Reinforcement Learning (RL)

Chapter 2:

Multi-Armed Bandit

Saeed Saeedvand, Ph.D.

Contents

In this Chapter:

- ✓ Multi-Armed Bandit (MAB) problem
- ✓ Exploitation vs Exploration
- ✓ ε-greedy algorithm
- ✓ Upper Confidence Bounds (UCB) algorithm
- ✓ Thompson Sampling algorithm

Aim of this chapter:

- ✓ Understand concepts of Multi-Armed Bandit problem as basic problem for Reinforcement Learning (RL) and discuss multiple approaches to deal with the problem.

Multi-Armed Bandit (MAB)

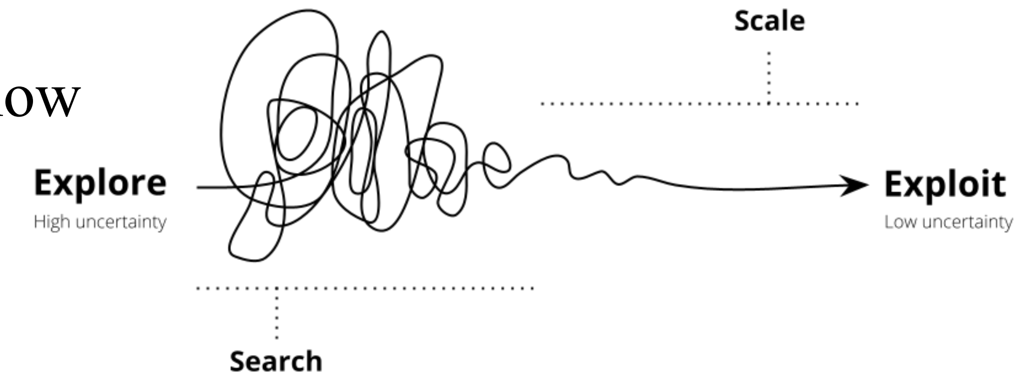
Exploitation vs Exploration:

Exploration: Try new things all the time

Take some risk to collect information about unknown options

Exploitation: Use previous experience

Take advantage of the best option we know



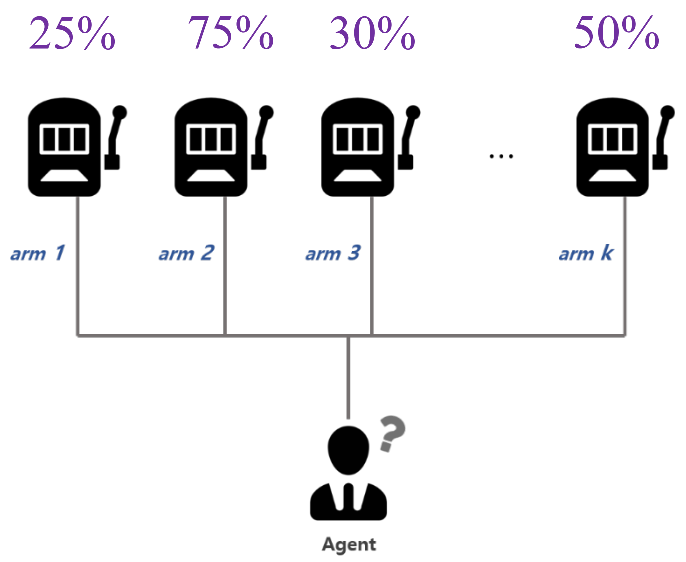
The best long-term strategy may involve short-term sacrifices

Multi-Armed Bandit (MAB)

- ✓ In a **casino facing multiple slot machines** and each is configured with an **unknown probability** of how likely you can **win or lose** at one play

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

- ✓ Considering **infinite number of trials** (to understand concept)

