***Single***	***Multi***	***Single***

Chapter 2, AIMA

Decisions of Agents

- A decision is an action by an agent
  - In what kind of environment will it be easy to make decisions?
  - Example
    - Consider an agent that can move UDLR in a grid
    - However, due to sensor/actuator errors, it ends up in its intended next square 80% of the time
    - 10% of the time it ends up at a right angle from the intended target
- To maximize the reward, what will be the sequence of decisions of the agent?

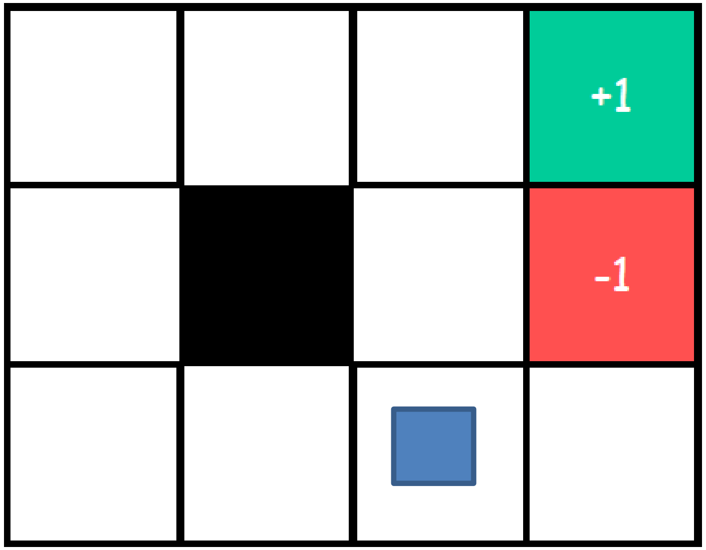




Handling Uncertainty

- In a non-deterministic environment
  - Assume the current state is "s"
  - An action "a" is performed in this state
  - The probability that the result of this action produces state s' is:

$$P(Result(a, s) = s' | a)$$

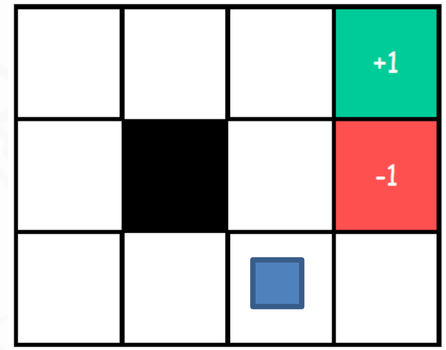


Assume that the agent executes an action "U" in state (3,1)

$$P(Result(U, (3,1)) = (3,2) | U) = 0.8$$
$$P(Result(U, (3,1)) = (2,1) | U) = 0.1$$
$$P(Result(U, (3,1)) = (4,1) | U) = 0.1$$

Handling partial observability

- Now also assume that the environment is only partially observable
  - We do not know what the current state is
    - No direct knowledge of the state
  - We do have an observation "e" which is related to the state
    - $P(s_{cur} = s|e)$ is the probability of being in state s if we observe "e"
    - In fully observable environments "e" fully determines "s"
  - Given the evidence "e", the probability of an action resulting in state s' from s' with action a:
    - $P(Result(a, s) = s'|a)P(s_{cur} = s|e)$



Assume that the agent executes an action "U" in state (3,1)

$$P(Result(U, (3{,}1)) = (3{,}2)|U) = 0.8$$
$$P(Result(U, (3{,}1)) = (2{,}1)|U) = 0.1$$
$$P(Result(U, (3{,}1)) = (4{,}1)|U) = 0.1$$

Assume:

$$P(s_{cur} = (3{,}1)|Smell) = 0.9$$
$$P(s_{cur} = (2{,}1)|Smell) = 0.1$$

Uncertainty and Partial Observability

- Assume you do not know what state the agent is in. All that is known is evidence "e".
- Now given this evidence, what is the probability that we reach a state s' by an action a

$$P(Result(a) = s'|a, e) = \sum_{s \in S} P(Result(a, s) = s'|a) P(s_{cur} = s|e)$$

