Reinforcement Learning Project Report

Unsupervised and Reinforcement Learning in Neural Networks

Han JU, Hao REN

18 décembre 2012

1 Introduction

In this project we implement a reinforcement learning experiment with continuous state space and a neural network model. Unlike the discrete state learning experiments, in a continuous state space we can't enumerate over all possible states, that makes the classic Q-learning algorithm not applicable here. Instead, with the help of a neural network model taught in the course, we are able to code the state space by a finite number of input neurons, namely the place cells. On the other side, we have the action cells, each represents an possible action of the agent in the experiment. Given this set up, the expected reward for a particular point in the state space is just the dot product of the point's activation of all place cells and the weight vector between the place cells and the action cells. So essentially we use the classic SARSA algorithm's framework, but what we update is the weight vector, and the Q-values are computed dynamically for choosing an action.

For the implementation, we adapt it from the given python code in the exercise session.

2 Learning curve

Right after the implementation according to the project set-up, we begin our experiment by conducting 10 independent runs with 50 trials each. Figure 1 is the resulting learning curve.

For this very first experiment, we set the epsilon parameter to 0.5, which means half of agent's moves are random. The leaning curve makes perfect sense : the first several runs are nearly random walks, which take over 5000 steps thanks to the randomness and the yet not very effective W-values. However, after several tastes of the goal, due to the effect of the eligibility trace, this information is quickly propagated throughout the weight vectors between the input space cells and the action neurons. As a result, half of the time the agent chooses the right direction that leads it to the goal area and avoids the wall hitting, the steps needed are decreasing drastically. We can observe that within 10 trials, this number goes down from 8000 to a stable value of aprroximately 100. Note that because of the randomness, there is some fluctuation on the curve.

As for the average reward, showed in red curve in the figure, it has generally the same trend as the latency curve. For example in the first several trials, the agent is essentially exploring the state space, so it hits often the wall and that causes a very low overall reward. As the agent's knowledge of the state space grows, its reward per trial converges to 10, which means it pursues directly the goal area without hitting too much the wall.

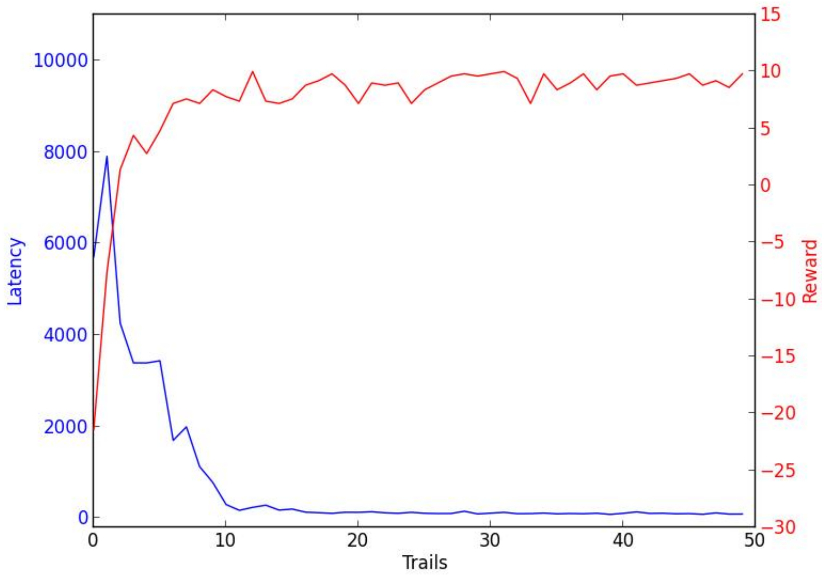

Figure 1 – Learning curve of a complete experiment

3 Exploration & Exploitation

4 Navigation Map