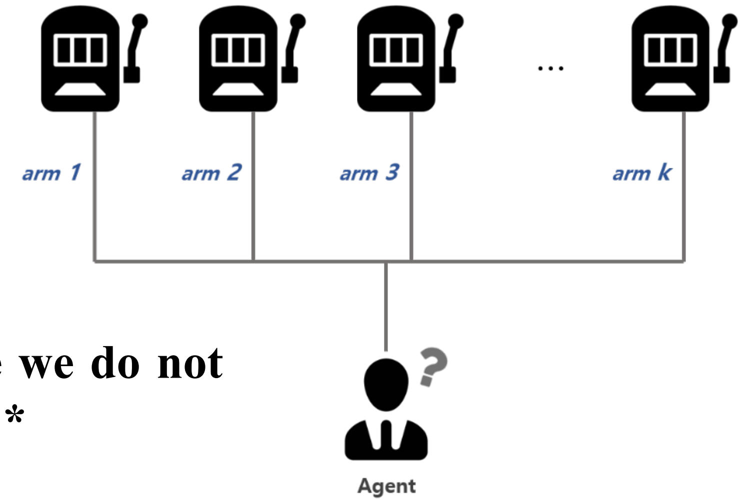


Multi-Armed Bandit (MAB)

Definition:

- ✓ A classic problem that demonstrates the exploration vs exploitation and basic concept of RL
- ✓ It is a **one state problem: (why?)**
  - ➢ Because rewards has no effect on the future rewards and **reward distributions are stationary**
- ✓ The goal is to **maximize the cumulative reward** by doing actions (arms do the actins)

Slot Machines



Note: It is a simple version of Markov Decision Process (MDP) **since we do not have states** (only one state) ****we will see multi state in next chapter****

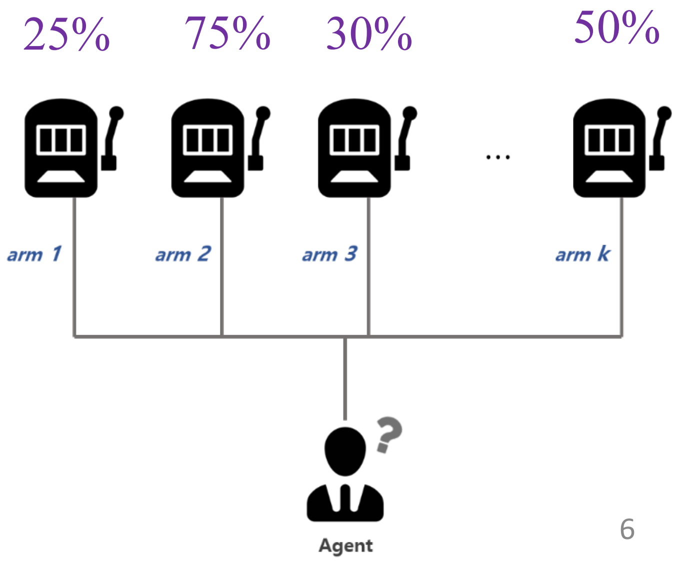
Multi-Armed Bandit (MAB)

The most basic Approach

- ✓ Playing with **one machine for many rounds** to get reward is simplest approach

- ✓ This **does not guarantees the best long-term reward**

- ✓ Also based on limited set of knowledge and resources we need to have **smart approach!**



Multi-Armed Bandit (MAB)

Definition:

- ✓ Considered as a tuple of (a, r)
- ✓ Take an action a and receive reward r (on one slot machine).
- ✓ K machines with reward probabilities of $\{P_1, P_2, \dots, P_K\}$
- ✓ The value of action a is the expected reward $V(a) = \mathbb{E}[r|a] = P$
- ✓ At the time step t the reward function $r_t = R(a_t)$ based on the probability

Multi-Armed Bandit

Example:

- ✓ The goal is to maximize the cumulative reward:

$$max \sum_{t=1}^{T} r_t$$

- ✓ If **we know** the optimal action and best reward the goal can be **minimizing** the **loss function**:

Optimal reward probability

$$\mathcal{L}_T = \mathbb{E} \sum_{t=1}^{T} \left(P^* - Q(a_t)\right)$$

$$P^* = Q(a^*) = \max_{1toK}(P_i)$$

The regret we might have by not selecting the optimal action up to the time step T

Value of taking action a in time t

Multi-Armed Bandit

Multi-Armed Bandit Strategies

1) Playing with **one machine for many rounds**
   - The most bad and naive approach (no exploration)
2) Exploration randomly
   - Not reliable
3) Exploration preference to uncertainty (**wise exploration**)

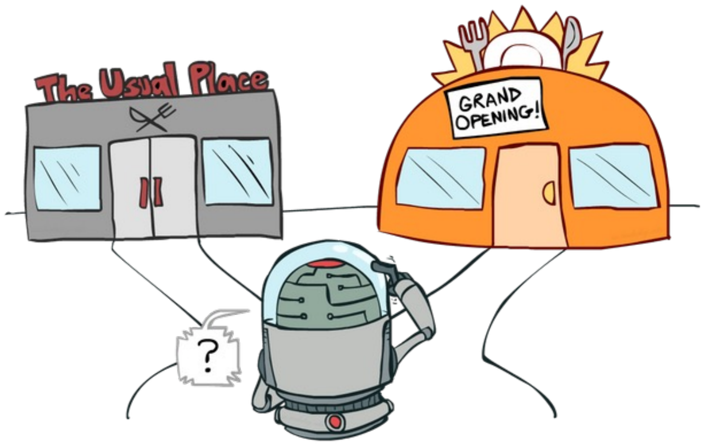
Multi-Armed Bandit

ε-greedy policy concept:

✓ The ε-greedy algorithm takes the **best action** or choose a **random action** exploration occasionally.

How?

a) Generate a random number (*rand* ∈ *[0, 1]*)
b) If *rand* < ε choose a greedy action (**exploitation)**
c) Otherwise choose a random action (**exploration)**



To avoid inefficient exploration, we can decrease the parameter ε in time (we call it epsilon decay)

Multi-Armed Bandit

Multi-Armed Bandit by ε-greedy policy:

✓ **Action value is estimated using the past experience:**

- By averaging the rewards of action that we have observed up to the current time step t:

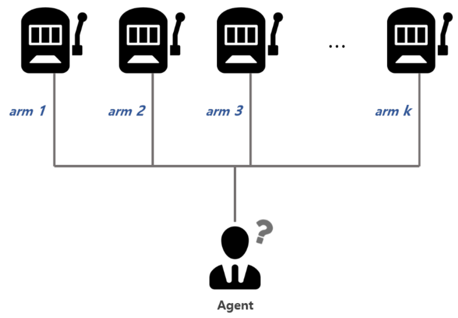
$$\hat{Q}_t(a) = 1/N_t(a) \sum_{t=1}^{T} r_t \mathbb{1}[a = a_t]$$

How many times the action a has been selected before time t

Now we pick the best action that we have learnt by choosing an action:

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

Best estimated action in time t



Multi-Armed Bandit

Challenge of ε-greedy policy:

- ✓ If we can tune ε parameter well we will have good result.
- ✓ **Defining proper value** of the ε sometimes **is challenging!**

Solution

- ✓ Having an **approach** that **relies on number times that we applied an action**.