S is the set of all possible states

Can you calculate the probability of reaching state (3,2) by executing the action "U" in a state where you get the smell:

$$P(Result(U) = (3,2)|U, Smell)$$

Utilities of states and actions

- What is the "expected" utility of executing an action "a" in a state where the evidence is "e"?
  - Denoted by $EU(a|e)$
  - Assume we have a utility function U(s) that tells us the utility or desirability of each state "s"
    - Those states are more desirable that have higher utility
      - E.g., can reflect the possible reward achievable from s
  - Thus, the utility of executing an action "a" when the evidence is "e" depends on the utilities of the resulting states
  - $EU(a|e)$ will just be the utility of the resulting state weighted by the probability that the agent ends up in that state
  - Mathematically,

$$EU(a|e) = \sum_{s' \in S} P(Result(a) = s'|a, e)U(s')$$

Making Decisions

- What decision should the agent make, i.e., what action should be taken:
  - Principle of Maximum Expected Utility (MEU)
    - It says that the rational agent should choose an action that maximizes the agent's expected utility

$$action = argmax_{a \in A}\, EU(a|e)$$

  - In a sense, this simple principle captures all of AI by defining how agents should behave in any kind of environment
  - This is a generalization of all cases!

Let's do it in reverse now!

- Given an observation or evidence "e" What action should be taken
  - The one that maximizes the expected utility

$$action = argmax_{a \in A}\, EU(a|e)$$

  - How can we calculate $EU(a|e)$
    - Assume that our belief of ending up in state s' as a consequence of action a is $P(Result(a) = s'|a, e)$
    - The utility of being in s' is $U(s')$
    - Thus:

$$EU(a|e) = \sum_{s' \in S} P(Result(a) = s'|a, e)U(s')$$

Reversing…

- How do we calculate the belief of ending up in state s' as a consequence of action a:
  - Assume that the belief of currently being in state s given an evidence “e” is $P(s_{cur} = s|e)$
  - Assume that the belief of ending up in state s’ from state s by performing action a is $P(Result(a, s) = s'|a)$
  - Thus, the belief of ending up in state s’ from state s when action a is performed and observing evidence e is: $P(Result(a, s) = s'|a)P(s_{cur} = s|e)$
  - Thus, the belief of ending up in state s’ when the evidence is e and an action a is performed is:

$$P(Result(a) = s'|a, e) = \sum_{s \in S} P(Result(a, s) = s'|a)P(s_{cur} = s|e)$$

Thought Experiments

- What will happen if the environment is fully observable?
  - Your observation uniquely and unambiguously determines the state, thus, $e \equiv s$
    - $P(s_{cur} = (3,1)|Smell) = 1.0$
- What will happen if the environment is deterministic?
  - Your actions have the intended consequences
    - $P(Result(U,(3,1)) = (3,2)|U) = 1.0$

Problems

- The principle doesn't present a solution in itself
- It tells, what action should I take given
  - The utilities of each state
    - Where do those come from?
    - We may not get immediate feedback of how good or bad the move is
      - Example: You can only know how good an action is based on whether you won or lost the game at the end
  - The probability values
    - $P(Result(a) = s'|a, e) = \sum_{s \in S} P(Result(a, s) = s'|a)P(s_{cur} = s|e)$
- What if the state, action or observation set is not finite or very large?
- How will the solution be represented?

These lectures

- How to solve these problems!
- Why?
  - Because, this allows cool things to be done like the ones we saw in the videos

Reinforcement Learning

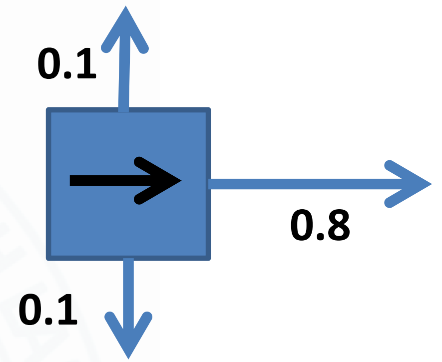
- Applies the MEU on cases where
  - Only the end state rewards are known (aka reinforcement)
  - The utilities of individual states is not known before hand
  - The probability values are not known
  - The action, state or observations space can possibly be very large
- Returns a rational action at each time step

