A(E)3M33UI — Semestral project 3:

Markov decision processes

Martin Macaš

Deadline: **18-05- 2014**

Introduction

The goal of this task is to make a practical experience with Markov Decision Processes by implementing the Value Iteration Algorithm.

Technical notes

For the labyrinth environment, implement the Value Iteration Algorithm, for which one can set the discount factor gamma, and the reward r0 (for any non-terminal state). The implementation consists of the java class MySolution.java, which must implement the following methods:

- getActionForState(solveMDP(Environment environment)
  - This method is called by the platform and gives all the states of environment, when environment.getAllStates() is called. A list of objects is obtained. The objects implement ui.StudentStateInterface. This interface filters the unneeded information (position, type of the state). Eventually, a probability that an action leads to a result can be set at the beginning of the solveMDP by calling the method environment.SetProbabilityOfCorrectTransition. The reward for states can be obtained by

```
StudentStateInterface stateA = environment.getAllStates().get(0); // prvni stav

double reward = environment.getReward(stateA);
```

  - Probability of transition from state A to B for a particular action can be obtained in the following way

```
StudentStateInterface stateB = environment.getAllStates().get(1); // druhy stav

environment.getTransitionProbability(stateA, Action.GO_NORTH, stateB);
```

  - The method solveMDP should contain all the necessary demanding computations
  - Once the method solveMDP is finished, the platform calls methods getActionForState(StudentStateInterface state) and fetValueForState(StudentStateInterface state) and shows the result

- getActionForState(StudentStateInterface state)
    - For a particular state, it gives one of possible actions (one from ui.Action). The states can be compared by ==.

- getValueForState(StudentStateInterface state)
    - For a particular state, it gives its value. This value depends on the particular solution. The states can be again compared by ==.

    - getActionForState(StudentStateInterface state)

This hint can help you to run the solution with NETBEANS, although there are other ways that you can use:

1. Download the archive mdptestbed-v0.5.zip, which contains all the needed support for the task solution
2. Extract the archive
3. In NETBEANS: NEW PROJECT – JAVA PROJECT WITH EXISTING SOURCES, in Existing sources, fill in the path to SRC folder from the extracted archive into Source Package Folders and create new archive
4. In package ui, create a new class MySolution.java, which implements ui.SolutionInterface. Inside the class, implement the value iteration algorithm
5. In NETBEANS, open File - Project Properties – Run, fill in Arguments with the class you implemented
6. Run project and open a maze from the subfolder mazes in the archive

The documentation is also included in the archive.

Experimental goals

Make experiments with the following values of the parameters:

- Discount factor gamma: 0.5, 0.9, 0.99, 1
- Reward r0 (for each nonterminal state): -10, -3, 0, 5
- Maximum error epsilon: 0.1, 0.01, 0.001

Report and discuss the results:

- The resulting strategies pi(s) and utility functions U(s) for 3 selected interesting combinations of the parameters gamma, r0 and epsilon.
- The dependence of number of iterations and the total runtime on the
    - Discount factor gamma (for epsilon=0.01 and r0=-10)
    - Maximum error epsilon (for gamma=0.99 and r0=-10)

- Compare the number of iterations needed for utility function convergence and for strategy convergence. Discuss the difference.

Evaluation

Maximum score is 10 points. If you will not submit a working implementation, you obtain 0 points.

Implementation covers 0-4 points and the evaluation criteria are:

- Satisfaction of the above specification.
- Successful pass in a validation trial consisting of one maze selected by the teacher

Report covers 0-6 points and the evaluation criteria are:

- A comprehensive and detailed description of the implementation (MySolution.java)
- Quantitative results of the experiments (tables, graphs)
- A reasonable discussion of the experimental results
- Grammatical and formal aspects

Bonus covers 0-2 points

- 1 point if the report is written in LATEX and its source-files are also submitted

Submission

ZIP archive must be submitted that contains the following files:

- File named *MySolution.java* with the required source code.
- All the other java codes that are necessary.
- File surname.pdf with the report, which includes the following sections: Implementation, Experimental results, Discussion.
- LaTeX sourced if LaTeX was used.

Deadline: 18-05- 2014