Upper Confidence Bounds (UCB)

Multi-Armed Bandit

Upper Confidence Bounds (UCB) algorithm

- ✓ **Favor exploration** of **actions** with a **strong potential** to have a optimal value!
- ✓ We have a **bound per each action** that indicates **how confident** we are about the **Q-value of that action.**
  - ✓ Less confident = Bound will be **big**
  - ✓ More confident = Bound will be **small**

Total number of choosing actions (t)

square root

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}, a \in A$$

Total number of times that specific action (a_i) has been selected

- ✓ Larger $N_t(a_i)$ gives us smaller $U_t(a_i)$

Multi-Armed Bandit

Upper Confidence Bounds (UCB) algorithm

- ✓ In fact, we prefer actions that we haven't had a confident value estimation for it yet.
- ✓ UCB algorithm measures this potential by an **Upper Confidence Bound** of the reward value then add to Q-value
- ✓ So, it **select the greediest action** to maximize the upper confidence bound as follows:

$$a_t^{UCB} = argmax_{a \in A} \, Q_t(a) + U_t(a)$$

Multi-Armed Bandit

Upper Confidence Bounds (UCB) algorithm

$$a_t^{UCB} = argmax_{a \in A}\, Q_t(a) + U_t(a)$$

Example

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}, a \in A$$

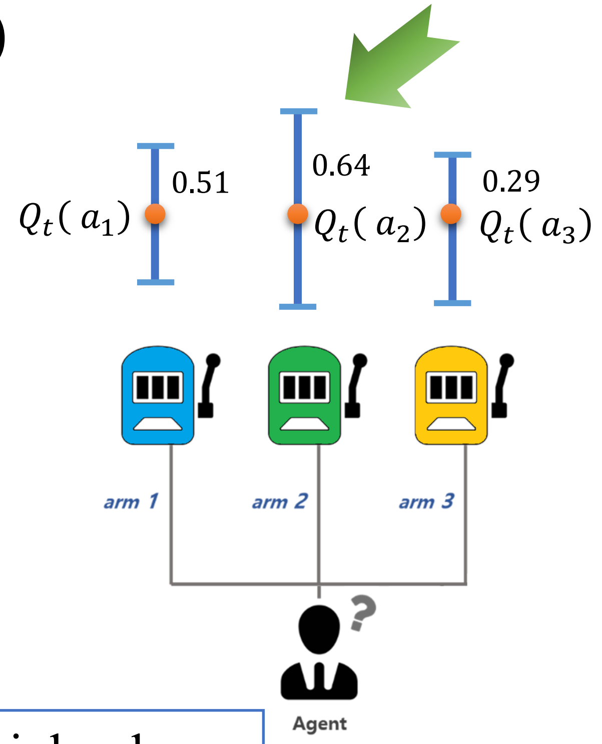
Assumptions:

$Q_{1000}(a_1) = 1$	$N_{1000}(a_1) = 200$
$Q_{1000}(a_2) = 1$	$N_{1000}(a_2) = 150$
$Q_{1000}(a_3) = 1$	$N_{1000}(a_3) = 750$

$$U_{1000}(a_1) = \sqrt{\frac{2 \log 1000}{N_t(a_1)}} = \sqrt{\frac{63.25}{200}} = 0.51$$

$$U_{1000}(a_2) = \sqrt{\frac{2 \log 1000}{N_t(a_1)}} = \sqrt{\frac{63.25}{150}} = \mathbf{0.64}$$

$$U_{1000}(a_2) = \sqrt{\frac{2 \log 1000}{N_t(a_1)}} = \sqrt{\frac{63.25}{750}} = 0.29$$



In machine two we have higher hope (not very sure about its Q-value)

Multi-Armed Bandit

Thompson Sampling algorithm

- ✓ At each time step, we want to select action a according to the probability that a is **optimal**
- ✓ Beta distribution can be considered as $Q(a)$, which is essentially the success probability θ
- ✓ The α and β correspond to the counts when we **succeeded** or **failed** to get a reward respectively.

Multi-Armed Bandit

Thompson Sampling algorithm

- ✓ It samples an expected reward $\hat{Q}(a)$ from the prior distribution $Beta(\alpha_i, \beta_i)$ for every action at each time t.

- ✓ Then the best action is selected among samples.