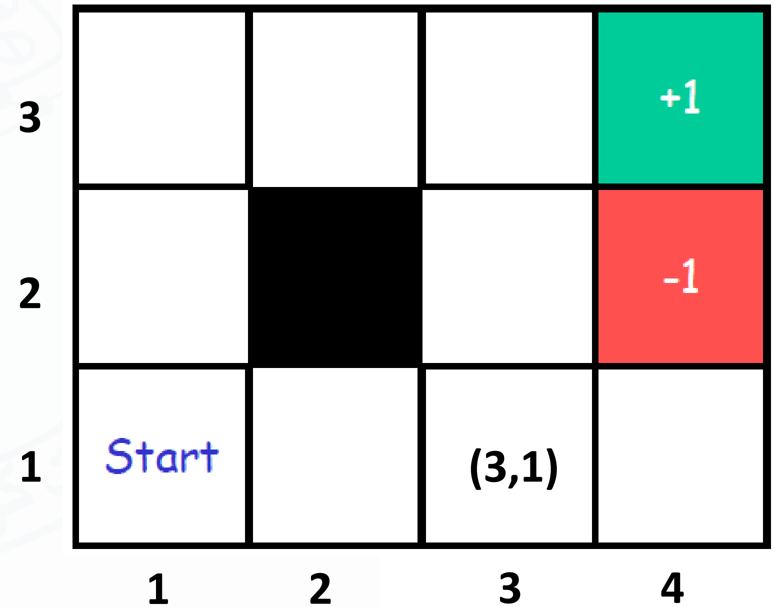
However For Now Assume

- The environment is sequential
- That the probability values are given
  - Technically called “Transition Model” or just the “Model”
- The environment is fully observable but not deterministic
  - Thus we can represent the probability of reaching s’ from s by action a as:
    - $P(s'|s,a)$
- The environment is Markovian
  - The probability of ending up in s’ from s depends only on s and not the previous states leading up to s and the action a taken in state s
- The utility function for the agent depends upon the sequence of its states or environment history
  - The agent receives a bounded **reward** R(s) in state s
  - The utility of an environment history (sequence of states) is a sum of the rewards received
- Such a decision problem is called a **Markov Decision Process** or **MDP**

Example (4,3) World

- The world is fully observable
- We know what state the agent is in
- The actions are stochastic
- P(s'|s,a) is dependent only on s and a and these values are given
- The reward function is: R(s) = -0.04 in all states except the terminal states which have rewards +1 and -1
- The utility function for a sequence of states is the sum of rewards of the states
  - For example: If an agent reaches the terminal state with reward after 10 steps, then the utility is 0.6
- Problem
  - Find the sequence of actions that maximizes the utility
  - This problem is a MDP
- If it were deterministic, the solution would be simple
  - UURRR
  - However, due to the uncertainties in the effects that the actions have, the agent might end up in a different state
  - **What are the chances of UURRR reaching the goal +1?**
    - **0.8x0.8x0.8x0.8x0.8 = 0.33**
  - **Are there other ways of reaching the solution using the command sequence UURRR to reach the +1**
    - **0.1x0.1x0.1x0.1x0.8 = 0.33**
  - **What other states can this sequence lead to?**

