Assignment 4 - Reinforcement Learning, Bayesian Nets

Artificial Intelligence
WS 2015

Due: 25th January 2016, 23:55 noon

General Information

Before you start, make sure you download the game framework version released for the fourth assignment from the Moodle site. Empty classes and methods of the algorithms to be implemented are available in the package `at.jku.cp.ai.learning`. Please write your code into these dummy classes.

To hand in your solutions, please create a **ZIP** file containing your implementations and the report, as a **PDF** file for the non-programming assignments. The **ZIP** file should contain the following items, **and nothing else**.

```
assignment4
├── report.pdf
└── src
    └── at
        └── jku
            └── cp
                └── ai
                    └── learning
                        └── QLearner.java
```

Make sure that the **ZIP** file also contains the whole directory structure, so when you unzip it, it unzips into a folder named `assignment4`, containing the rest of the tree outlined above. Upload the **ZIP** file via MOODLE.

You can test your implementations on a set of unit tests available in the package `at.jku.cp.ai.tests.assignment4`. Note however that passing all these tests does not necessarily mean that your implementation has no flaws, since the tests do not cover all possible invariants! You are welcome to create your own tests and submit them together with the rest of your code (these would go in an optional package `at.jku.cp.ai.tests.assignment4` in the **ZIP** file)

1 Bayesian Nets – Constructing a Net

Construct a Bayesian Net that models the state of mind of a unicorn during a game of Rainbows and Unicorns. The first thing we want to capture is whether the unicorn is **stressed out** or not. A unicorn tends to be stressed out if there are **rainbow seeds** near. But if the unicorn is **experienced**, it tends to be relaxed. Being stressed out influences whether the unicorn makes **good moves**. If it makes good moves, it won't go **sailing** as quickly, which makes the unicorn **happy**. A unicorn can either be *very happy, neutral, or not happy*. Also, being stressed out lowers the unicorn's happiness.

(A) Draw the best Bayesian network you can think of for this domain. Explain every step in detail (i.e., the variable ordering you chose, why you chose this ordering, why you model each dependency, ...). *(3 points)*

(B) How many independent values are contained in the joint probability distribution, given the nodes from above and assuming that no conditional independence relations are known to hold among them? How many independent values would your network tables contain?[1] *(2 points)*

[1] Hint: These numbers also depend on the number of values the variables can take on. For example, a table holding the joint probability distribution of two boolean variables needs to store *three* values (the fourth can be computed), while a table for two variables that can take on three different values (e.g. low, medium, high) needs to store *eight* (the ninth can be computed)!

2 Q-Learning

In this assignment you are presented with a very specific challenge within the rainbow and unicorns framework. The board contains only one unicorn and one cloud. The unicorn can move freely around the board. To win the game the unicorn has to evaporate the cloud and stay alive. The game is lost when the unicorn is sailing away.

This assignment has to be solved using Q-learning, i.e. you have to write an algorithm that *learns* a strategy to solve this task. Solutions based on any other algorithm (e.g. naive search algorithms) are not allowed.

Implementation of Q-Learning

Implement the Q-learning algorithm as presented in the lecture slides ("Algorithm for Learning Q", slide 19) within the provided framework. The algorithm will have to be adapted to your needs. In particular, you are dealing with a case of *episodic learning*. This means that you cannot use one 'infinite' loop for learning like shown in the slide, but that you have to restart the learning procedure every time you end up in a goal state. You can find a proposed structure of the algorithm in pseudo code and some more hints regarding implementation issues in the dummy class. Make sure you implement both the learning method `learnQFunction` and the `getMove` method.

(A) Implementation of Q-Learning (15 points)
`at.jku.cp.ai.learning.QLearner`

Theory

In the framework you can find a comment saying that for our problem at hand the discount factor γ does not influence the learning at all (as long as $\gamma > 0$).

(B) Explain why! (2 points)

(C) Describe the role of γ in the Q-learning algorithm. Can you give an example problem (within the rainbows and unicorns world) for which different values for γ would possibly lead to different solutions? (2 points)

Bonus

In `at.jku.cp.ai.tests.assignment4` you will also find 3 bonus test cases. Normally, in naive implementation of Q-learning, these cases should fail most of the time. You can earn additional bonus points by improving the Q-learning algorithm with (simple!) strategies that make it applicable for these slightly larger problems. Use the same class as in task **(A)** for the implementation, but **clearly mark** your adaptations using source code comments.

(D) Q-Learning (Bonus) (5 points)
`at.jku.cp.ai.learning.QLearner`