Beta Distribution

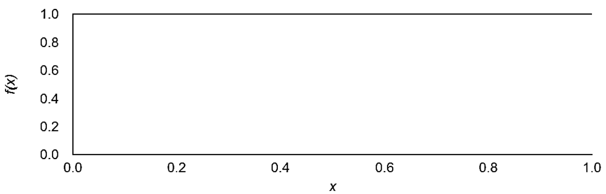
$$P(\theta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \qquad \theta \in [0,1], \alpha, \beta > 0$$

$$B(\alpha,\beta) = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1} du$$

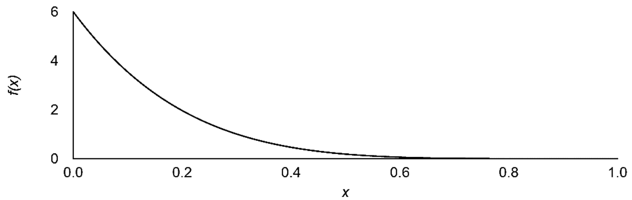
Multi-Armed Bandit

Beta Distribution

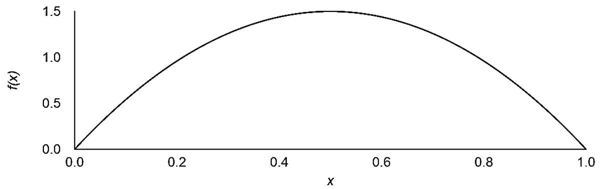
$$P(\theta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$



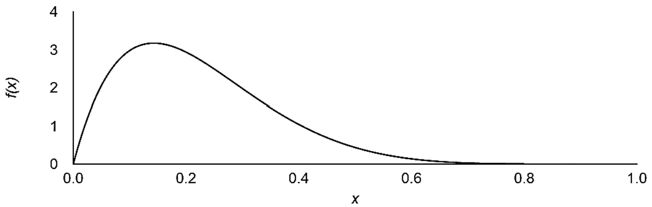
$(\alpha = 1, \beta = 1)$



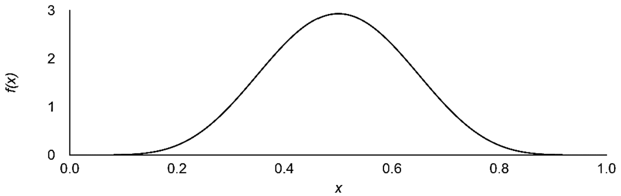
$(\alpha = 1, \beta = 6)$



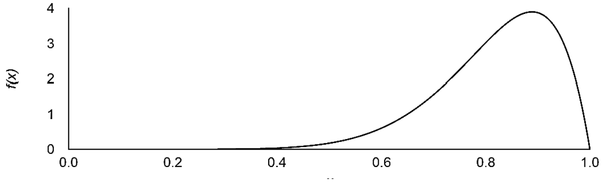
$(\alpha = 2, \beta = 2)$



$(\alpha = 2, \beta = 7)$



$(\alpha = 7, \beta = 7)$



$(\alpha = 9, \beta = 2)$

You can try online:
https://homepage.divms.uiowa.edu/~mbognar/applets/beta.html

Multi-Armed Bandit

Thompson Sampling algorithm

$$P(\theta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

Idea

- ✓ Idea is to ***estimating*** $\boldsymbol{\alpha, \beta}$ **values by change them continuously.**
- ✓ It changes the shape of the distributions related to possible actions.
- ✓ If we **repeat it many times** the sampling the distribution and getting Q-value we will form our desired (true) estimation

