

CS 121, Summer 2009 Homework #6

Out: July 29, 2009
Due: August 10, 2009

How to complete this Homework: Your answers can be typed or carefully hand-written. Please begin each problem on a new page and make sure your name is written on each page of your assignment. Print out this cover sheet and fill in your name and email address, as well as any people you collaborated with. Staple all of your work together and turn in at the beginning of class, August 10, 2009. If you will not be in class you can turn it in prior to class under the door of Gates 132 (with the time of submission written on the homework), or email it to the course staff. Late homeworks will not be accepted.

Your Name:

Your email address:

Note on Honor Code: You must not look at previously published solutions of any of these problems in preparing your answers. You may discuss these problems with other students in the class (in fact, you are encouraged to do so) and/or look into other documents (books, web sites), with the exception of published solutions, so long as your final submission is prepared on your own without referring to any notes taken during such collaboration. If you have discussed any of the problems with other students, indicate their name(s) here:

.....

Any intentional transgression of these rules will be considered an honor code violation.

General Information: Justify your answers, but keep explanations short and to the point. Excessive verbosity will be penalized. If you have any doubt on how to interpret a question, tell us in advance, so that we can help you understand the question, or tell us how you understand it in your returned solution.

Grading:

Problem #	Max. Grade	Your grade
1	18	
2	14	
3	18	
4	0	
5	50	
Total	100	

1. **Markov Decision Process [18 points]** Consider a state space consisting of 6 states $\{1, 2, \dots, 6\}$. Our agent is initially in state 1. The states 4, 5 and 6 are terminal states.

The agent has 4 actions available, actions A , B , C , and D . There is uncertainty in these actions, so each action may have several possible outcomes. The agent's sensors are perfect, so the agent always knows the state that it is in.

Actions A and B can only be performed when the agent is in state 1. If A is executed, the agent moves to state 2 or 3, with probabilities 0.8 and 0.2, respectively. If B is executed, the agent moves to state 2 or 3 with probabilities 0.4 and 0.6, respectively.

We represent these transition rules by:

$A:$	$1 \mapsto \{2(0.8), 3(0.2)\}$
$B:$	$1 \mapsto \{2(0.4), 3(0.6)\}$

Similarly, the transition rules for C and D are defined as follows:

$C:$	$2 \mapsto \{4(0.8), 5(0.2)\}$
$D:$	$2 \mapsto \{4(0.6), 6(0.4)\}$
$C:$	$3 \mapsto \{6(1)\}$
$D:$	$3 \mapsto \{5(0.6), 6(0.4)\}$

The agent must move to a terminal state, but the three terminal states provide different rewards:

State	Reward
4	100
5	200
6	50

Also, there is no reward for the agent to be in a non-terminal state and actions have no cost. The utility of a state is defined recursively as follows:

- The utility of a terminal state is the reward provided by that state
 - The utility of performing an action a in a non-terminal state s is the expected utility of the state reached by performing that action in that state
 - The utility of a non-terminal state s is the maximum utility of performing any action in s
- (a) **[4 points]** Assume that the agent has first executed action A and is now in state 2. What is the utility of action C ? What is the utility of action D ?
- (b) **[6 points]** Assign states 1, 2 and 3 an initial utility of 0. Use value iteration to update the utilities for each state. Report on the utility of every state after each of the first 2 steps of value iteration.
- (c) **[8 points]** A policy is a complete mapping of every non-terminal state into an action executable in that state. For example $\Pi = \{1(A), 2(C), 3(C)\}$ is a policy that tells the agent to perform A in state 1, and C in states 2 and 3. What is the optimal policy Π^* for the above problem?

2. **Normal-form game [14 points]** Consider the following normal form game:

	a	b	c
A	3 4	0 7	2 2
B	1 1	6 5	2 0
C	2 0	7 9	8 3

Player 1 can choose between actions A , B and C , while Player 2 can choose between a , b and c . The utility of an outcome is listed in the bottom left corner of the square for Player 1, and in the top right for Player 2. For example, if Player 1 played B and Player 2 played b , then Player 1 would get 5 units of utility and Player 2 would get 6 units.

