Semester 2 2013
COMP3702/7702 ARTIFICIAL INTELLIGENCE
ASSIGNMENT 2: A Simplified Game of Spy

Note:

- You can do this assignment individually or in a group of 2-3 students.
- For those who choose to work in a group:
  - All students in the group must be enrolled in the same course code, i.e., all COMP3702 students or all COMP7702 students.
  - Send the group name and members (name and student ID) to comp3702-staff@itee.uq.edu.au with subject **group for Assignment2** before **Thursday, September, 19th 11.59am**. If we do not receive your e-mail by then, you will need to work on the assignment individually.
- Please submit your source code and report by e-mail to comp3702-staff@itee.uq.edu.au with subject **Assignment2** before **Monday, October, 7th 11.59pm**.
  - If your code is more than one file, please put them in 1 folder named assg2_studentID(s) and send the folder as a zipped file.
  - If your code consists of only one file, please name it assg2_studentID(s).[extension].
  - Example for naming the folder/file of your source code:
    For a group of students with IDs 12345 and 98231, the folder/file name would be assg2_12345_98231 (followed by appropriate extension).
  - Please ensure that your program can be compiled from command line, and can be run from command line as :
    [java/python/…] assg2_studentID(s) input_file_name output_file_name
  - For the report, please submit your report in .pdf format.
- Please also submit a hardcopy format of your report to 78-349 on **Tuesday, October, 8th**.

Congratulations! You finally land your dream job as a game developer in eArts. For your first project, you are assigned to a team who has been developing a spy game. Your task is to develop a demo for the stealth-tracking module of the game. For this purpose, you need to develop a computer program to track a target, without the target realizing that he is being followed.

Technical details

The tracker and target are represented as points moving in a 2D environment populated by obstacles. To simplify the problem, we assume the environment is normalized, i.e, the positions are in [0,1]X[0,1] and the obstacles are rectangles and axis-aligned. Furthermore, we assume the time step is discrete, with alternating turns between the tracker and the target. At each time step the target

moves or the tracker moves. At the end of its move, the tracker/target receives a reward. When the target is within the tracker's view, a reward of +1 is given to the tracker. When the tracker is within the target's view, a reward of +1 is given to the target. When the target is out of the tracker's view, the tracker can ask help from his headquarters to find the exact location of the target. However, since such an observation from the headquarters may blow up the cover of many other agents nearby, the cost of the information is relatively high, i.e., -5. The game ends when the target enters a designated end region, which has a rectangular shape. In this assignment, your program will play as the tracker. The tracker wins when the total reward he gathered is larger than the total reward the target gathered.

Target

A target is parameterized by its 2D position and heading angle, ie., (x, y, θ). To simplify the problem, let's assume the position space (XY) is discretized into uniform grid cells of size mXm. Each cell is represented by one position, which is the position at the middle of the cell. The target's heading angle is one of the following eight values: {0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0}.

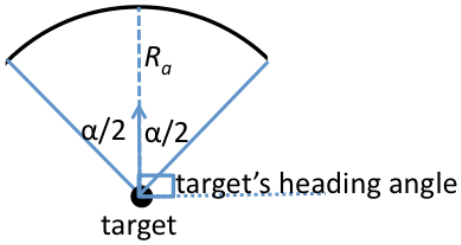

Fig. 1. The target's field of view.

The target has perfect vision, in the sense that he can see any object (including the tracker) and knows the position of the object's exactly, as long as the object is within his field of view and is not occluded by obstacles. To be precise, a point q is visible from another point p whenever the straight line segment between p and q lies inside the field of view of p and the straight line segment between p and q does not intersect any obstacle in the environment. The field of view is limited to a circular sector of a disc with radius R_a and central angle α, spanning [heading_angle - α/2, heading_angle + α/2] (see Fig. 1 for illustration).

The target moves from a given starting position towards a given end region in one of the following ways.

