Week 1

Sunday, August 25, 2019 8:51 AM

Reinforcement Learning is a subfield under machine learning. In RL, the agent collects it's own data by trial and error. Every time, an agent is asked to make a decision, it is given a few options. The agent gets a reward for selecting an option. The agent's job is to maximize the reward for it's actions.

If the reward for actions was known, the problem of RL would be trivial. The agent can greedily select the actions with the highest reward. However, the reward most often is not known. What we know are the estimates of these rewards. The estimates are known as action values.

Theoratically, the expected reward is given as:-

$$q^*(a) \doteq \mathbb{E}[R_t \mid A_t = a] \quad \forall a \in \{1, \ldots, k\}$$

ie the expected reward is the summation of expected rewards received if an action a is taken at times 1 .... k.

This can be found by

$$q^*(a) = p(r|a) \cdot r$$

Again the goal is to maximize this expected reward which can also be represented as

$\arg\max_a q_*(a)$ ie the selection of actions yielding max expected reward.

In practical terms, we can find this by doing an experiment multiple times and plotting the results or using other statistical methods to

an experiment multiple times and plotting the results or using other statistical methods to find the kind of distribution. Then just find the expected value for that distribution.

↓ mean of the distribution

This problem is called the problem of bandits. It is a smaller problem under RL and is a good intro to the field of RL.

Method of Estimating $q^*(a)$ [Sample Average Method]

One method of estimating $q^*(a)$ is called the sample average method. In the sample average method, we estimate the action value using previous values.

$$q^*(a)_n = \frac{a_1 + a_2 + \cdots a_{n-1}}{n-1}$$

$$= \frac{\sum_{i=1}^{n-1} a_i}{n-1}$$

It's $n-1$ because the value is based on previous values.

This method has a problem that in the above form, the time and space complexity grows by $O(n)$. We can make this better using previous values as follows :-

$$q^*(a)_{n+1} = \frac{\sum_{i=1}^{n} a_i}{n}$$

$$= \frac{1}{n}\left[a_n + \sum_{i=1}^{n-1} a_i\right]$$

$q^*(a)$

$$= \frac{1}{n}\left[a_n + \sum_{i=1}^{n} a_i\right]$$

$$= \frac{1}{n}\left[a_n + \frac{(n-1)}{(n-1)} \sum_{i=1}^{n-1} a_i\right] \quad \rightarrow q^*(a)_n$$

$$= \frac{1}{n}\left[a_n + (n-1)\, q^*(a)_n\right]$$

$$= \frac{1}{n}\left[a_n + n\, q^*(a)_n - q^*(a)_n\right]$$

$$\boxed{q^*(a)_{n+1} = q^*(a)_n + \frac{1}{n}\left[a_n - q^*(a)_n\right]}$$

This is a very common form in RL. It can also be written as

New Estimate = Old Estimate + StepSize [Target − Old Estimate]

or New Estimate = Old Estimate + Step Size × Error

The $\frac{1}{n}$ term is called the step size.

The choice of $\frac{1}{n}$ is varied according to the problem.

If we do not have a stationary RL problem, ie the distribution of action values varies with time, we use a constant step size b/w $(0, 1]$.
What this does is that it weighs the newer values more in the calculation of action value estimates.
This can be seen as follows :-

$$q^*(a)_{n+1} = q^*(a)_n + \alpha\left[a_n - q^*(a)_n\right] \quad \leftarrow \text{constant step size}$$

$$= q^*(a)_n + \alpha a_n - \alpha q^*(a)_n$$

$$= q^*(a)_n + \alpha a_n - \alpha q^*(a)_n$$
$$= \alpha a_n + (1-\alpha) q^* a_n \quad \text{---- (1)}$$
$$= \alpha a_n + (1-\alpha)\left[\alpha a_{n-1} + (1-\alpha) q^* a_{n-1}\right] \quad [\text{using (1)}]$$
$$= \alpha a_n + (1-\alpha)\alpha a_{n-1} + (1-\alpha)^2 q^* a_{n-1}$$
$$= \alpha a_n + (1-\alpha)\alpha a_{n-1} + (1-\alpha)^2\left[\alpha a_{n-2} + (1-\alpha) q^* a_{n-2}\right]$$
$$= \alpha a_n + (1-\alpha)\alpha a_{n-1} + (1-\alpha)^2 \alpha a_{n-2} + (1-\alpha)^3 q^* a_{n-2}$$

$$= \alpha a_n + (1-\alpha)\alpha a_{n-1} + (1-\alpha)^2 \alpha a_{n-2} + \cdots + \cdots + (1-\alpha)^{n-2}\alpha a_2 + (1-\alpha)^{n-1}\alpha a_1 + (1-\alpha)^n q^*(a),$$

$$= \boxed{(1-\alpha)^n q^*(a) + \sum_{i=1}^{n}\left[\alpha (1-\alpha)^{n-i} a_i\right]}$$

From this $=n$, we can see that as we go closer to the current value, the weightage goes higher. This is because $(1-\alpha) < 1$ and greater the value of i, smaller is $n-1$ and higher will be $(1-\alpha)^{n-1}$.

We also see that $q^*(a)$, ie the initial estimate influences the estimate hugely.
This can be used to influence exploration or exploitation as will be seen later.

