Introduction to Machine Learning - Assignment 3

Instructor: Dan Lizotte

Due at 12h00 noon on Monday, 14 May 2007

This assignment will explore the difference between two reinforcement learning algorithms: Sarsa and Q-learning. It will also familiarize you with the basics of Bayes classifiers.

This assignment is marked out of 50 and is worth 15% of your final mark. **For this assignment, place the relevant files in a directory called `yourname_a3/`, and e-mail a `.zip` or `.tar.gz` archive of this directory to `dlizotte@ru.is` with `ITME Assignment 3` in the subject line before noon on Monday, 14 May 2007.**

---

1. [*30 points*] **Comparing Q-learning and Sarsa** *Alpaydin Chapter 15. Sutton & Barto Chapter 6.* For this question, you will take an existing Q-learning agent and modify it to create a Sarsa agent. You will then test each agent in two different environments and plot the resulting learning curves.

   To do this, you will use the `RUGridWorldDemo` package available from the course webpage. `http://www.cs.ualberta.ca/~dlizotte/teaching`

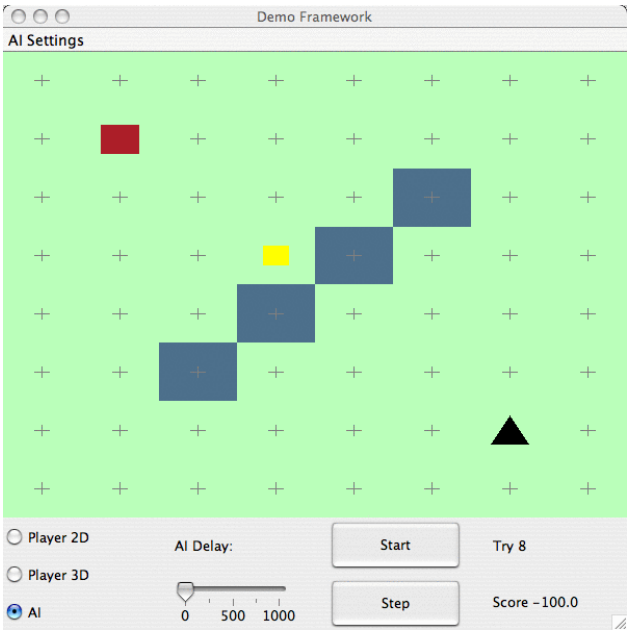
   `RUGridWorldDemo` is a reinforcement learning demo developed by Andrew Butcher and Anna Koop in the Reinforcement Learning and Artificial Intelligence lab at the University of Alberta in Edmonton, Canada. (`http://www.rlai.net/`) The environment is an 8 by 8 grid, with an agent we refer to as the "mouse" represented by a black triangle, an adversary called the "cat" represented by a red square, and some "cheese" represented by a yellow square. There is a unique state for each position of the mouse and position of the cat — therefore there are $(8 \cdot 8) \cdot (8 \cdot 8) = 4096$ states. There are four possible actions for the mouse: Move up, down, left, or right. Any of the grid positions can be marked as a "wall." If an action would move the mouse into a wall, nothing happens. For this assignment, the mouse receives a reward of 1 for each timestep it survives, a reward of 100 for finding cheese, and a reward of -100 for being caught by the cat. Being caught ends the current episode and restarts the mouse and cat in random grid positions. The cat's policy is to move toward the mouse if possible, and to take a random move if this is not possible.

   The GUI that operates this demo shows the mouse, cat, cheese, and walls, and allows the user to create or destroy walls by clicking on grid squares, as well as to drag the agents around on the grid. It also allows the user to run the demo continuously (click "Start") or to run one step at a time (click "Step"). When running continuously, the "AI Delay" slider controls how fast the program runs. The radio buttons on the left choose whether a human controls the "mouse" with the arrow keys ("Player 2D" and "Player 3D") or whether an RL agent controls the "mouse." ("AI".)