A.1. The target is given a policy. However at each step, the target may diverge from the policy, and cause the target to end at any one of the eight direct neighbors of the target's current position. In this case, we do not have a stochastic model of the divergence.

A.2. The target is given a policy. However at each step, the target may diverge stochastically. The probability of diverging can be extracted from past motion data of the target.

At each turn of the target, the target can stay where it is or make a move. Each move of the target consists of a turn and a step forward in its heading direction. We can represent his action as one of the eight wind directions, i.e., NW, N, NE, W, E, SW, S, and SE.

Tracker

A tracker is parameterized by its 2D position and heading angle. At each turn, the tracker performs a single primitive action. A primitive action may be one of:

- Rotating its heading to any direction and then move forward with a constant speed of 1/m unitLength/unitTime (m: the number of cells in the target's XY space) for a single unitTime. Note that the tracker can rotate without moving forward or move forward without rotate. We can think of this type of action as a displacement vector.
- Shortening or lengthening the line segment in C.2.
- Asking for help from the headquarters.

The motion of the tracker may be one of the following.

B.1. Deterministic. He always moves to where he plans to be.

B.2. Non-deterministic. Due to lack of light, the tracker may make a mistake in its motion. The probability of the error and the probability of making a mistake can be extracted from past motion data of the tracker.

The tracker observes his environment using one of the following two methods.

C.1. His eyes. This is similar to the target's observation model. The tracker has perfect vision but limited field of view. The field of view is a circular sector of a disc with radius R_b and central angle β, spanning [heading_angle - β/2, heading_angle + β/2].

C.2. A handheld video camera that streams live to the tracker's gGlass. The tracker can only observe its environment through the camera. For simplification, we model the tracker as a line segment with varying length between [minLength, maxLength] (see Fig. 2 for illustration). The line segment is always perpendicular to the tracker's heading. One end of the segment represents the tracker's body, while the other end represents the video camera. We assume the field of view of the video camera is the same as the field of view in C.1, i.e, a circular sector of a disc with radius R_b and center v (the end of the segment that represents the video camera), and central angle β spanning [heading_angle - β/2, heading_angle + β/2]. The tracker can shorten or lengthen the segment. The target will only notice the tracker if a sub-segment with length more than half of the segment is within the target's field of view, or if the end segment that represents the tracker's body is within the target's field of view. In this option, the position of the tracker refers to the position of the tracker's body.

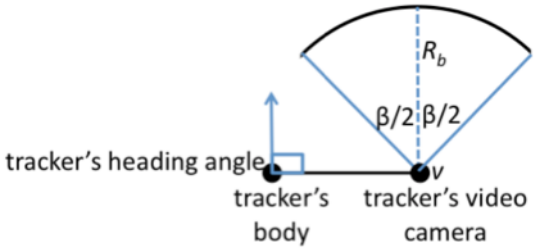

Fig. 2. The tracker's field of view for C2 option.

Submission information

You need to submit your solution in the form of source codes and a report. The report is at most 3 A4 pages, and should contain:

- A description of the proposed method.
- An explanation why you think the proposed method is a good solution.
- An explanation on the limitations of the proposed method.

Please use the provided template for your main function and an example of the code for the target (a2-tools.zip, available in the class website). The template is provided to help the interaction between the tracker (your code) and the target.

Note: *The target's strategy that we will use when marking your code may be different from the strategy encoded in the example code we provide.*

The code takes a text file as input and outputs a log of the game in a text file. The template code already handles reading the input file and outputting the log of the game. Hence, you may not need to read the input file or writing the output file.

