CSCI 446 Artificial Intelligence Project 4 Design Report

Roy Smart Nevin Leh Brian Marsh

December 2, 2016

1 Introduction

Within the field of machine learning, there is a type of unsupervised learning known as reinforcement learning. An agent using reinforcement learning makes actions in its environment and observes the rewards that it obtains from such actions. Using many observations of the environment, the agent may be trained to maximize its cumulative rewards through the problem that it attempts to solve.

2 The Racetrack Problem

Using reinforcement learning, we will attempt to solve the racetrack problem. The goal of this problem is simply to control the movement of a race car as it moves along a track. Performance is based upon the number of time steps that the car requires to reach the finish. If the car moves into a wall, a penalty will be applied, thus decreasing performance. The car is represented by four variables at any given state: $x(t)$ and $y(t)$ (horizontal and vertical components of the car's location at time t) and $\dot{x}(t)$ and $\dot{y}(t)$ (horizontal and vertical components of the car's velocity at time t). Control of the car can only be achieved by manipulating the velocity of the car, using the variables a_x and a_y (horizontal and vertical components of the acceleration vector). The problem is complicated by the addition of a 20% chance of failure for any acceleration action. Additionally, the agent will be tested on multiple tracks of varying shape.

3 Reinforcement Learning Algorithms

3.1 Value Iteration

4 Q-Learning

4.1 Description

Q-learning is a model-free reinforcement learning method for determining optimal action-selection policies [Russell and Norvig, 2010]. An agent using this method leverages a quantity known as the Q-value to derive the optimal action a for each state s. The Q-value, $Q(s, a)$ describes the expected utility for every action in every state within the environment and is learned by the agent using temporal difference learning. An expression to calculate the Q-value is given by [Russell and Norvig, 2010] as

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where a is the action that was executed in state s that resulted in state s' and $R(s)$ is the reward function. The constants α and γ are known respectively as the learning rate and the discount factor. Equation 1 is used as an update rule to adjust the value of $Q(s, a)$ for each action-state pair in every time trial undertaken

by the agent. Using this simple update rule and randomly initialized Q-values for every action-state, an agent can learn how to navigate an environment.

4.2 Design Implications

In the racetrack problem Q-Learning will need to learn the Q-value for all nine acceleration options for every possible velocity vector at every possible position vector. Since the racetracks are reasonably small, we can afford to sore the table of Q-values in memory. We will base our Q-learning agent off of the function Q-Learning-Agent provided in Figure 21.8 of [Russell and Norvig, 2010]. To ease the training time, we will incrementally train the Q-learning agent by beginning training close to the finish line, and then increasing the distance as the agent learns each section.

5 Software Design

For this project we will be using an environment and agent model similar to the wumpus world. Instead of a board we will have a `Track` class that reads in the different tracks from a text file. To run the experiments we will have an `Environment Engine` for each `Track` and `Agent` combo.

The `Agent` will be a virtual class that contains data such as the max acceleration values. It will also contain the virtual method `learn`. Each concrete class will be the implementation of our reinforcement learners. Each of these classes will contain an overridden learn method and algorithm specific functionality. The key difference between the wumpus world `Agent` and this `Agent` is the fact that in this case the `Agent` can see the whole board from the start.

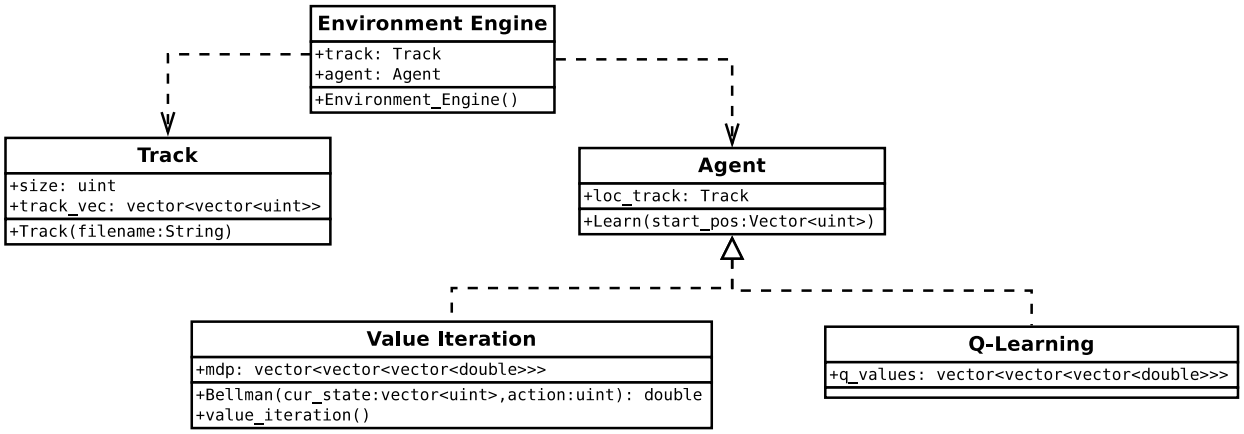

Figure 1: *UML of our design*

6 Experiment Design

For each agent we will generate learning curves that describe the number of steps required to reach the goal state vs. the number of training iterations. We will also track the number of times each agent runs into a wall and needs to be restarted. Each agent will be trained at least 10 times to determine reliable learning curves.

7 Summary

References

[Russell and Norvig, 2010] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey 07458, 3rd edition.