- (a) **[5 points]** List all of the pure strategy Nash equilibria of this game
 - (b) **[3 points]** List all of the dominated strategies for either player in this game
 - (c) **[3 points]** As discussed in class, a rational player should never choose a dominated strategy. So, intuitively, we should be able to remove a dominated strategy from the game and not affect the play of the game. Draw the game which results when all of the dominated strategies from Question 2b are removed.
 - (d) **[3 points]** Are any strategies now dominated that weren't before? If yes, draw the game that results from removing these dominated strategies as well.
3. **Decision Trees [18 points]** Suppose we want to learn the predicate *PlayTennis* as a decision tree using up to 4 observable attributes *Outlook* (with possible values Sunny, Overcast, and Rainy), *Temperature* (with 3 possible values Hot, Mild, and Cool), *Humidity* (with 2 possible values High and Normal), and *Wind* (with two possible values Weak and Strong). We are given the following training set made of 14 examples:

Ex #	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	False
2	Sunny	Hot	High	Strong	False
3	Overcast	Hot	High	Weak	True
4	Rainy	Mild	High	Weak	True
5	Rainy	Cool	Normal	Weak	True
6	Rainy	Cool	Normal	Strong	False
7	Overcast	Cool	Normal	Strong	True
8	Sunny	Mild	High	Weak	False
9	Sunny	Cool	Normal	Weak	True
10	Rainy	Mild	Normal	Weak	True
11	Sunny	Mild	Normal	Strong	True
12	Overcast	Mild	High	Strong	True
13	Overcast	Hot	Normal	Weak	True
14	Rainy	Mild	High	Strong	False

- (a) **[5 points]** Which attribute would be selected as the root of the decision tree by the DTL (inductive Decision Tree Learning) algorithm described in class? Why?
- (b) **[13 points]** Show the complete decision tree that would be built by the DECISION-TREE-LEARNING algorithm (page 658 in R & N) to classify correctly all 14 examples in the training set

4. Configuration Space [0 points]

This problem covers material that could be on the final, but this problem will itself not be graded. We invite you to work on this problem as if it were a graded problem, but you do not need to turn anything in for it

Consider the two-link/two-joint robot arm shown in Figure 1. The first link rotates about point J_1 (which is fixed); the second link rotates about point J_2 (which moves with link 1). The orientation of the first link is defined by the angle $\theta_1 \in [0, 2\pi]$ and the orientation of the second link is defined relative to the first link by the angle $\theta_2 \in [0, 2\pi]$. Both angles are measured as shown in Figure 1. The black regions (a square box and a vertical wall) depict obstacles. The initial configuration of the robot is in plain line and its goal configuration is shown in thick dotted line.

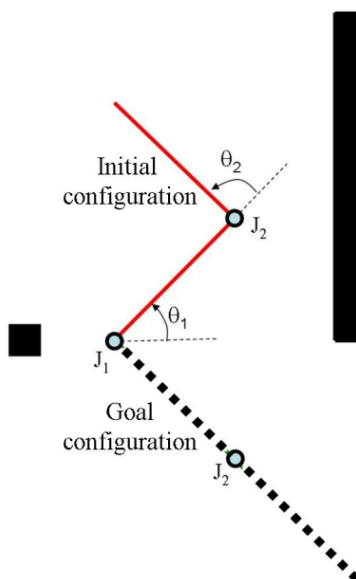


Figure 1: Robot arm and obstacles

- (a) The 4 drawings in Figure 2 present four copies of the configuration space $[0, 2\pi] \times [0, 2\pi]$ of this robot. In each, the horizontal axis corresponds to θ_1 and the vertical axis to θ_2 , as shown in drawing (a); the dark regions are candidate maps of the obstacles. Which drawing depicts the correct map: (a), (b), (c), or (d)? [Here, we do not ask you to check the details of the drawings. Very simple observations should allow you to eliminate three of drawings.]
- (b) In the drawing you have selected, mark the start configuration with a plus (+) and the goal configuration with a bold bullet (\bullet). Draw a collision-free path for the robot to move from the initial configuration to the goal configuration.

