Reinforcement Learning Notes

1. *k*-armed Bandits

1.1. The k-armed Bandit Problem

Setup

- K different actions to choose from at each time step
- $p(\text{reward}|\text{action})$ is a stationary (not changing over time) probability distribution
- Goal of the agent is to maximize the cumulative reward (over some time span)

What is difficult about solving this problem?

- Do not know what conditional distributions (over reward) are, which means that we need to
- Estimate the expected values of underlying distributions using observed data
- Given that our estimate is never perfect, an important question arises, that is, under what circumstances should we exploit / explore?

1.2. Action Value Methods

Perfect Information (Full Knowledge of Underlying Reward Distributions)

Notation:

- A_t: action (out of k actions) selected at time step t
- R_t: reward received at time step t (after an action has been taken in that time step) (a random variable)

Since we are first dealing with stationary reward distributions, values of actions do not depend on time step. Therefore, the t subscripts merely serve to emphasize that A and R happens in the same time step. In other words, A caused the agent to receive R.

Expected reward given an arbitrary action a:

$$q_\star(a) = \mathbb{E}[R_t|A_t = a]$$

Obviously, the action that should be chosen at each and every time step is $\text{argmax}_a q_\star(a)$.

Imperfect Information (Can Only Sample From Underlying Reward Distributions)

Since we do not know the true expected rewards (for all actions) but can only sample from their corresponding distributions, we need a way to estimate the expected rewards using the finite samples.

This is intuitively done using the **sample-average method**:

$$\begin{aligned} Q_t(a) &= \frac{\text{sum of rewards when a was take prior to t}}{\text{number of times a was taken prior to t}} \\ &= \frac{\sum_{i=1}^{t-1} R_i \mathbb{I}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i=a}} \end{aligned}$$

Different algorithms use the sample-average rewards in different ways:

- Greedy method
  - Initialization: randomly choose between actions
  - Action selection: $A_t = \text{argmax}_a Q_t(a)$ (never explores)
  - Disadvantage: It is very likely that the true expected rewards of other non-greedy actions are higher, but this algorithm would never know because it never explore.
- ϵ-greedy method
  - Initialization: randomly choose between actions
  - Action selection:
    * Behave greedily most of the time: $A_t = \text{argmax}_a Q_t(a)$
    * Once in a while, with small probability ϵ, select randomly from all other action
  - Advantage: In the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that $Q_t(a)$ converges to $q_\star(a)$.
  - Disadvantage: Such asymptotic guarantees say little about its practical effectiveness.
- Optimistic initial values method
  - Initialization: set the action values to very high (and non-realistic) values, randomly choose between actions
  - Action selection:
    * $A_t = \text{argmax}_a Q_t(a)$, but explores much more than greedy method without optimistic initial values and here's why:
    * For example, let's say that $q_\star(a)$'s in a multi-armed bandit problem are selected from a normal distribution with mean 0 and variance 1. In this case, an initial estimate of +5 is very optimistic for all $q_\star(a)$'s.
    * Whichever action is initially selected, the reward is less than the starting estimate and the estimate is decreased; the learner therefore switches to other actions.
    * All actions are explored several times before the value estimates $q_\star(a)$ converge.
- Upper confidence bound (UCB) method
  - Initialization: randomly choose between actions
  - Action selection:
    * $A_t = \text{argmax}_a [Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$
    * Motivation:
      · ϵ-greedy forces non-greedy actions to be tried, but indiscriminately.
      · It would be better to select among non-greedy actions
        1. according to their potential for being actually optimal ($Q_t(a)$)
        2. take into account uncertainties in their value estimates ($\sqrt{\frac{\ln t}{N_t(a)}}$)
  - Disadvantage: hard to generalize to other reinforcement learning tasks (we do not know why yet)

1.3. The 10-armed Testbed

The 10-armed testbed consists of 1000 tasks similar to the one shown in Fig. 1.

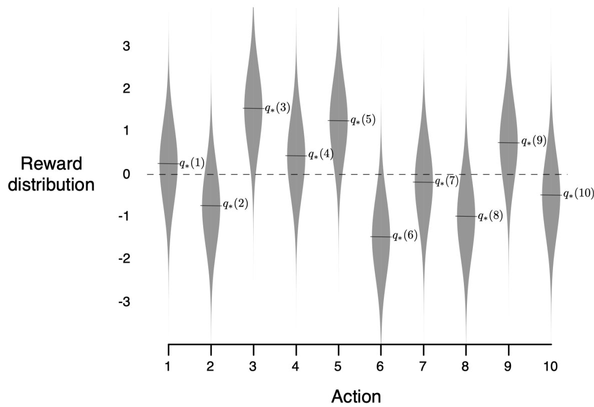

Figure 1: One task from the 10-armed testbed.

In Fig. 2, we can see that the highest average reward over first 1000 steps is achieved by UCB.

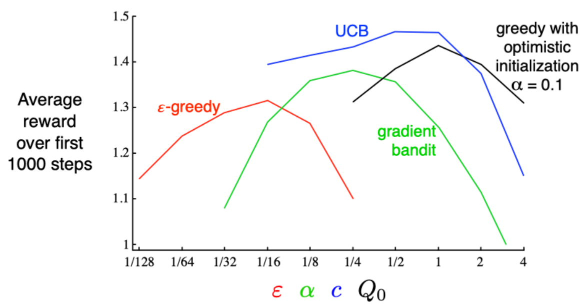

Figure 2: Performance of various algorithms on the 10-armed testbed.

