

Notes

January 20, 2019

A collection of notes derived from the book, *Reinforcement Learning* by Richard Sutton and Andrew Barto. Available at <http://incompleteideas.net/book/the-book.html>

1 Action selection

Most RL methods require some form of policy or action-value based action selection method.

- **Greedy Selection:** Choosing the best action.

$$A = \operatorname{argmax}_a Q(a)$$

- **ϵ -greedy Selection:** Simple exploration with ϵ -probability.

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1 - \epsilon \text{ (breaking ties randomly)} \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

- **Upper Confidence Bound (UCB):** Takes into account the proximity of the estimate to being maximal and the uncertainty in the estimates. Does not perform well on large state spaces.

$$A_t = \operatorname{argmax}_a \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Where:

- $c > 0$ is the degree of exploration
- $N_t(a)$ is the number of times that action a has been selected prior to time t .
If $N_t(a) = 0$, then a is considered to be a maximizing action.

2 Performance Measures

Methods for comparing the performance of different parameters and algorithms.

- **Optimal Action %:** Requires knowledge of the workings of the environment and whether the action was optimal. Plot % over steps.

- **Average reward:** Simply plot the average reward over steps. Good for comparing specific implementation of agent in specific implementation of environment.
- **Average reward w.r.t. parameter:** Plot the average reward over first $n=1000$ steps against input parameter(s) (ε , α , c , Q_0 etc) on a logarithmic scale. Good for comparing learning algorithms' general effectiveness and finding the best parameter value.
- **Mean Square Error:** Plot the mean square error (averaged over $n=100$ runs) of the value of a single state (error = actual-estimate) over the number of episodes run before achieving the estimate, with the episodes on a logarithmic scale. Good for Monte Carlo method, where you can form an estimate of a single state without forming an estimate of the others.

3 Algorithms

Reinforcement learning algorithms covered by the book.

- **Dynamic Programming / Value Iteration:** Updating state values by sweeping through all states. Computationally expensive, especially on large state spaces.

```

1 Parameters:
2   a small threshold  $\theta > 0$ 
3   Initialize  $V(s) \forall s \in S^+$  arbitrarily, except that
4    $V(\text{terminal}) = 0$ 
5
6 Loop:
7    $\Delta \leftarrow 0$ 
8   Loop for each  $s \in S^+$ :
9      $v \leftarrow V(s)$ 
10     $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$ 
11     $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
12 until  $\Delta < \theta$ 
13
14 Output a deterministic policy  $\pi$ , such that
15  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$ 

```

- **Off-policy Monte Carlo control:** Using a soft policy to explore, while estimating the optimal policy. Does not bootstrap, i.e. doesn't use estimates of previous states to estimate the current state value.

```

1 Parameters:
2   For all  $s \in S, a \in A(s)$ :
3     Initialize  $Q(s,a)$  arbitrarily
4      $C(s,a) \leftarrow 0$ 
5      $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$  (with ties broken consistently)
6 Loop for each episode:

```

```

7   $b \leftarrow$  any soft policy
8  Generate an episode using  $b: S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$ 
9   $G \leftarrow 0$ 
10  $W \leftarrow 1$ 
11 Loop for each step of episode,  $t = T-1, T-2, \dots, 0$ :
12    $G \leftarrow \gamma G + R_{t+1}$ 
13    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$ 
14    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$ 
15    $\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$  (with ties broken consistently)
16   If  $A_t \neq \pi(S_t)$ , exit inner Loop (proceed to next episode)
17    $W \leftarrow W \frac{1}{b(A_t|S_t)}$ 

```