Input and Output Format

The input file contains 10+t+n+1 lines, where t is the number of targets and n is the number of obstacles in the environment. The file format is:

Line-1: Number of targets (t).
Line-2: Type of target motion (A1 or A2).
Line-3: File name of the target policy if A1 is used. File name of the target policy and data file for target motion separated by a white space, if option A2 is used.
Line-4: Sensing parameters for the target, in the format of "α R_a". The angle is in degree.
Line-5: Type of tracker motion (B1 or B2).
Line-6: File name of data file for tracker motion if option B2 is used, "-" if option B1 is used.
Line-7: Type of tracker sensor (C1 or C2).
Line-8: Sensing parameters for the tracker, in the format of "minLength maxLength β R_b" if C2 is used, and "β R_b" if C1 is used. The angle is in degree.
Line-9: The initial of the tracker in the form "x y heading" if C1 is used, or "x y heading initialLength" if C2 is used.
Line-10 -> Line-(10+t-1): Initial target states (one per line), in the form "x y heading".
Line-(10+t): Goal region in the form of a quadruple of the x-y coordinates of the vertices (in counter-clockwise order).
Line-(10+t+1): Number of obstacles (n).
Line-(10+t+2) -> Line-(10+t+n+1): obstacles (one per line)
Each line starting from line-10+t+2 is a quadruple of the x-y coordinates of the vertices (in counter-clockwise order) of each rectangular obstacle.

The file targetPolicy.txt consists of m+1 lines where m is the number of rows in the discretized XY space. The format is:

The first line is the number of rows and columns of the discretized XY space, separated by a white space.

Each of the next m lines consists of m number, where the i^{th} number represents the action to perform if the target is at row m column i.

The actions are either staying where he is or moving in one of the 8 wind direction. The codes for these actions are: 0 for NW, 1 for N, 2 for NE, 3 for W, 4 for stay, 5 for E, 6 for SW, 7 for S, and 8 for SE.

Both targetMotionHistory.txt and trackerMotionHistory have the same format. Each file contains n+1 lines, where n is the number of data. The first line is the number of motion data. The next n lines are the data. Each line of data has the format: action next_position. The current_position is always the middle point in the middle cell. The actions are coded as in the targetPolicy.txt for target, and as in "**Note for tracker**" for tracker. The next states are coded relative to the existing state, as shown in Fig. 3a for the target's next state and Fig. 3b for the tracker's next state.

0	1	2
3	4 Current state	5
6	7	8

Fig. 3a. Code for the target's next states.

0	1	2	3	4
5	6	7	8	9
10	11	12 current state	13	14
15	16	17	18	19
20	21	22	23	24

Fig. 3b. Code for the tracker's next states.

From the history data, you can calculate conditional probability of where the next position is, given the current position and action (i.e., transition function in MDP/POMDP) for the target and the tracker. However, the history data are generated in an empty space to understand motion error of the target and tracker. When obstacles and boundaries are present, there might be non zero probability mass that next_position is invalid. We add such probability mass to the probability mass of remaining at the current position.

Note for tracker:

The data discretizes the action into 17 actions, i.e., actions that will move the tracker to cell 1, 2, 3, 4, 5, 6, 8, 9, 10, 14, 15, 16, 18, 19, 21, 22, and 23 if there is no error in the motion. We code these actions according to the cell index, e.g., action code 1 refer to the action that moves the tracker to cell 1. If you use continuous action (i.e., displacement_vector), you can assume the error of your action is the same as the error of the nearest action available in the data.

The data discretizes the environment around the current position of the tracker. If you represent the set of all possible positions of the tracker as a continuous space, the discrete distribution is a marginal distribution over the cells, and inside a cell, all states are equally likely. For example, if from the data we have P(next_position is in cell-2 | current position, action = 1) = 0.8, P(next position is in

cell-0 | current position, action = 1) = 0.1, P(next position is in cell-4 | current position, action = 1) = 0.1 (the cell index follows Fig. 3b) and if the next position without motion error (i.e., current_position + displacement_vector) ends up in cell-2, then the actual next_position with error will be a point inside cell-2 80% of the time, inside cell-0 10% of the time, and inside cell-4 10% of the time. Which point exactly, is sampled uniformly at random from all possible points inside the cell.

