CourseWare Wiki

NAVIGATION

PUI

assignments

Assignment #1 – Planner

Assignment #2 -- Wumpus World

FF-Replan Tutorial: Robot Emil

Exam

PUI lecture plan 2017/2018

Tutorials plan 2017/2018

MY COURSES

SUMMER 2017 / 2018

BE4M35KO – Combinatorial Optimization

BE4M36PUI – Planning for Artificial Intelligence

BE4M36SMU – Symbolic Machine Learning

ALL COURSES

Summer 2017 / 2018

Older

AE4M36PAH - faculty web

A4M36PAH Czech version of the course

OPPA European Social Fund Prague & EU: We invest in your future.

–Table of Contents

Assignment #2 -- Wumpus World

Help robot Emil find its way to the pot(s) of gold in a world filled with deadly pits. he implementation is possible only in Java. This assignment is undergoing some change for this semester, grading and details are subject to change.

News

- 4/12/18 - The deadline for Task 1 is soft, meaning there will be 1 point penalty for every day of submission after the deadline.

FAQ

Q: How does the testing map look?

A: It is the same size as the public map, it has the same gold and obstacle ratio. It only differs in the seed used to generate the map.

Q: What average utility do I need to achieve to get 7 points?

A: You can get 1, 4, or 7 points in Task 2 and Task 3. The thresholds are identical in both assignments. When you achieve avg. utility > 0, you get 1 point. When you achieve avg. utility > 250, you get 4 points. When you achieve avg. utility > 385, you get 7 points.

Description

Implement decision making routine for an agent in a Wumpus world. The assignment consists of three tasks that will be handed out in parts.

Task	Handed-out	Deadline	Max. Points
Task 1	30.4.2018	13.5.2018 23:59	6 (-1 point per late day)
Task 2	7.5.2018	27.5.2018 23:59	7
Task 3	7.5.2018	27.5.2018 23:59	7

The assignment is considered as “successfully completed” if you get at least 10 points.

The deadline for Task 1 is soft, meaning there will be 1 point penalty for every day of submission after the deadline.

If you have any questions, you can use Assignment 2 Discussion Forum.

Environment

Robot Emil moves in grid world with 4 possible actions that have stochastic effects. Gold, pits and wumpus are terminal states. Reaching gold gives agent large positive utility while falling into a pit or encountering Wumpus gives agent large negative utility. Agent also gets small penalty for each move.

- Robot can execute following actions with stochastic effects (class `Action`):
  - NORTH – Actual effect: 80% NORTH, 10% EAST, 10% WEST
  - SOUTH – Actual effect: 80% SOUTH, 10% EAST, 10% WEST
  - EAST – Actual effect: 80% EAST, 10% NORTH, 10% SOUTH
  - WEST – Actual effect: 80% WEST, 10% NORTH, 10% SOUTH
- If the robot executes a move action that would end up in an obstacle, it bounces back to its current-position.
- The environment is a matrix MxN, where the first index represents columns (x-coordinate) and the second index represents rows (y-coordinate). The columns (rows) are indexed starting from 0, i.e. we have columns (rows) 0,1,…,M-1 (N-1).
- Each cell can contain (class `CellContent`):
  - EMPTY
  - OBSTACLE
  - GOLD
  - PIT
- The simulation finishes if the agent reaches the gold (only in Task 1), falls into a pit, or after h steps.

Rewards/Penalties

- -1 at each move action
- -100 + (-1) for action that results in a pit
- -100 + (-1) when a Wumpus moves to the same cell as the agent
- 100 + (-1) for action that reaches the gold

Task 1 (Value Iteration)

Download the Wumpus world environment with the solution template: Solution template.

Implement decision making for an agent in Wumpus world environment using Value Iteration algorithm. The agent must act optimally over a finite horizon h=200. For more info see e.g. mdps-exact-methods.pdf, page 8.

We will call wumpus.Agent.nextStep() to execute the policy. Any auxiliary code should be in `wumpus.agent` package. To run simulation, execute `WumpusWorldCreator` class.

Your IDE must be configured to use the jar file in `/lib` as a dependency.

Implementation

You can use `WorldModel.getTransitions()` to obtain all transition (along with the transition probability and the reward received) that may occur when a particular action is applied in a particular state.