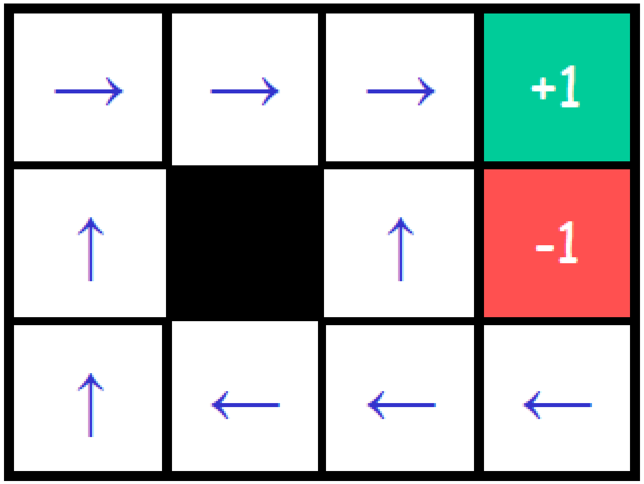




Policies

- Our problem here is a stochastic generalization of the search problems discussed earlier
- What does the solution to the problem look like?
  - A fixed sequence of actions will not solve the problem
    - Such a sequence can end up in states other than the goal
  - Thus, we should specify what the agent should do in any state
  - Such a solution is called a "policy"
  - Denoted by π, such that $\pi(s)$ is the action recommended by the policy π in state s
  - Given a policy the agent always knows what to do

Policy Example



Policy Evaluation

- We can measure how good is a policy by averaging the summation of rewards in states reached by following that policy after starting in a given state. Mathematically, the expected utility obtained by executing π, starting in state $S_0 = s$ is

$$U^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

$0 \leq \gamma \leq 1$ is called the discount factor. It weights the reward in each state in a decreasing manner, i.e., makes rewards in the distant future insignificant

Policy Evaluation

- It can be approximated by say using a Monte Carlo Simulation by generating N trials from the policy and averaging the sum of the utilities of observed states in each trial

$$U^{\pi}(s) = \mathrm{E}[\sum_{t=0}^{\infty} \gamma^{t} R(S_t)] \approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T(i)} \gamma^{t} R\left(S_{i(t)}\right)$$

  - $T(i)$ is the number of states generated in trial i

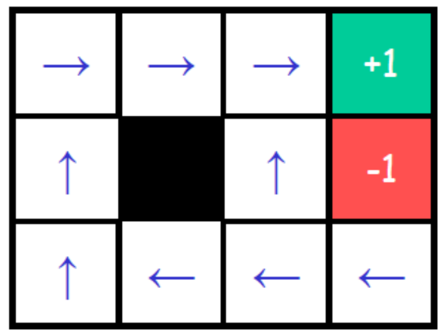
Example

- Utility values $U^{\pi}(s)$ for each state for R(s) = -0.04 for non-terminal nodes and $\gamma = 1$

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

Optimal Policy

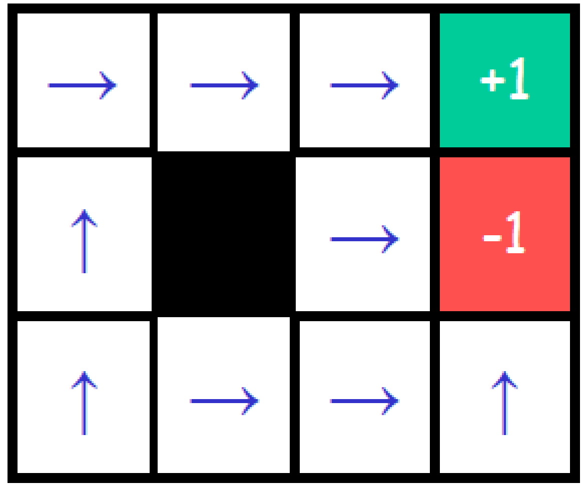
- The solution of the MDP is a policy that maximizes the Expected Utility, i.e.,
  - $\pi^* = argmax_{\pi} U^{\pi}(s)$
- For R(s) = -0.04 for non-terminal nodes, the optimal policy is:



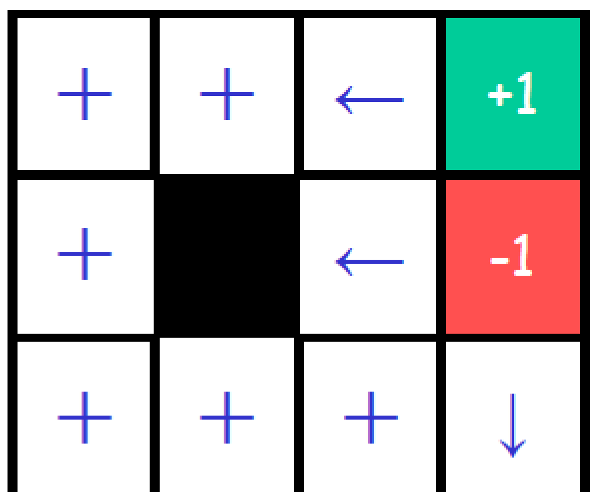
- Changing R(s) will change the optimal policy
  - What is the optimal policy if R(s) > 0?
  - What is the optimal policy if R(s) << 0