https://homepage.divms.uiowa.edu/~mbognar/applets/beta.html

Multi-Armed Bandit

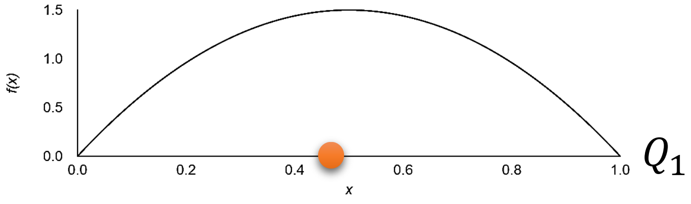
Thompson Sampling algorithm

Iteration 1

Q-value is Maximum: play this bandit

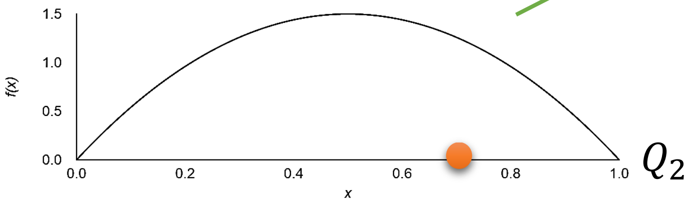
- ✓ Assume initial distribution of $(\alpha = 2, \beta = 2)$ for each bandit

$P(Q_1)$



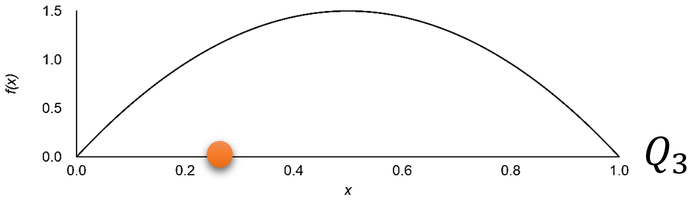
Bandit 1

$P(Q_2)$



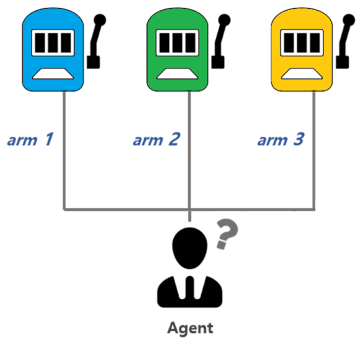
Bandit 2

$P(Q_3)$



Bandit 3

- ✓ Sample Q-value from each of the distributions (Random based on distribution probability)
- ✓ Update the α and β values for the winner (α +1) or loser (β+1) bandit (here Bandit 2)