1.4. Compute Sample-Averages Incrementally

If we focus on one action a_k, let R_i denote the reward received by the agent after the ith selection of this action and let Q_n denote the estimated reward:

$$Q_n = \frac{R_1 + \cdots + R_{n-1}}{n-1}$$

Native implementation: Keep a list of R_i's and compute the sum every time.
Incremental implementation: One can show that Q_{n+1} can be conveniently expressed as a function of Q_n:

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

where R_n is a new reward received as a consequence of a_k.

1.5. Tracking Non-stationary Problems

In this case, it makes sense to give more weight to recent rewards than to long-past rewards. One of the most popular ways of doing this is called **exponential-recency-weighted average** (still a way to estimate the value function).

The incremental update rule is modified to:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

where $\alpha \in (0, 1]$ and a common choice of α is 0.99.

To understand this update rule more intuitively, let's expand it. When expanded, this rule becomes:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

We can see, as i decreases, the weight given to the ith reward decreases exponentially.

1.6. Gradient Bandit Algorithms

In previous sections we've estimated the value of each action and used those estimates to take semi-informed actions. Decision making based on value estimates can work quite well, but it is not the only selection method available. One alternative is learn a preference for each action denoted $H_t(a)$ and makes decisions based on the relative preference of one action over another. Here we will use the *soft-max distribution* to define preference as a distribution over all possible actions.

$$Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a) \tag{1}$$

In this case $\pi_t(a)$ is the probability of taking action a at time t. All preferences are initially set to be the same.

Now we will begin to introduce an algorithm for this setting based on stochastic gradient descent using the following pair of formulas:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), \tag{2}$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha(R_t - \bar{R}_t)\pi_t(a), \forall\, a \neq A_t \tag{3}$$

Where $\alpha > 0$ is the step size parameter and $\bar{R}_t$ is the average reward of all rewards up to time t. Note: if $\bar{R}_t$ is set to be a constant 0 the performance of the algorithm is reduced (e.g. any baseline works so long as it exists and isn't constant).

Now let's walk through how the gradient bandit algorithm approximates to gradient ascent.

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}, \tag{4}$$

Where the expected reward is:

$$\mathbb{E}[H_t(a)] = \sum_x \pi_t(x) q_*(x),$$

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)}[\sum_x \pi_t(x) q_*(x)] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&= \sum_x (q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)}
\end{aligned}$$

We can add the B_t during the final step above because the total change in $H_t(a)$ must be zero since all probabilities must sum to one.

$$\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$$

Now we multiply by $\pi_t(x)/\pi_t(x)$

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x (q_*(x) - B_t)\frac{\partial \pi_t(x)}{\partial H_t(a)}\pi_t(x)/\pi_t(x)$$

Now we have an expectation that will allow us to sum over all possible values of x of the random variable A_t

$$= \mathbb{E}[(q_*(A_t) - B_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)]$$
$$= \mathbb{E}[(R_t - \bar{R}_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)]$$

In this case we let $B_t = \bar{R}_t$ and $R_t = q_*(A_t)$, which we can do since $\mathbb{E}[R_t|A_t] = q_*(A_t)$

Briefly let $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$ without proof netting us:

$$= \mathbb{E}[(R_t - \bar{R}_t)\pi_t(A_t)(\mathbb{1}_{a=A_t} - \pi_t(a))/\pi_t(A_t)]$$
$$= \mathbb{E}[(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a))]$$

Substituting this back into (4) we get:

$$H_{t+1}(a) \doteq H_t(a) + \alpha(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a)), \forall a$$

Which turns out to be equivalent to what was laid out in (2) and (3).

Now let's go back and show $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$ starting with the quotient rule:

$$\frac{\partial}{\partial x}[\frac{f(x)}{g(x)}] = \frac{\frac{\partial f(x)}{\partial x}g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}$$

Thus:

$$
\begin{aligned}
\frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(x) \\
&= \frac{\partial}{\partial H_t(a)} [\frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}}] \\
&= \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{(\sum_{y=1}^{k} e^{H_t(y)})^2} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{(\sum_{y=1}^{k} e^{H_t(y)})^2} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{(\sum_{y=1}^{k} e^{H_t(y)})^2} \\
&= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x) \pi_t(a) \\
&= \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))
\end{aligned}
$$

1.7. Associative Search (Contextual Bandit)

Up to this point the formulations of the bandit problem have been non-associative tasks, meaning there is no need to tie certain actions to certain situations. The algorithms we have seen thus far have sought to find a best action that is either stationary or very slowly changing, but if we want to expand our horizons to the general reinforcement learning task we need to be able to learn a policy (a mapping from a situation a set of actions).

If for example we saw a version of the bandit problem where the value of each action changed randomly and drastically with each step the algorithms we've previously seen would be worthless. But if we have some distinct and predictable clue that queues us into how the situation has changed (but not its action values) then we may be able to begin to learn a policy associating that clue with a best action.

1.8. Summary

Despite the simplicity of these methods they can actually be used to approach very complex problems and are in some way cutting edge. In the form we've laid out so far, however, they lack the complexity needed to tackle the full reinforcement learning problem, but as we'll see later on they can be used in that context when proper considerations are taken.

Another method for approaching the k-armed bandit problem not mentioned is a

References

Reinforcement Learning: An Introduction, 2nd Edition, Richard S. Sutton, Andrew G. Barto, 2018.