Optimal Policies

$R(s) < -1.6284$



$R(s) > 0$



Utilities $\leftrightarrow$ Policies

- Earlier we saw how we can move from a policy to utilities
- We can also move from a Utility table to a policy by using the MEU principle
  - Choose the action that maximizes the expected utility of the subsequent state
    - This utility is given by

$$U(next) = max_{a \in A(s)} \sum_{s' \in S} P(s'|s,a)U(s')$$

    - Thus, the optimal action becomes

$$\pi^*(s) = argmax_{a \in A(s)} \sum_{s' \in S} P(s'|s,a)U(s')$$

Algorithms for finding optimal policies

- Using Optimization
  - Represent the policy
  - Develop a monte carlo implementation of the policy evaluation function which returns the utility of a given policy and use this function as a fitness function
  - Use local search
- Quite numerical and kinda brute force!
- We discuss two other algorithms
  - Value Iteration
  - Policy Iteration

Value Iteration

- Based on **Bellman equations**
- The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming the agent choose the optimal action

$$U(s) = \mathrm{E}[\textstyle\sum_{t=0}^{\infty} \gamma^t R(S_t)] = \mathrm{E}[R(s) + \textstyle\sum_{t=1}^{\infty} \gamma^t R(S_t)]$$

$$= \mathrm{E}\left[R(s) + \gamma \sum_{t=0}^{\infty} \gamma^t R(S_{t+1})\right] = R(s) + \gamma U(next)$$

- We know that the utility of the subsequent state is given by

$$U(next) = max_{a \in A(s)} \sum_{s' \in S} P(s'|s,a) U(s')$$

- Thus:

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

Value Iteration

- How many equations will we get?
  - 9
- How many unknowns?
  - 9
- However, the equations are nonlinear
- Value Iteration is an algorithm to efficiently solve the set of nonlinear Bellman Equations
  - It uses previous values of the utility table to get the next one until the algorithm converges

Value Iteration

- The Bellman Equation for state (1,1) is:

$$U(1,1) = -0.04 + \gamma \max\{ 0.8\, U(1,2) + 0.1\, U(2,1) + 0.1\, U(1,1), \quad (U)$$
$$0.9\, U(1,1) + 0.1\, U(1,2), \quad (L)$$
$$0.9\, U(1,1) + 0.1\, U(2,1), \quad (D)$$
$$0.8\, U(2,1) + 0.1\, U(1,2) + 0.1\, U(1,1) \} \quad (R)$$