The output file contains n+2 lines, where n is the time steps when the target reaches the end region. The format is:

Line-1: Number of turns.
Line-2: Number of targets (t).
Line-3: The initial state of the tracker in the form "x y heading" if C1 is used, or "x y heading initialLength" if C2 is used.
Line-4 -> Line-(4+t-1): The initial state of target-(t-1) in the form "x y heading".
Line-(4+t): The state of the tracker in the form "x y heading reward" if C1 is used, or "x y heading initialLength reward" if C2 is used at turn-1.
Line-(4+t+1) -> Line-(4+2t) : The state of the target- in the form "x y heading reward" at turn-1.
:
:

All example files are available in a2-tools.zip (from the class website).

Grading

Solution & program: 15%.
Report: 5%.

Marking scheme for solution & software:

COMP3702:

- 4: Solve the problem with A.1 for the target's motion, B.1 for the tracker's motion, and C1 for the tracker's sensing. Minimax algorithm is sufficient to solve this problem.
- 5: Solve the problem with A.2 for the target's motion, B.1 for the tracker's motion, and C1 for the tracker's sensing. Minimax and Utility theory are sufficient to solve this problem.
- 6: Solve the problem with A.2. for the target's motion, B.2 for the tracker's motion, and C1 for the tracker's sensing.
- 7: Solve the problem with A.2 for the target's motion, B.2 for the tracker's motion, and C2 for the tracker's sensing.

COMP7702:

- 4: Solve the problem with A.2 for the target's motion, B.1 for the tracker's motion, and C1 for the tracker's sensing. Minimax and Utility theory are sufficient to solve this problem.
- 5: Solve the problem with A.2. for the target's motion, B.2 for the tracker's motion, and C1 for the tracker's sensing.

- 6: Solve the problem with A.2 for the target's motion, B.2 for the tracker's motion, and C2 for the tracker's sensing.
- 7: Up to 3 targets. Solve the problem with A.2 for the motion of each target, B.2 for the tracker's motion, and C2 for the tracker's sensing.
  All targets will have the same sensing capability and the same policy. However, they may start from different positions. At each turn of the target, all targets may make a move. The total reward the targets receive is the sum of the reward of each target. The game ends when at least one of the targets enters the end region.

*) Solve: 3 wins out of 5 games.

Marking scheme for report:

COMP3702:

- 4: A "core dump" of code documentation.
- 5: Clear explaination on the concepts of the proposed solution.
- 6: Requirements for getting a 5 + clear explaination on why you think your solution is suitable for the problem.
- 7: Requirements for getting a 6 + clear explaination on what are the limitations of your solution.

COMP7702:

- 4: Clear explaination on the concepts of the proposed solution.
- 5: Requirements for getting a 4 + clear explaination on why you think your solution is suitable for the problem.
- 6: Requirements for getting a 5 + clear explaination on what are the limitations of your solution.
- 7: Requirements for getting a 6 + convincing arguments that the proposed solution is the most suitable, compared to other existing solutions.

Change log

28 September 2013

Fig. 3b: The size of each cell in this figure is 1/2m.

P.5: "The data discretizes the action into 17 actions, i.e., actions that will move the tracker to cell 1, 2, 3, 4, 5, 6, 8, 9, 10, 14, 15, 16, 18, 19, 21, 22, and 23 if there is no error in the motion. "
should be
"The data discretizes the action into 17 actions, i.e., actions that will move the tracker to cell 1, 2, 3, 5, 6, 8, 9, 10, 12, 14, 15, 16, 18, 19, 21, 22, and 23 if there is no error in the motion."

P.1: "Please submit your source code and report by e-mail to comp3702-staff@itee.uq.edu.au with subject **Assignment2** before **Monday, October, 7th 11.59pm**."
should be

"Please submit your source code and report by e-mail to comp3702-staff@itee.uq.edu.au with subject **Assignment2** before **Tuesday, October, 8th 11.59pm**."

P1: "Please also submit a hardcopy format of your report to 78-349 on **Tuesday, October, 8th**."
should be
"Please also submit a hardcopy format of your report to 78-349 on **Wednesday, October, 9th**."