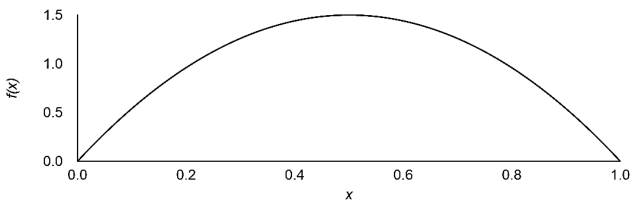


Multi-Armed Bandit

Thompson Sampling algorithm

If machine 2 wins

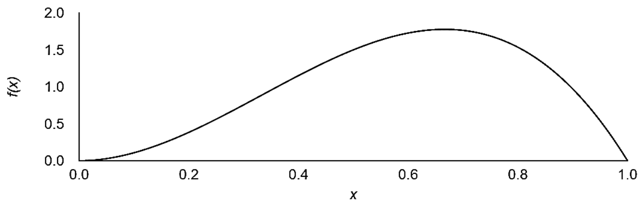
$P(Q_1)$



Q_1

Bandit 1

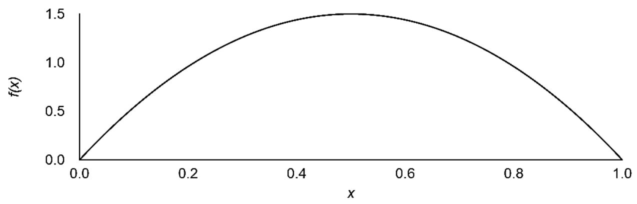
$P(Q_2)$



Q_2

Bandit 2

$P(Q_3)$

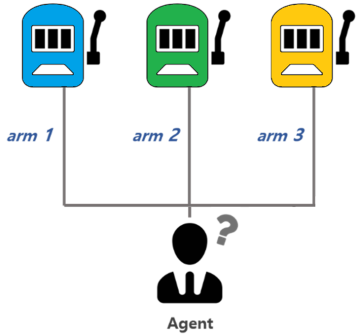


Q_3

Bandit 3

α +1

$(\alpha = 3, \beta = 2)$



Multi-Armed Bandit

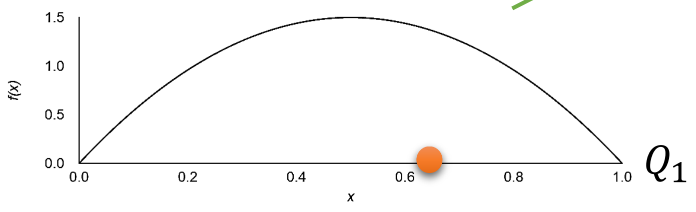
Thompson Sampling algorithm

Iteration 2

Q-value is Maximum: play this bandit

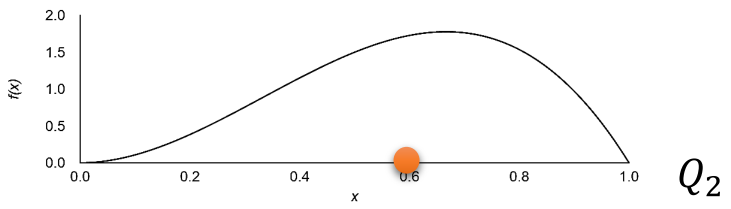
Sample again

$P(Q_1)$



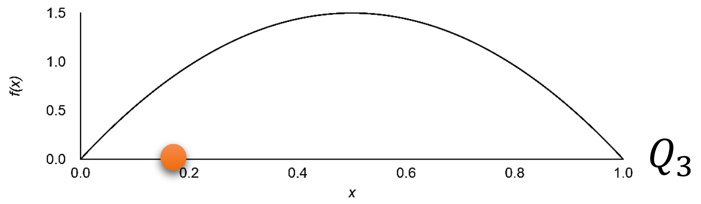
Bandit 1

$P(Q_2)$ α +1

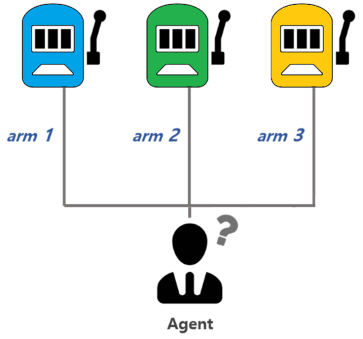


Bandit 2 $(\alpha = 3, \beta = 2)$

$P(Q_3)$



Bandit 3



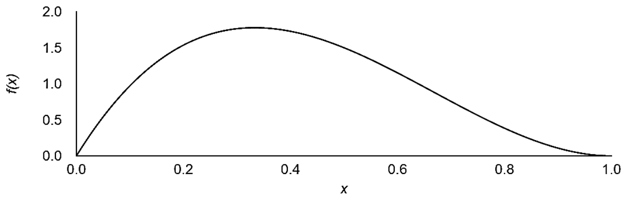
Multi-Armed Bandit

Thompson Sampling algorithm

Iteration 2

If machine 1 loses

$P(Q_1)$



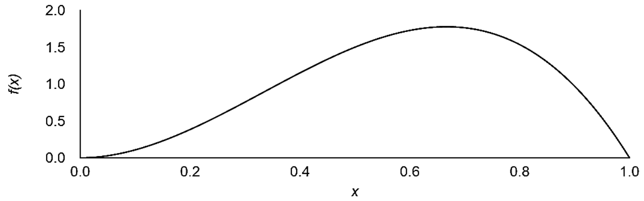
Q_1

Bandit 1

β+1

$(\alpha = 2, \beta = 3)$

$P(Q_2)$

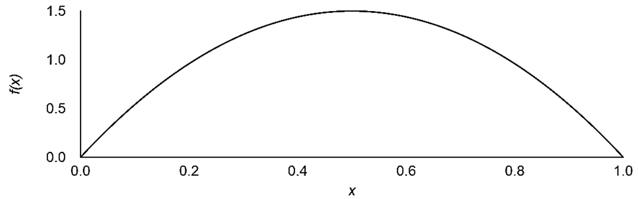


Q_2

Bandit 2

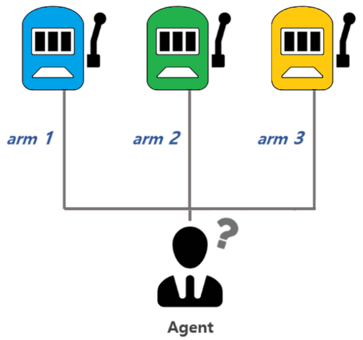
$(\alpha = 3, \beta = 2)$

$P(Q_3)$



Q_3

Bandit 3

