RL Project Proposal

Airi Shimamura Khoi Trinh

[This proposal is very detailed and ambitious and is for a group of two people working together. Clearly, if you are working alone your goals will be half as much.]

[First I tell the reader what the heart of my project will be. More detail is given in later paragraphs.] For our RL project this semester; we want to build an RL agent that can easily beat the CartPole game.

[Talk about the problem domain in more detail.] CartPole is a game environment provided in the Gym package from OpenAI. In this environment, there is a pole, attached to a cart (hence the name CartPole). The cart moves along a frictionless track. Force is applied in the left and right direction of the cart. The goal of the game is to keep the pole upright for as long as possible. For each step taken, a +1 reward is given, including the termination step. The maximum points achievable in the game is 475. There are a few conditions that, if met, will end the game.

First, if the pole angle is greater than 12 degrees, the game ends.

Second, if the cart position is greater than 2.4 (or the center of the cart reaches either end of the display), the game ends.

Finally, if episode length is greater than 500, the game ends.

[After discussing the project domain, you should discuss your intended reinforcement learning approach. Give as much detail as you know (I realize you are still learning RL).] Amy will train her hearts player using TD learning. Anna will also use RL but she intends to use Q-learning. Both of us plan to explore rollouts as a method to improve player performance. We will reward our players by giving them the negative number of points won in each hand. Thus, each point taken is a punishment and the player should seek to minimize punishment. This will be tricky because shooting the moon should not be viewed as a punishment. To solve this, we will not train until the end of the game but will instead save the entire game in memory. Once the final point distribution is known, we will train using that game.

Since we will have two players, we intend to train them against each other at first. We will also experiment against heuristic players available in the simulator. As each player improves, we will enter them in the online competition and will save the training experience from those games. In addition, we will play games against the player ourselves and save that experience. Once the two players are working well together, we will focus on the combined player and expect that will be the best overall player.

Because hearts is a partially observable domain, the state representation will be tricky. We both propose to create feature vectors that summarize the cards in the player's hand as well as the important information about the past history (critical cards being played, such as the queen of spades, whether points have spread among the players yet, etc).

[Next discuss how you will evaluate your project, how you will know it is working well (or poorly), and how you will stop training.] We will evaluate our players in several ways. First, we will examine the number of matches won over time, as the player trains. A match generally goes to 250 or 500 points. Second, we will examine the average reward that the player is receiving as it trains. Because our players will be stochastic, we will measure the average reward over time. However, it is too computationally intensive to enter multiple agents in the online competition. We will wait to enter the agents into the online competition until we feel that they are competitive. Once an agent has entered, we will measure its progress by the nightly ladder performance. This progress will not be averaged. We hope that our player will move to the top 10% of all players and we will try to enter it into the competition by November so that it has time to move to the top of the ladder.

[Other details such as intended use of shared code, etc.] In order to compete in the online competition, we will make use of the generic java hearts player code available at *URL*. We will place all of our code in separate files (except the call to the original code) and note any changes that we make to the downloaded code.

[Your written proposal must be no longer than one and a half column if you use the style that we have for typesetting. If you use a single-column style for the writeup, then the proposal must be no longer than one and a half page.]