CSC3022H: Machine Learning

Practical Assignment:
Reinforcement Learning

Department of Computer Science
University of Cape Town, South Africa

September 19, 2016

Due: Monday, 10 October, 2016, 10.00 am

Overview

This assignment requires you to implement *Q-learning* [Watkins and Dayan, 1992] to solve a simulated mine-sweeping task.

Simulation Framework

Q-learning is to control a set of autonomous mine sweeper agents in a two-dimensional world of mines and super-mines (figure 1). The exact parameters (mines, sweepers, iteration length, frame rate, etc.) for the simulation are set in the `CParams.ini` file.

When a sweeper collides with the one of the super-mines both the sweeper and mine are destroyed. Destroyed super-mines are re-created in the next iteration. If a sweeper collides with a mine, the mine is removed from the field and the sweeper's statistics are updated.

The framework to set up the world, rendering and update loop is supplied. It should not be necessary to modify the framework's core functionality (ie. the draw and update loops), aside from the Q-learning algorithm controller and supporting methods:

- `CQLearningController.cpp`

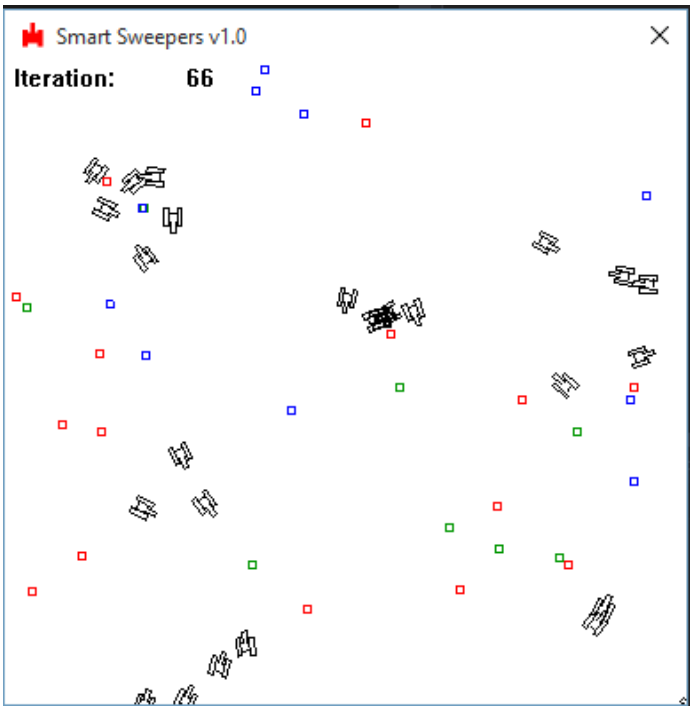

Figure 1: Simulation world consisting of minesweepers, mines and super-mines. Mines and sweepers are spawned at random positions in the world at the start of the simulation. The world wraps around on both the x and y axis (i.e. if the sweeper leaves the world on the window on the one side they reappear on the opposite end of the grid). Sweepers should learn to remove mines (green squares) and avoid super-mines (red squares).

Each of the controllers implements a carefully defined interface.

`CQLearningController.cpp` is a discrete (grid) environment and inherits from `CDiscCollisionObject.cpp`.

The controller `CQLearningController.cpp` overrides the following methods:

- `virtual void Update(void);` This method must call the Update method of the parent controller.
- `virtual void InitializeLearningAlgorithm(void);`

You will need to fill in the details of the methods for the controller and supporting classes, paying special attention to the comments in each file.

The statistics of the current minesweepers are drawn by the PlotStats() function when the F-key is pressed. These graphs (depicting maximum and average number of mines gathered) will be drawn correctly when your mine sweepers learn to gather mines.

The framework uses the Windows `WIN32` windowing API and is bundled as a Visual Studio 2013 project. Both the Senior and the Games lab have copies of Visual Studio installed.

Implement the Q-learning Algorithm

The Q-learning algorithm is a reinforcement learning approach. For a detailed description of the algorithm refer to the Q-learning paper by Watkins and Dayan [Watkins and Dayan, 1992], and Chapter 13 of Mitchell [Mitchell, 1997] (both are available on Vula). There are also many other resources online[1].

The algorithm keeps a table containing all possible states and actions, and progressively updates the table according to the reward function. Since all possible state-action pairs have to be tracked it is easy to see why the world must consist of a finite number of grid positions. You can assume that each mine-sweeper can only move *up*, *down*, *left* and *right*, but that they must move every step of the update cycle. The sweepers can for instance be rewarded for completing the task at hand (clearing the field) and penalized for finding nothing. You can decide on the exact details, learning rates and discount factors to use.

NOTE:

1. The learning simulations can run for as as many iterations as is necessary for your Q-learning algorithm to learn the required behavior.
2. The configuration of mines used in the learning simulations should not be the same as the test environments (see below).
3. You may find keeping a separate table for each mine-sweeper useful when implementing the algorithm for more than one sweeper.

[1] You can use the library *ezproxy* tool http://ezproxy.uct.ac.za/ to download most articles from sites such as IEEE Explore when you're off campus.

Test Environments

To verify your Q-learned behaviour, agent controllers must be tested on the following test environments. For each environment, be sure to record the number of mines gathered and mine-sweepers destroyed.

Environment 1: Initialise the environment with 30 mine-sweepers, 40 mines and 10 super-mines.

Environment 2: Initialise the environment with 30 mine-sweepers, 25 mines and 25 super-mines.

Environment 3: Initialise the environment with 30 mine-sweepers, 10 mines and 40 super-mines.

NOTE:

- For each environment, one simulation can run for only 2000 iterations (simulation ticks).
- For each simulation (run), the mine-sweepers and mines should be initialised in random positions on the grid-world.
- At the end of each run, mine-sweeper task performance is measured by the number of mines gathered, and the number of mine-sweepers remaining.

Submission Details

- You should run the *clean project* option before the ZIP and submit.
- Your report should be compiled as a PDF document, and include:
    1. Pseudo-code, the Q-learning parameters, and a brief justification of parameter values selected and discussion (***200 words maximum***) of your results (why your Q-learning agents perform as they do).
    2. A histogram of average results, which includes the average *number of mines removed* and *agents destroyed*, averaged over 20 runs (for your Q-learned behaviour).
    3. The histogram must present average results (mines swept, agents destroyed) for each of the three (3) test environments.
    4. Error bars (indicating standard deviation) should be included in the histogram for each average result.
- You must submit a working Visual Studio 2013 Solution or a *makefile* to compile your program. If it does not compile a 50% penalty will apply.
- If you add additional files and or have any compilation instructions you must provide a README file explaining what each file submitted does and how it fits into the program as a whole. The README file should not explain any theory that you have used. These will be used by the tutors if they encounter any problems.
- Please ensure that the ZIP archive of your git repo works and is not corrupt. Corrupt or non-working archives will not be marked. Please do not use other formats like WinRAR / 7zip / etc.

References

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw Hill, New York, USA.

[Watkins and Dayan, 1992] Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(1):279–292.