   The package includes the `GridWorld` shell script for running the program, and a `makefile` to build the Java classes. These have been tested under MacOS X and Linux. To compile and run the program in either of these environments, you must edit the `GridWorld` script and the `makefile` to uncomment the appropriate reference to the included `jogl` library, which provides graphics support. For example, for a linux platform uncomment only the line `JAVA_PLATFORM=jogl-1.1.0-rc2-linux-i586` in each file. Once you have selected the correct library, you should be able to run the environment with the `GridWorld` script, and compile the source files by typing `make`.

   On Windows, you need to do the same thing but modify and use the `WinGridWorld.bat` and the `WinMake.bat` files instead.

Figure 1: Configuration for the "Diagonal wall" experiments.



The `src/` directory contains all of the java source files, including two which you will have to modify: `Demo.java` and `QlearningAgent.java`. To create your Sarsa agent:

(a) Copy the `QlearningAgent.java` file into a file called `SarsaAgent.java`.

(b) Rename the class to `SarsaAgent`.

(c) Alter the `public Integer step(Double R, Integer obs)` function to perform Sarsa updates instead of Q-learning updates. *Clearly comment the changes you make.*

(d) To choose which agent the demo uses, you need to alter `Demo.java` near line 34 by uncommenting the type of agent you wish to use and then re-compile the demo.

Once you have written your agent, you should run four different experiements:

(a) Empty grid with Q-learning. Output: `q_empty.txt`

(b) Empty grid with Sarsa. Output: `sarsa_empty.txt`

(c) "Diagonal wall" with Q-learning. Output: `q_wall.txt`

(d) "Diagonal wall" with Sarsa. Output: `sarsa_wall.txt`

For each of these, you should run the demo for a minimum of 5000 episodes if possible, and dump the results to the corresponding output file specified above. If, for any of these experiments, the episodes start taking a really long time before you get to 5000 and you are getting bored, you may stop the experiment early. Make a note in your submission of approximately how long the episodes are taking if this happens. (This may happen if the agent becomes very effective at avoiding the cat.) For the "Diagonal wall" experiments, before you begin, set up walls as shown in Figure 1.

The progam outputs four columns of data: The episode number, the total reward for that episode, the average total reward over the past 100 episodes, and the maximum reward over the past 100 episodes. Use the first column (episode number) for the x-axis values, and the *third* column (100-episode average)

for the y axis. (Don't worry about the other two columns.) Create two plots: one with the two `empty` datasets (i.e. Q and Sarsa) and one with the two `wall` datasets (again Q and Sarsa).

You can use whatever plotting software you like; however, the output of the program is formatted such that `gnuplot` is particularly easy to use. For example, to create the plot for the `empty` data, you can just type:

```
plot "q_empty.txt" using 1:3 with lines, plot "sarsa_empty.txt" using 1:3 with lines
```

at the `gnuplot>` prompt to generate the first plot, for example.

Include your `SarsaAgent.java` file, the four output files, and the two plots in the archive you are submitting for this assignment.

2. [*20 points*] **Bayes Classifiers**
Consider the following table of data:

A	B	C
1	1	1
1	0	1
1	1	1
1	0	1
0	0	1
0	0	1
0	1	0
0	1	0
1	1	0
0	0	0

Using these data,

(a) [*10 points*] Estimate all of the probabilities necessary to build a Bayes classifier for the class variable C by using a full **joint** model for $P(A, B|C)$.

(b) [*5 points*] Estimate all of the probabilities necessary to build a Bayes classifier for the class variable C by using a **naïve** model for $P(A, B|C)$.

(c) [*5 points*] Using the two classifiers you have constructed, fill in the following table:

A	B	$P(C = 1\|A, B)$ using Joint	$P(C = 1\|A, B)$ using Naïve
0	0		
1	0		
0	1		
1	1		

Write your answer in a text file called `bayes.txt` (or `.pdf` or `.ps` or `.html` or `.doc`) and include it in the archive you are submitting for this assignment.