K-Armed Bandit

An agent chooses between "k" actions and receives a reward based on the action it takes.

To decide which action is the best, we must define the value of taking each action --> action values or action value function.

How is the bandit problem similar or different to the supervised learning problem?

Similarities:

- Both problems are involved to learn a rule from data to make decisions under uncertainty. Both aim to optimize an objective function.

Differences:

- Supervised Learning is to learn a rule that generalizes well to unseen data, to map the data to a fixed set of labels. Bandit arm is to maximize cumulative reward over time by taking action and interacting with the environment.

Action value is the expected reward when the action is taken

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a] \quad \forall a \in \{1, 2, \dots, k\} = \sum_r p(r|a) \cdot r$$

Objective is to maximize the value

$$\underset{r}{\text{argmax}}\ q_*(a)$$

How to estimate action value $q_*(a)$ → Sample-Average Method

$$Q_t(a) \doteq \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

- Incremental update rule

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ &= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)Q_n\right) \\ &= Q_n + \frac{1}{n}\left(R_n - Q_n\right) \end{aligned}$$

- Generalize the incremental update rule

$$Q_{n+1} = Q_n + \alpha_n\left(R_n - Q_n\right), \quad \alpha_n \to [0,1]$$

Non-Stationary Bandit Problem

Reward of one or some of bandits are changing along with the time.

Hence we can decay the past rewards.

$$Q_{n+1} = Q_n + \alpha_n\left(R_n - Q_n\right) = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

↳ consistant stepsize and effectively solve the problem with time

Exploration vs. Exploitation

- Exploration allows the agent to improve his knowledge about each action leading to long-term benefit.
- Exploitation exploits the agent's current estimated values to try to get the most reward as short-term benefit.
- How to choose -> choose randomly? (below are different methods to drive exploration)

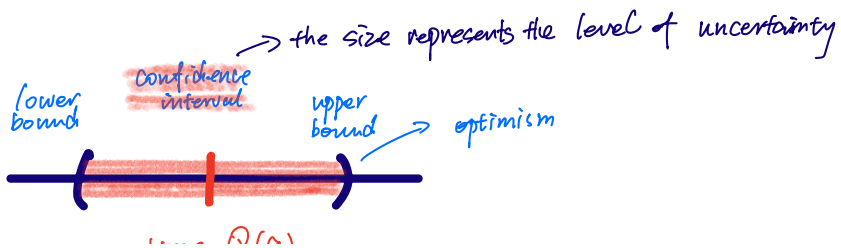
Epsilon-Greedy Algorithm: The epsilon-greedy algorithm is used to balance exploitation and exploration. Instead of always choosing the action with the highest estimated reward (exploitation), it occasionally chooses a random action (exploration). Specifically, with probability \epsilon , it selects a random action uniformly, and with probability 1 - \epsilon , it selects the action with the highest estimated reward.

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\ Q_t(a) & \text{w/ probability } 1-\varepsilon \\ a \sim \text{Uniform}(\{a_1, \cdots, a_k\}) & \text{w/ probability } \varepsilon \end{cases}$$

(Better solution to balance) Initial Optimistic Value: set a large initial value of Q1 for each arm.

- Encourage exploration more only in early learning stage
- Not suitable for non-stationary problems as the value may change over time and on optimistic agent may have already settled on a particular action
- what the optimistic value should be is unknown

Upper-Confidence Bound (UCB) Action Selection



true $q(a)$

UCB Algorithm

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

$Q_t(a)$: exploit

$c\sqrt{\frac{\ln t}{N_t(a)}}$: explore → $C\sqrt{\frac{\ln \text{timesteps}}{\text{times action } a \text{ taken}}}$

suppose $\ln t$ fixed, when $N_t(a)$ small.
this term is large which encourages exploration.

More Algorithms

- Gradient Bandit (a numerical method to learn the preference over each action without any reward interpretation. This leverages gradient ascent to update the preference and determine the action probabilities according soft-max distribution.

- Contextual Bandits. It tries to solve an associative search task, so called because it involves both trial-and-error learning to search for the best actions, and association of these actions with the situations in which they are best.