Attempts to RL an introduction

Attempts to non-programming exercises in RL an introduction

Chapter 1

Introduction

1.1 An Extended Example: Tic-Tac-Toc

Exercise 1.1:

Both sides are trying to win and are learning optimal strategies. They are going to learn a policy similar to that of learnt by minmax search. Hence they would not learn different policy for selecting moves (from each other).

Exercise 1.2:

Amend the learning process by assigning same value and value updates to symmetrically equivalent states. This way we can speed up learning process and converge to equilibrium faster (if exists).

Yes, we should still take advantage of symmetry. Note that symmetrically equivalent *positions* do not necessarily have same values, but symmetrically *states* do.

Exercise 1.3:

The greedy player, on average, and in the long run, does not perform as well as a reasonable nongreedy player. In some cases greedy player may still outperform non-greedy players. However, it is certainly not the best player in general.

The program is greedy player may stuck in local optimals and fail to learn a better policy because it does not explore at all.

Exercise 1.4:

When we learn from explorations, the value of each state is taking an weighted average with the original values and the mean of expected value of all possible states. Every value is closer to the local mean of the possible states.

When we do not learn updates when doing explorations but keep doing explorations, the value of each state is converging to a local optimal estimate of probability of winning from that state. We call the set of learnt probabilities 'original values' in the previous paragraph.

The set with learning from explorations is a better set is learn assuming we can run enough number of runs. It will result in more wins. The reason is that the weighted average does not change the ranking of values of next possible states given a current state. In addition to that, the algorithm is less likely to stuck in a local optimum, due to additional randomness added to the learning process. It should explore a larger space before converges.

Exercise 1.5:

We may initialize the inital values uniformly randomly except those of value 0 and 1. We repeat the experiments multiple runs and may search a larger space. Moreover, each opponent may have different levels of 'tendency to variate', we may devise a technique for the RL to adjust its step-size parameter based on variability of opponent's play rather than hand specify the step sizes. Finally, the training an agent against an opponent that never changes is of little interests. We should train an agent to play against any player. We can train a group of RL agents of different priors and let them play against each other. Because we initialized the states

differently, they may learn different values of states. If they converge to learn the same set of values, this set is more likely to be the global estimate.

Chapter 2

Multi-armed Bandits

2.1 Action-value Methods

Exercise 2.1:

$$P(greedy) = 0.5 + 0.5 * 1/2 = 0.75$$

Exercise 2.2:

In step 4 and 5, ϵ definately occured. And ϵ may occur at any time.

Exercise 2.3:

In the long run, cumulative reward and probability of selecting the best action should be dominated by the average converged values.

In figure 2.2 in the book, ϵ-greedy ($\epsilon = 0.1$) performed the best. Assuming the step 1000 represents the local equilibiurm that each algorithm is stuck with, ϵ-greedy ($\epsilon = 0.1$) is about 8% better than ϵ-greedy ($\epsilon = 0.01$) and 60% better than greedy in terms of rewards. ϵ-greedy ($\epsilon = 0.1$) is about 50% better than ϵ-greedy ($\epsilon = 0.01$) and 250% better than greedy in terms of the probability of selecting the best action.

Exercise 2.4:

$$Q_{n+1} = \prod_{i=1}^{n}(1-\alpha_i)Q_1 + \sum_{j=1}^{n}\alpha_j R_j \prod_{k=j+1}^{n}(1-\alpha_k)$$