Attempts to RL an introduction

Attempts to non-programming exercises in RL an introduction

Chapter 1

Introduction

1.1 Exercise

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).

1.2 Exercise

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.

1.3 Exercise

The greedy player, on average, and in the long run, does not perform as well as a reasonable non greedy 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.

1.4 Exercise

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.

1.5 Exercise

We may initialize the initial 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 Exercise

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

2.2 Exercise

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

2.3 Exercise

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.

2.4 Exercise

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

2.5 Exercise

See https://da-niao-dan.github.io/RL/RLIntroductions/Exercise2pt5/Exercise2pt5.html

2.6 Exercise

The optimistic initial values drive the algorithm to explore every options in the early steps, even if the algorithm is greedy. In particular, a large number of individual runs of algorithms may choose the same action to perform early on. When many of them choose the optimal choice, the spikes in the graph emerges, and when many of them move on to other choices due to optimistic initial values, the drop may be steep. Therefore the graph oscillates more in early steps.

2.7 Exercise

First, claim that $o_n = 1 - (1-\alpha)^n$. Note that

$$o_0 = 1 - (1-\alpha)^0,$$

and that

$$\begin{aligned} o_{n+1} &= o_n + \alpha(1 - o_n) && \text{by definition} \\ &= 1 - (1-\alpha)^n + \alpha - \alpha(1-(1-\alpha)^n) && \text{by induction hypothesis} \\ &= 1 - (1-\alpha)^n(1-\alpha) \\ &= 1 - (1-\alpha)^{n+1}. \end{aligned} \tag{2.1}$$

Hence the claim is true by induction for all $n \in \mathbb{N}$. And in addition, we have

$$\beta_n = \frac{\alpha}{1-(1-\alpha)^n}.$$

Now,

$$\begin{aligned} Q_{n+1} &= Q_n + \beta_n(R_n - Q_n) \\ &= (1-\beta_n)Q_n + R_n \\ &= \beta_n R_n + (1-\beta_n)(Q_{n-1} + \beta_{n-1}[R_{n-1} - Q_{n-1}])) \\ &= ... \\ &= \sum_{i=0}^{n-1} \beta_{n-i} R_{n-i} \prod_{j=0}^{i}(1-\beta_{n-j+1}) + \prod_{i=1}^{n}(1-\beta_i)Q_1 \qquad \beta_1 = 1, 1-\beta_1 = 0 \\ &= \sum_{i=0}^{n-1} \beta_{n-i} R_{n-i} \prod_{j=0}^{i}(1-\beta_{n-j+1}) \qquad \text{indepdent of } Q_1 \end{aligned} \tag{2.2}$$

Moreover, $\beta_n = \frac{\alpha}{1-(1-\alpha)^n} \downarrow \alpha$ as $n \to \infty$. Hence as $n \to \infty$, all weights given to original R_i will decay exponentially, by a factor of $(1-\alpha)$ each step. Hence we have an exponential recency-weighted average without initial bias.

2.8 Exercise

For step 1 to 10, all runs of the algorithm will explore 10 actions one by one, due to the fact that "an action is considered optimal if $N(\alpha) = 0$. On the step 11, all runs of this algorithm will choose the optimal step. In this case, the action reward goes to the top.

However, on step 12, we visited optimal action twice but all others only once. So the variances of the other action values are higher than the optimal. And for this reason in some runs the upper confidence for some sub-optimal actions turn out to be higher (for larger c, this effect strengthens). Hence the average rewards drops on the subsequent step due to choices of sub-optimal actions by large variance in confidence level.

2.9 Exercise

Suppose we have two actions: a and b with preferences for them $H(a)$ and $H(b)$. Let A denote action random variable.

The softmax distribution is given by:

$$Pr\{A = a\} = \frac{e^{H(a)}}{e^{H(a)} + e^{H(b)}} = \frac{1}{1 + e^{H(b)-H(a)}} = sigmoid(H(b) - H(a)),$$

and $Pr\{A = b\} = 1 - Pr\{A = a\}$, so in case of two actions, distributions given by softmax and sigmoid are the same. To see this more clearly, and without loss of generality, set one of $H(a)$ or $H(b)$ to be 0.