C6: Multi-Armed Bandit Problem

▼ Table of Content

The MAB Problem
- Epsilon-greedy Policy
- Softmax Exploration Algorithm/Boltzmann Exploration
- Upper Confidence Bound Algorithm

Applications of MAB

The MAB Problem

- MAB is a classic RL problem to express the exploration & exploitation dilemma.
- Essentially, MAB problem goes like this:
  - In a casino, you have a number of slot machine(s) with each of the Slot Machine having its own probability distribution.
  - The goal of MAB is to find the k number of slot machine(s) which will gives us maximum Cumulative Reward over sequence of time.

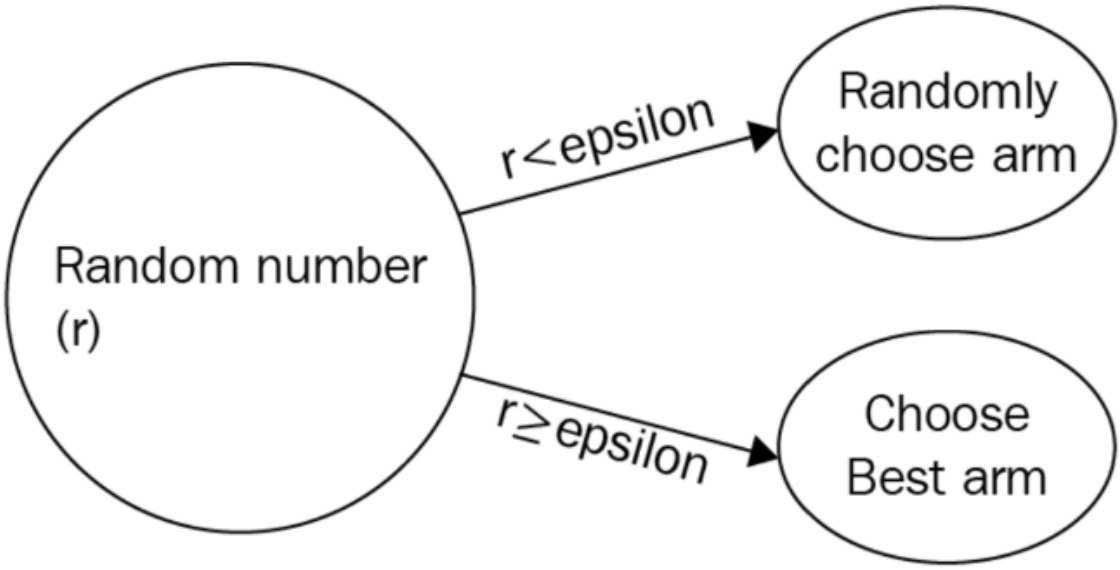
$$Q(a) = \frac{\text{Sum of rewards received from the arm}}{\text{Total number of times the arm was pulled}}$$

  - The exploration-exploitation dilemma is as follows:
    To achieve our goal in limited number of steps, how do we find the best arm?
    - Explore all arms and make decision or
    - Choose the arm that already gave us a maximum cumulative reward.
- The dilemma can be solved using various exploration strategies as follows:
  - Epsilon-greedy policy
  - Softmax exploration
  - Upper confidence bound algorithm
  - Thomson sampling technique

Epsilon-greedy Policy

- In epsilon-greedy policy, the action is selected using 1-epsilon (always the best solution) or random action based on probability epsilon:

```
def epsilon_greedy(epsilon):
    rand = np.random.random()
    if rand < epsilon:
        action = env.action_space.sample()
    else:
        action = np.argmax(Q)
        return action
```



Softmax Exploration Algorithm/Boltzmann Exploration

- In the epsilon-greedy policy, we consider all of the non-best arms equivalently (i.e. we only select the policy that gives maximum q-value),
  but in softmax exploration, we select an arm based on a probability from the Boltzmann distribution (i.e. we randomly select the actions that has softmax-probability greater than random threshold)
  - Boltzmann distribution

$$P_t(a) = \frac{exp(Q_t(a)/\tau)}{\sum_{i=1}^{n} exp(Q_t(i)/\tau)}$$

- Softmax Exploration works as follows
  - Temperature Parameter τ is used to determine how many random arms to explore
    - τ is high, all arms will explored equally
    - τ is low, high-rewarding arms will have higher chance to be chosen

```
def softmax(tau):
    total = sum([math.exp(val/tau) for val in Q])
    probs = [math.exp(val/tau)/total for val in Q]
    threshold = random.random()
    cumulative_prob = 0.0
    for i in range(len(probs)):
        cumulative_prob += probs[i]
        if (cumulative_prob > threshold):
            return i
    return np.argmax(probs)
```

Upper Confidence Bound Algorithm

- In UCB Algorithm, the upper bound of a confidence interval is chosen to solve the Exploration and Exploitation dilemma.
- The reason is at the start, the constructing of confidence interval is not as accurate and thus more different UCB could be chosen and thus, exploration.
- Overtime, the Confidence Interval shrink closer to the True mean as the sample size increase and thus, the maximum UCB would be chosen and thus, exploitation.

$$Arm = argmax_a[Q(a) + \sqrt{\frac{2log(t)}{N(a)}}]$$

```
def UCB(iters):
    ucb = np.zeros(1)
    #explore all the arms
    if iters < 10:
        return i
    else:
        for arm in range(10):
            # calculate upper bound
            upper_bound = math.sqrt((2*math.log(sum(count))) / count[arm])
```

```
            # add upper bound to Q value
            ucb[arm] = Q[arm] + upper_bound
        # return the arm which has maximum value
        return np.argmax(ucb)
```

Applications of MAB

- In a normal AB-Testing, Exploration and Exploitation is done sequentially which could takes time and might not be suitable for certain use case.
- That is where the MAB problems comes into play where exploration and exploitation is done concurrently in an adaptive fashion.
- Examples of Bandits are used for website optimization, maximizing conversion rate, online advertisements, campaigning and so on.