Debug Visualization

In order to assist you with debugging, we have prepared three visualization layers:

- `State-Value` Layer is activated by pressing “s” and shows a float value for each cell of the environment. The data to be visualized can be set using `DebugVis.DebugVis.setStateValues(float[][] stateValues)` .
- `State-Action-Value` Layer is activated by pressing “a” and shows a float value for each move from each cell. The data to be visualized can be set using `DebugVis.setStateActionValues(float[][][] stateActionValues);` The `stateActionValues` array has three dimensions:
  - x-coordinate of cell
  - y-coordinate of cell
  - direction (0 = NORTH,1 = SOUTH,2 = EAST,3 = WEST)
- Policy Layer is activated by pressing “p”. It shows an arrow indicating the optimal action at each state (cell). The data to be visualized can be set using `DebugVis.setPolicy(Action[][] policy);`

Task 2 and Task 3 (MCTS)

Implement decision making for an agent in Wumpus world environment using Monte-Carlo Tree Search algorithm. The agent should attempt to maximize the cummulative reward over the given horizon. For more info about MCTS in context of MDP planning see e.g. 07-mcp.pdf, slides 19-72 or the following survey. There are many different flavors of Monte-Carlo planning techniques for MDPs and you can chose any of them. For example, you can try single-state Monte-Carlo with random or other baseline policy rollouts, sparse-sampling, adaptive monte-carlo tree search with UCB or epsilon-greedy exploitation-exploration policy or any other instantiation of THTS algorithmic scheme.

Please download the Solution template

Your IDE must be configured to use the jar file in `/lib` as a dependency. To run simulation, execute `Task2Creator` and `Task3Creator` class for Task2 and Task 3 respectively.

Your task is to implement action selection mechanism within wumpus.agent.Agent.nextStep() method such that the agent attempts to maximize the sum of rewards received in the Wumpus world over the horizon of 50 steps and 100 steps in Task 2 and Task 3 respectively.

The solution should consists of wumpus.agent.Agent class and any other supporting classes that however need to be in `wumpus.agent` packages or some subpackage of `wumpus.agent`. The simulator will call wumpus.agent.Agent.nextStep() at each step to obtain an action that the agent executes at the particular state. The agents for Task 2 and Task 3 can be identical or different, whatever you find more appropriate.

In contrast to Task 1, in both Task 2 and Task 3, the simulation continues when the robot reaches gold. On top of that, in Task 3, there are several wumpuses moving randomly in the world, depicted as red circles. When a wumpus steps on the same cell as the agent, the agent dies (reward: -99) and the simulation ends.

Implementation

You can use `WorldModel.performAction(s, a, rnd)` to simulate the outcome of action `a` being executed in state `s` using random number generator `rnd`. The random number generator should be initialized at the beginning of each simulation run as `rnd = new Random(seed)`. If `seed` is a fixed constant, then the random number generator will generate the same sequence of random numbers on each program execution, which is handy for debugging.

Debug Visualization

- State-Label Layer is activated by pressing “l” and shows a the coordinates of each cell of the environment.

Runtime limits

Automatic evaluation scripts cannot run longer than 15 minutes. There will be 15 simulations for Task 2 (with 50 decision points during each simulation), and 10 simulations for Task 3 (with 100 decision points during each simulation). Thus, please limit the time to make a decision at each step of simulation to

- 1.15 second for Task 2
- 0.85 second for Task 3

Performance expectations

In order to get full points for the assignment, your agent should be “reasonably” efficient. If your agent achieves average utility of > 500 in Task 2 and the algorithm generalizes also to different maps, you can expect to get 7 points for Task 2 by AE. Similarly, if your algorithm achieves utility of > 600 in Task 3, you can expect to get 7 points for Task 3 by AE.

Submitting

Submit your solution through CW Upload system before the deadline as a single zip file. At the top-level of the zip file should be `src` directory that contains the source code. That is, the contents of the archive should have the following structure:

- src
  - wumpus
    - agent
      - … (your Java source files)
    - world
      - … (our Java source files)
    - WumpusWorldCreator.java

Tips & Tricks

- Use `javax.vecmath.Point2i` class to represent (x,y) pair of integers.
- You can speed-up/slow-down the simulation on line 101 in `WumpusWorldCreator.java` by changing the third parameter of simulate method, which represents the delay between two actions in miliseconds.
- Numerical value of an action can be obtained as `Action.ordinal()` . I.e. `Action.NORTH.ordinal() == 0` .
- Using `WorldModel.performAction` output contains the `WorldState` object. To determine whether the state is terminal or not, use `WorldModel.getActions(state)` .

courses/pui/assignments/assignment2.txt · Last modified: 2018/05/21 12:43 by mrkosja1