Multi-Arm Bandits

These notes are compiled from various sources and contain introduction to multi-arm bandits .

Multi Arm Bandits

The exploration and exploitation dilemma exists in many aspects of our life. The multi-arm bandit problem is a classic problem that demonstrates the exploration vs exploitation dilemma. Imagine you are in a casino facing multiple slot machines and each is configured with an unknown probability of how likely you can reward at one play. The question becomes,

```
What is the best strategy to achieve highest long-term rewards ?
```

Let us first consider a setting of having infinite number of trials. The restriction on a finite number of trials introduces a new type of exploration problem. For instance, if the number of trials is smaller than the number of slot machines, we cannot even try every machine to estimate the reward probability and hence we have to behave smartly w.r.t. a limited set of knowledge and resources (i.e time).

The naive approach in this case might be to keep playing with one machine for many rounds so as to eventually estimate the "true" reward probability according to the law of large numbers. However this is quite wasteful and does not guarantee the best long term reward.

Definition

A bernoully multi-arm bandit can be descibed as a tuple of <A, R> , where:

- We have K machines with reward probabilities $\theta_1, \theta_2, \theta_3, \ldots$.
- At each time step t, we take an action a on one slot machine and reciece a reward r.

- A is a set of actions, each referring to the interaction with one slot machine. The value of action a is the expected reward, $Q(a) = \mathbb{E}[r|a] = \theta$. If action at the time step t is on the i-th machine, then $Q(a_t) = \theta_i$.
- R is a reward function. In the case of Bernoulli bandit, we observe a reward r in a *stochastic* fashion. At the time step t, $r_t = R(a_t)$ may return reward 1 with a probability $Q(a_t)$ or 0 otherwise.

It is a simplified version of Markov Decision Process, as there is no state S.

The goal is to maximize the cumulative reward $\sum_{t=1}^{T} r_t$. If we know the optimal action with the best reward, then the goal is same as to minimize the potential regret or loss by not picking the optimal action.

The optimal reward probability θ^* of the optimal action a^* is:

$$\theta^* = Q(a^*) = max_{a \in A} Q(a) = max_{1 \geq i \leq K} \theta_i$$

The loss function in our case will capture the regret we might have by not selecting the optimal action up to the time step T:

$$L_T = E[\sum_{t=1}^{T} (\theta^* - Q(a_t))]$$

Bandit Strategies

Different solutions can be developed depending on the way we are exploring:

- *No Exploration* : This is the most naive approach.
- *Random Exploration*
- Exploring Smartly with preference to uncertainty.

ϵ - Greedy Algorithm

The ε-greedy algorithm takes the best action most of the time, but does random exploration occasionally. The action value is estimated according to the past experience by averaging the rewards associated with the target action a that we have observed so far (up to the current time step t):

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^{t} r_t 1[a_\tau = a]$$

where *1* is the binary indicator function and $N_t(a)$ is how many times the action a has been selected so far, $N_t(a) = \sum_{\tau=1}^{t} 1[a_\tau = a]$

According to the ε-greedy algorithm, with a small probability $\epsilon\epsilon$ we take a random action, but otherwise (which should be the most of the time, probability 1-ϵ) we pick the best action that we have learnt so far:

$$\hat{a_t^*} = argmax_{a \in A} \hat{Q}_t(a)$$

Upper Confidence Bounds

Random exploration gives us an opportunity to try out options that we have not known much about. However, due to the randomness, it is possible we end up exploring a bad action which we have confirmed in the past (bad luck!). To avoid such inefficient exploration, one approach is to decrease the parameter ε in time and the other is to be optimistic about options with *high uncertainty* and thus to prefer actions for which we haven't had a confident value estimation yet. Or in other words, we favor exploration of actions with a strong potential to have a optimal value.

The Upper Confidence Bounds (UCB) algorithm measures this potential by an upper confidence bound of the reward value, $U^t(a),$ so that the true value is below with bound

$$Q(a) < \hat{Q}_t(a) + \hat{U}_t(a)$$

with high probability. The upper bound $U^t(a)$ is a function of $N_t(a)$; a larger number of trials $N_t(a)$ should give us a smaller bound $\hat{U}_t(a)$.

In UCB algorithm, we always select the greediest action to maximize the upper confidence bound:

$$a_t^{UCB} = argmax_{a \in A} \hat{Q}_t(a) + \hat{U}_t(a)$$