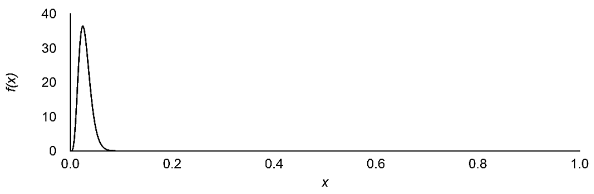


Multi-Armed Bandit

Thompson Sampling algorithm

✓ After **continuing enough iterations** our probability estimations will be close to real machines winning probability

$P(Q_1)$

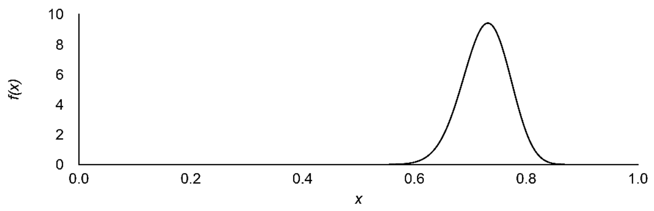


Q_1

Bandit 1

$(\alpha = 6, \beta = 200)$

$P(Q_2)$

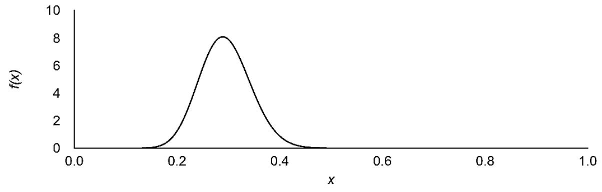


Q_2

Bandit 2

$(\alpha = 80, \beta = 30)$

$P(Q_3)$



Q_3

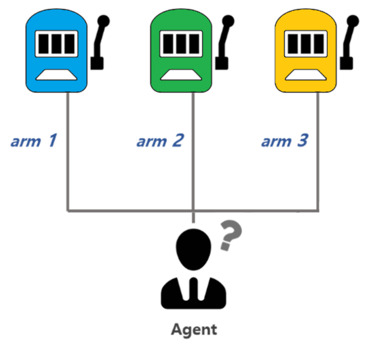
Bandit 3

$(\alpha = 6, \beta = 200)$

Now with sampling we get higher Q-value from **second bandit**. So play most of the time in **Bandit 2.**

Multi-Armed Bandit

Thompson Sampling algorithm



- ✓ We discuss a python solution.

Assignment

- ✓ Compare MAB problem with three algorithms (Thompson Sampling, UCB, and ε-greedy)

  Note: plot and compare mean total reward and time for three algorithms, 1000 iterations (as shown in the example)

Summery

- ✓ We discussed introduction to Multi-Armed Bandit (MAB) problem
- ✓ We learnt ε-greedy algorithm
- ✓ Discussed importance of UCB algorithm
- ✓ Finly used Thompson Sampling algorithm to Solve MAB problem