CIS603 Project 2b: JiaoTong Initializer

Due: 25 Nov 2013

This is version 1.1 of this project description

1 Tentative Timeline for Course Projects

- *Project 2a* – (Due 16 Nov.) Compile and run code, and begin it modify it.
- *Project 2b* – (Due 26 Nov.) Implement and evaluate a simple model-based learner and a satisficing learner
- *Project 2c* – (Due 4 Dec.) Implement and evaluate a more complex model-based learner and an expert algorithm
- *Project 3* – (Due 18. Dec.) Implement and evaluate your own winning algorithm. Enter your algorithm in The JiaoTong Cup.

2 Deliverables for Project 2b

This project builds on Project 2a. In this project, you will code up two algorithms for making decisions in JiaoTong.

1. Code up two algorithms: ε-greedy and satisficing (aspiration) learning.
2. Evaluate the performance of the two algorithms in various scenarios (combinations and numbers of algorithms).
3. Produce a write-up evaluating the algorithms. You won't submit your code at this stage, just your write-up.

3 Code for Project 2b

You can build onto the same client you created in Project 2a. However, the server has been modified slightly, so please use a different version of the JiaoTongServer, which can be downloaded from the Moodle website. The following are a list of important changes for this version of the JiaoTongServer:

1. The JiaoTongServer now requires you to specify a reward file, which designates the payoff that each vehicle receives when it reaches each node. This reward file must be placed in the `games` directory.
2. You can now change the speed at which the game is run, which will be helpful for gathering results. This is done via a command-line argument.
3. The final results for a round are now logged in results.txt, which appears in the directory of the JiaoTongServer executable after the completion of the game. This file specifies the final score of each player in the game.

The new command-line arguments for running the JiaoTongServer are:

```
./JiaoTongServer theworld scenario1 [numVehicles] [gameLength] ffwd=[speed]
```

For example, if you want to run a 120 minute game with 35 vehicles at four times the speed, you would enter:

```
./JiaoTongServer the world scenario1 35 120 ffwd=4.0
```

Algorithm 1 A simple ε-greedy learner.

```
Initialize:
    t = 1
    $v(a) = u(a) = 100$ for each cycle $a$
    $\kappa(a) = 0$ for each cycle $a$
    $\varepsilon = \frac{1}{2+t/3}$
repeat
    Determine the current time $T_0$ (in seconds)
    Select a cycle $a_t$ using Eq. (1)
    Once the cycle is completed observe the current time $T_1$ and the total reward ($r_t$) obtained in the cycle
    $v(a_t) = v(a_t) + \frac{r_t}{T_1 - T_0}$
    $\kappa(a_t) = \kappa(a_t) + 1$
    $u(a_t) = \frac{v(a_t)}{\kappa(a_t)}$
    $t = t + 1$
    $\varepsilon = \frac{1}{2+t/3}$
until Game Over
```

4 Two decision-making algorithms

You will code up two different algorithms as part of Project 2b: a ε-greedy learner and a satisficing learner. Recall that in Project 2a, we simplified the game so that your agent could pick 1 of 5 cycles. You should use the same mechanism for each of these algorithms.

4.1 A simple ε-greedy learner

Pseudo-code for the ε-greedy learner is given in Algorithm 1. The algorithm simply computes from its experiences the expected value for selecting each cycle. Then, with probability $1 - \varepsilon$, it selects the cycle with the highest expected value. Otherwise, it selects a cycle randomly. Formally, let $u(a)$ be the expected reward for taking cycle a. Then, this algorithm selects a cycle as follows:

$$a_t \leftarrow \begin{cases} \arg\max_a u(a) & \text{with probability } 1 - \varepsilon \\ \text{select randomly} & \text{otherwise} \end{cases} \tag{1}$$

A couple of items to note:

- ε is decreased over time. You can experiment with different exploration rates if you desire.
- $u(a)$ is updated (in the fourth line of the `repeat` block with the *normalized reward.* Since the cycles are of different lengths, we must normalize the reward to reflect this.

4.2 A simple satisficing learning algorithm

Pseudo-code for the satisficing learner is given in Algorithm 2. This algorithm first enters an exploratory phase (for N cycles) to estimate the highest possible payoff. It then sets its initial aspiration α_t to this value and then executes the satisficing learning. Cycles are selected as follows:

$$a_t \leftarrow \begin{cases} a_{t-1} & \text{if } R_t \geq \alpha_t \\ \text{select randomly} & \text{otherwise} \end{cases} \tag{2}$$

A couple of notes to keep in mind:

- You might consider trying different learning rates $\lambda \in (0, 1)$.
- You might consider different lengths of the exploration phase (i.e., try different values of N)
- As with the ε-greedy learner, we use the *normalized reward.*

Algorithm 2 A simple satisficing learner.

```
**Initialize:**
  $Rmax = -99999$
  $\lambda = 0.9$
**for** t=1:N    // *exploration phase*
  Determine the current time $T_0$ (in seconds)
  Randomly pick a cycle $a_t$
  Once the cycle is completed observe the current time $T_1$ and the total reward ($r_t$) obtained in the cycle
  $R_t = \frac{r_t}{T_1 - T_0}$
  **if** $R_t > Rmax$ **then**
    $Rmax = R_t$
**end**
t = N
$\alpha_t = Rmax$
**repeat**
  Determine the current time $T_0$ (in seconds)
  Select a cycle $a_t$ using Eq. (2)
  Once the cycle is completed observe the current time $T_1$ and the total reward ($r_t$) obtained in the cycle
  $R_t = \frac{r_t}{T_1 - T_0}$
  $\alpha_{t+1} = \lambda \alpha_t + (1 - \lambda) R_t$
  $t = t + 1$
**until** Game Over
```

5 Evaluation and Write-up

After coding up each algorithm and making sure they work properly (probably by observing the performance of a single vehicle in JiaoTong), evaluate the algorithms in JiaoTong. To do this, you'll likely want to evaluate them in JiaoTong games with 35 or more vehicles.

Your job is to find out things that are interesting. Here are some questions you might consider answering (you can answer these and/or others – just make it interesting):

- How well does each algorithm do in self play (all vehicles use the same algorithm)?
- How well do the algorithms perform against each other (vary the percent of population using each algorithm)?
- What kinds of behavioral tendencies do each of the algorithms learn in the various circumstances?
- Experiment with different algorithmic parameters (and priors). Do these values impact the performance of these algorithms?
- What happens if you change the payoff structure of the game (values of the nodes)? Does that change the performances and behaviors of the algorithms. Note, you do this by specifying a different file from the command line other than `scenario1`.

You won't submit any code as part of this project. You'll only be evaluated by the write-up you submit. The more interesting the questions you try to answer and the correctness of your analysis, the better you'll do. But avoid being long-winded. Be concise but thorough.