CMPSCI 689: Mini Project 3

Due: December 7th 2015

Abstract

This is the third and last mini project, and covers the material on deep learning and reinforcement learning. Read the entire document before starting the project. Your solutions must be computer formatted and submitted online on Moodle by **December 7 2015**. There will be no extensions of this deadline. Each student must turn in an individual solution based on their own work, with no joint work. **Do not wait until the last weekend to work on this project. You will not be able to complete it!**

Question 1 (25 points)

Consider a Markov decision process (MDP) defined by $M = (S, A, P, R, \gamma)$, where S is a finite set of states, A is a set of actions, P is the transition matrix where $P(s'|s,a)$ gives the distribution of future states s' conditioned on doing action a in state s, and R is the expected reward where $R(s,a)$ is the reward for doing action a in state s.

part a: (10 points) Assume that $\pi : S \to A$ is a fixed policy mapping states to actions for which we desire to compute its associated value function V^π. Bellman's equation tells us that V^π is the fixed point of the Bellman operator

$$T^\pi(V) = R^\pi + \gamma P^\pi V$$

so that $T(V^\pi) = V^\pi$. Here, $R^\pi = R(s, \pi(s))$ and $P^\pi = P(s'|s, \pi(s))$. However, when S is very large, then it is preferable to compute an approximation $\hat{V}^\pi = \Phi w$, where Φ is a *basis matrix* where each column is a basis function (e.g., radial basis function, polynomial, or some equivalent approximator). We can try to find the weights w for the *composition* of the Bellman operator T^π and the orthogonal projector Π_Φ onto the column space of the basis matrix Φ. In particular, show that the solution to the following minimization problem

$$w_{FP} = \text{argmin}_w \|\Pi_\Phi T^\pi(V) - V\|$$

is given by the weights $w_{FP} = (\Phi^T(I - \gamma P^\pi)\Phi)^{-1}\Phi^T R^\pi$.

part b: (15 points) Rather than computing the fixed point of the composition of the orthogonal projector and the Bellman operator, a simpler approach is to solve the fixed point equation $T(\Phi w) = \Phi w$ in the least-squares sense. Show that the least-squares projection is defined by $\hat{V} = \Phi w_{LS}$ where

$$w_{LS} = (\Phi^T(I - \gamma P^\pi)^T(I - \gamma P^\pi)\Phi)^{-1}\Phi^T(I - \gamma P^\pi)^T R^\pi$$

Question 2 (25 points)

The goal of this question is to read the paper `Playing Atari with Deep Reinforcement Learning` (download it from `http://arxiv.org/pdf/1312.5602v1.pdf`) and answer the following questions.

Part a (10 points): Derive the gradient based Q-learning algorithm given in the paper as Equation 3.

Part b (5 points): What is the difference between the deep learning approach to Atari game playing and the earlier feedforward neural net Q-learning technique for playing backgammon?

Part c (5 points): Why does Deep Mind use experience replay in their Atari game learner?

Part d (5 points): Comment on the graphs shown in Figure 2. What does each curve show, and are there differences in the performance on Breakout vs. SeaQuest?

Question 3: Programming Project (50 points)

This question asks you to test a deep reinforcement learner that trains a simple 2D robot on a spatial task that requires navigating through the environment and picking up rewarding objects. The code is written entirely in Javascript and can be run from a browser. You can run the code from the web site `http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html`. The goal of this task is to experiment with various parameters that specify the architecture of the deep learning as well as the learning algorithm used (Q-learning). The parameters for this task are given as:

```
var tdtrainer_options = {learning_rate:0.001, momentum:0.0, batch_size:64, l2_decay:0.01};
opt.experience_size = 30000;
opt.start_learn_threshold = 1000;
opt.gamma = 0.7;
opt.learning_steps_total = 200000;
opt.learning_steps_burnin = 3000;
opt.epsilon_min = 0.05;
opt.epsilon_test_time = 0.05;
```

Part b (10 points): Vary the learning rate and momentum and plot the the improvement or lack of improvement as a function of discount factor. You can measure performance using the average reward.

Part a (10 points): Vary the discount factor γ from 0.7 to 0.99, and measure the improvement or lack of improvement as a function of discount factor. You can measure performance using the average reward.

Part b (10 points): Vary the experience size, and measure the improvement or lack of improvement as a function of discount factor. You can measure performance using the average reward.

Part c (10 points): Vary the experience size, and measure the improvement or lack of improvement as a function of discount factor. You can measure performance using the average reward.

Part d (10 points): Show examples of learned weights from your trained neural network.