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

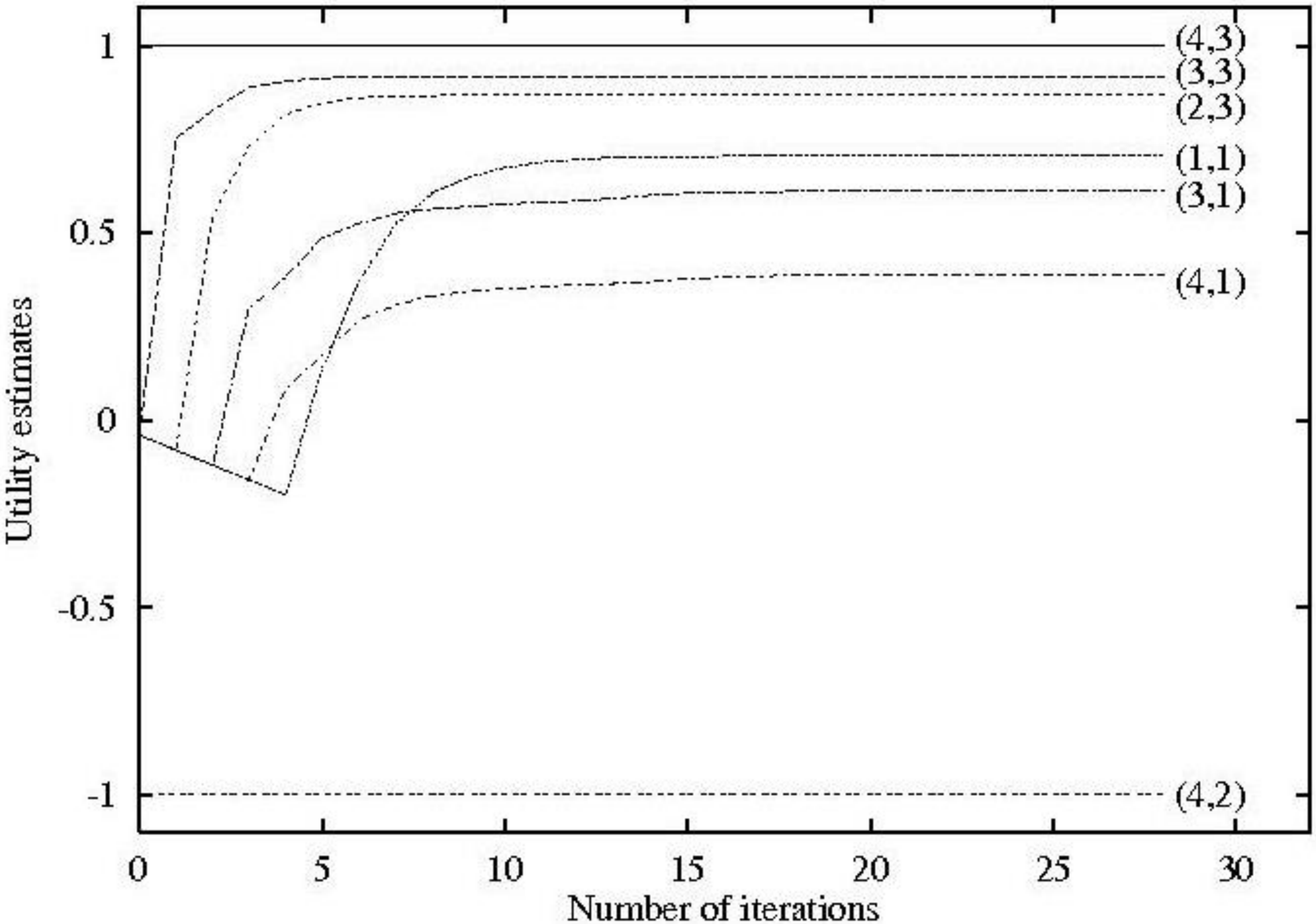
- Update U(1,1) using previous values

$$U(1,1) = -0.04 +$$
$$\gamma \max\{ 0.6096 + 0.0655 + 0.0705 = 0.7456, \quad (U)$$
$$0.6345 + 0.0762 = 0.7107, \quad (L)$$
$$0.6345 + 0.0655 = 0.7000, \quad (D)$$
$$0.5240 + 0.0762 + 0.0705 = 0.6707 \} \quad (R)$$

Algorithm

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    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**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return U



Utility estimates
Number of iterations
1
0.5
0
-0.5
-1
0
5
10
15
20
25
30
(4,3)
(3,3)
(2,3)
(1,1)
(3,1)
(4,1)
(4,2)

Policy Iteration

- The utility function values don't need to be accurate as long as the relative values result in an optimal policy
- Requires
  - Policy Evaluation
    - Given a policy, calculate the utility of each state if the policy is to be executed
    - Easy Peezy: $U_i = R(s) + \gamma \sum_{s' \in S} P(s'|s, \pi_i(s)) U_i(s')$
      - These are linear equations. Can be solved in $O(n^3)$ or we can use modified policy iteration
  - Policy Improvement
    - Calculate a new policy based on the current estimates of the utility

Policy Iteration

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Code

- MDP.PY

End of Lecture

Humans Are the World's Best Pattern-Recognition Machines, But for How Long?

http://bigthink.com/endless-innovation/humans-are-the-worlds-best-pattern-recognition-machines-but-for-how-long