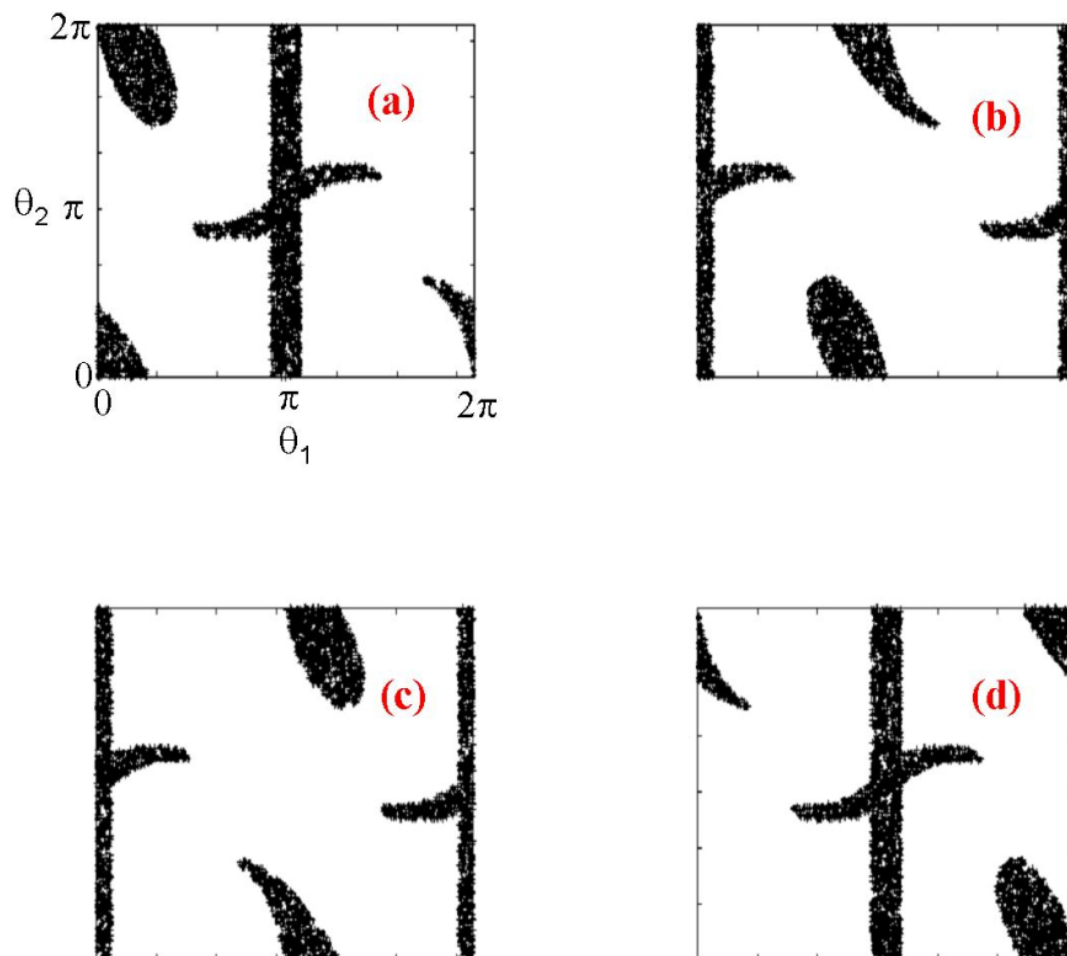


Figure 2: Potential configuration spaces for Question 4

5. **Programming Assignment [50 points]** On homework 3 we investigated the formulation of a few different approaches for solving a Sudoku puzzle. In this programming assignment you will have the chance to actually write code that will solve any Sudoku puzzle given to it. The details of this assignment are as follows:

- This programming assignment can be completed in teams of two students. If you are unfamiliar with programming, or it has been a while, we encourage you to find a partner who can help you in that area. If you need a partner and cannot find one, we encourage you to use the class newsgroup or contact us.
- You are free to do the assignment in any programming language. We have provided a few utilities to assist you in a Matlab file (available from the course website). If you choose a programming environment besides Matlab, we might not be able to assist you.
- Your main task is to write code that will solve any Sudoku puzzle given you as input.

(For details on the Sudoku puzzle see homework 3). You may choose any approach you like, but we encourage you to use ideas and approaches you have learned throughout the course for this specific problem.

- You will turn in to us a short (1-2 page) writeup describing the approach you took, analysis of how well it worked, and commentary on what you might change if you were to do it again.
- Extra credit: There will be an opportunity for extra credit on this assignment. Up to 25 points of extra credit will be available if you implement another, significantly different algorithm for the same problem. (For instance, if you implemented a CSP style approach first, you could also try a local search approach). If you do this, you should also include in your write-up a description of the second approach implemented, and a comparison/contrast of the two approaches. (Your write-up will necessarily be a bit longer). Extra credit will be given at our discretion based on the quality and completeness of both the implementation and write-up.
- You should turn in with your homework the writeup, and then email a copy of your code to the course staff. Please comment your code appropriately so that we can tell what it is doing.
- Honor code: This assignment should be completed without referencing other Sudoku solvers. We expect you to create your approach using the things you have learned in this class or other ideas that you have, not the product of someone else's work. It will be considered an honor code violation to reuse code from someone else.

You will be graded as follows

- (a) **[35 points]** Code. Does your approach work? Does it solve Sudoku puzzles? You should demonstrate this in your write-up both by examples and high-level argument as you describe the algorithm you used.
- (b) **[15 points]** Write-up. Do you clearly describe your approach? Do you analyze its performance carefully and thoughtfully consider improvements.
- (c) **[up to 25 points]** Extra credit. Did you implement a second approach? Did it differ significantly from your first? Did you carefully compare and contrast the two approaches, using real data from example problems?

Please let us know if you have any questions, or need any help with the programming. Good luck