Project 5: Wobbie’s World

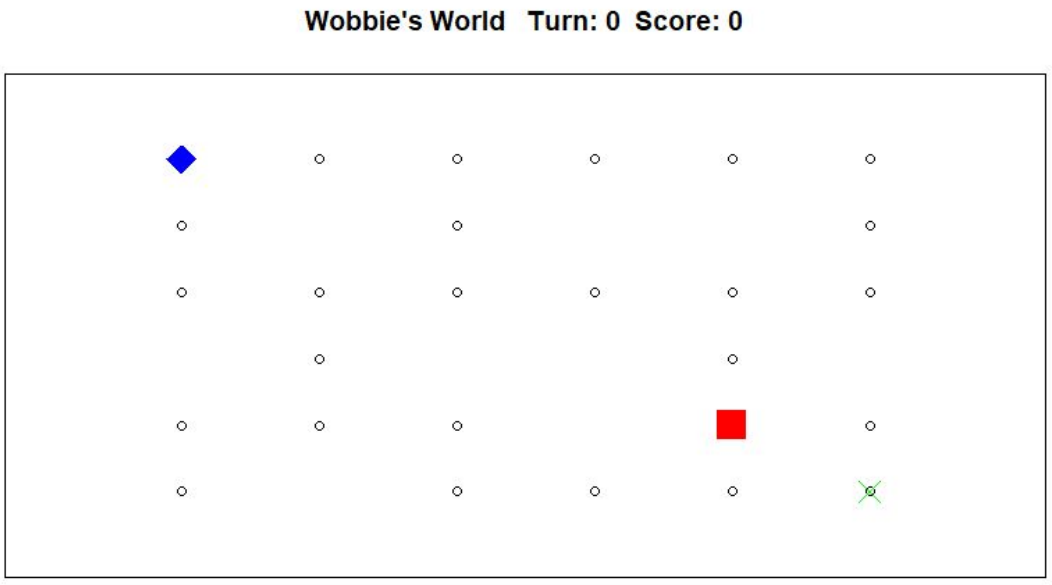
Artificial Intelligence (IDL340)

Basics

- This project should be performed alone or in pairs.
- You will use the R programming language.
- The use of third party code libraries/R packages is not permitted.
- The required code is available in the Diagnostics_1.0.0.zip file on student portal.
- See the R FAQ document on student portal for questions about using R.

Project Overview

Wobbie’s World is a simple game. Wobbie is seeking to escape from a maze. He starts in the top left, and needs to reach the bottom right while avoiding a monster (see image below, Wobbie is a blue diamond, the monster a red square, and the exit a green cross).



Wobbie can move only on the round circles (the define the maze), or stay still, using keyboard movements (2,4,5,6,8). If he tries to make an illegal move, he will stay still. If the monster and Wobbie are on the same square, the game ends. The score at each turn is 0 minus the number of moves taken. If Wobbie is caught, 1000 points are subtracted from the score. The monster moves randomly (always making a valid moves, and never staying still).

Your task is to implement a Q-learning reinforcement learning system that can learn to play Wobbie's world as well as possible. It have 10000 games from which learn how to play, and then will be evaluated on 500 additional games.

You will provide

An R script with a function that can be passed to the runWW function.

Pass Requirements

Your function needs to:

- Equal or surpass par performance
- Meet the execution time requirement

See documentation on the runWW function in the WheresCroc package for details.

Important Functions

The important functions in the WobbiesWorld package are:

Function Name	**Description**
runWW	Performs training and testing for a given controller. The help documentation contains important information required for completing this project.
runGame	Runs one game of Wobbies world. The help documentation contains important information required for completing this project.
learn	Runs the learning portion of the runWW function. You can call it directly.
test	Runs the testing portion of the runWW function. You can call it directly.

Example Control Functions

In addition, there is a controller factory function included, that you can pass to the runWW function:

1. makeRandomController

Examine its help documentation for details. Note that it does not implement a Q-learning solution.

Dangers

If you are seeking very high performance, you might be tempted to go beyond table based Q-learning and fit a parameterized model, such as a DQN-based model. I advise you not to, as this makes the project much more work, and unsuitable for implementation in R.

You do not want to include random actions when performing the evaluation tasks. The package documentation explains how to set up your Q-learner object so that a flag is set allowing you to stop random actions. Make sure to implement your Q-learner so that it looks at this flag when considering whether to make random actions!

Competition Hints (how to improve on the par function)

There are no obvious ways of making big improvements on the par performance without undertaking too much work to be feasible (see dangers).

Exact results, and therefore placings, will depend significantly on chance. So you need to maximize your probability of doing well. The way to do this is to analyze the effect of different learning parameters (learning rate, randomness etc).

If you are ambitious, you could check when the random flag is set to false (see Dangers and package documentation). This will indicate that learning is over, and evaluation runs have begun. You might want to do additional actions at this point, such as try to make all q-values in your Q-learner table consistent with the Bellman equations.

Remember: Implement the basic algorithm and ensure you have a passing implementation before trying to be ambitious!