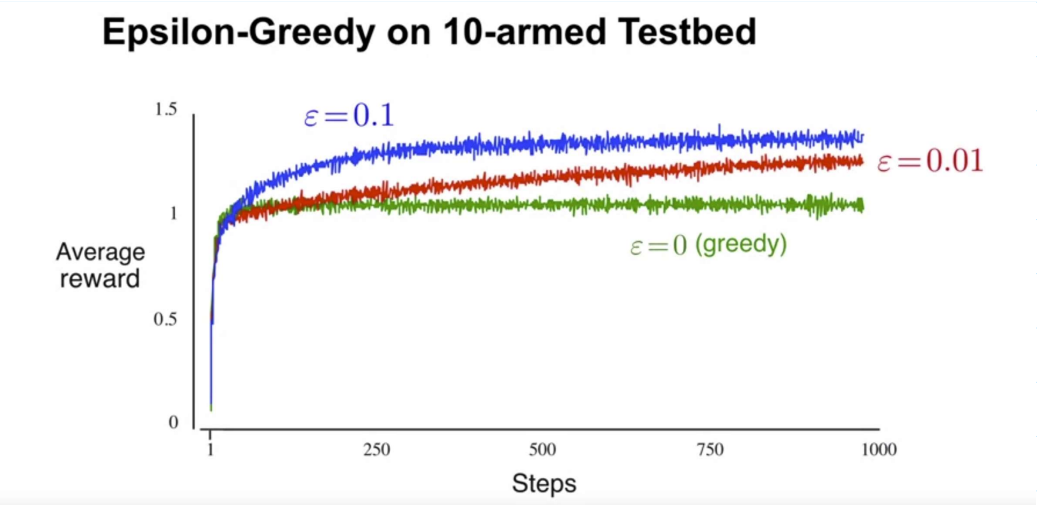
Exploration - Exploitation Dilemma

When initially an experiment is started, the agent has no knowledge of the estimates for actions. The agent can explore initially to get some estimates. After some time, it has to now decide if it should explore more or exploit the knowledge it has acquired. The agent can exploit its current knowledge by greedily selecting the estimates with the highest expected values. It may learn better actions if it explores more

estimates with the highest expected values". It may learn better actions if it explores more or it may not.

One way to approach this problem is to behave greedily most of the time and sometimes decide to explore. The agent can explore with a probability ε and exploit with a probability $1-\varepsilon$. This method is known as ε greedy method of exploration.

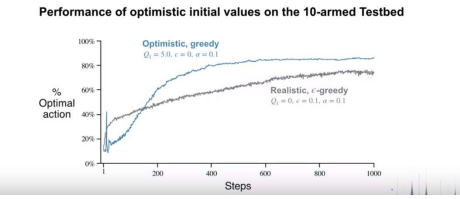
The textbook shows the following results for this strategy as compared to the strategy with $\varepsilon = 0$:-



Optimistic Initial Values

Optimistic initial action values makes the agent to explore more in the initial part of training. As we learnt earlier that if we set the step size to be constant, the initial value influences the estimated action values. If we set the initial value to be higher than what it can achieve, it will make the agent to explore different options as each time an action will be selected, the agent will be disappointed and this will make it to switch to other actions. Thus setting

the agent will be disappointed and this will make it to switch to other options. Thus setting higher initial value estimates will make the agent explore more initially. The results of this method on the 10 arm testbed problem as given in the textbook are as follows:-



This method although has a few limitations:-

1. The exploration only occurs initially and vanishes with time.
2. The method doesn't work with non-stationary problems.
3. We may not know what a high value of reward is.

Upper Confidence Bounds [UCB]

In ε greedy, we explore ε % of time and we exploit $(1-\varepsilon)$ % of time. When we explore we randomly explore any of the available actions. This can be improved, if we explore on the options which we expect to be better.

We do so using upper confidence intervals. The principle behind UCB is that in the face of uncertainty, we assume that we will get better results by trying it. Either we will get better results or our estimates will get better.

The estimate using UCB is given as:-

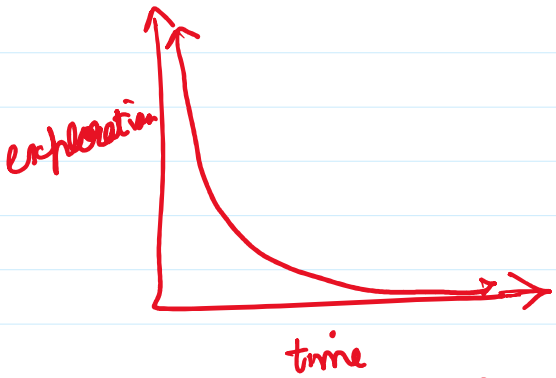
The estimate using UCB is given as :-

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

($Q_t(a)$: exploitation; $\sqrt{\frac{\ln t}{N_t(a)}}$: exploration)

The $c \sqrt{\frac{\ln t}{N_t(a)}}$ is called the exploration term.

The exploration for items which don't yield favourable results fade favourably as follows :-



If we take actions 1000 times and that action is selected 100 times, we get the exploration term as $c \sqrt{\frac{\ln(1000)}{100}}$ which will clearly be smaller than if that action was selected 1000 times. Thus with the passage of time, the actions getting selected more will start getting higher expected values.

The graph below shows the results of epsilon greedy compared with UCB method :-

Performance of optimistic initial values on the 10-armed Testbed