



# Introduction to Bandits in Recommender Systems(1)

[https://www.youtube.com/watch?v=rDjCfQJ\\_sYY&list=WL&index=31](https://www.youtube.com/watch?v=rDjCfQJ_sYY&list=WL&index=31)

추천에서 Bandit을 어떻게 사용할지를 나타내는 프로그램이다.

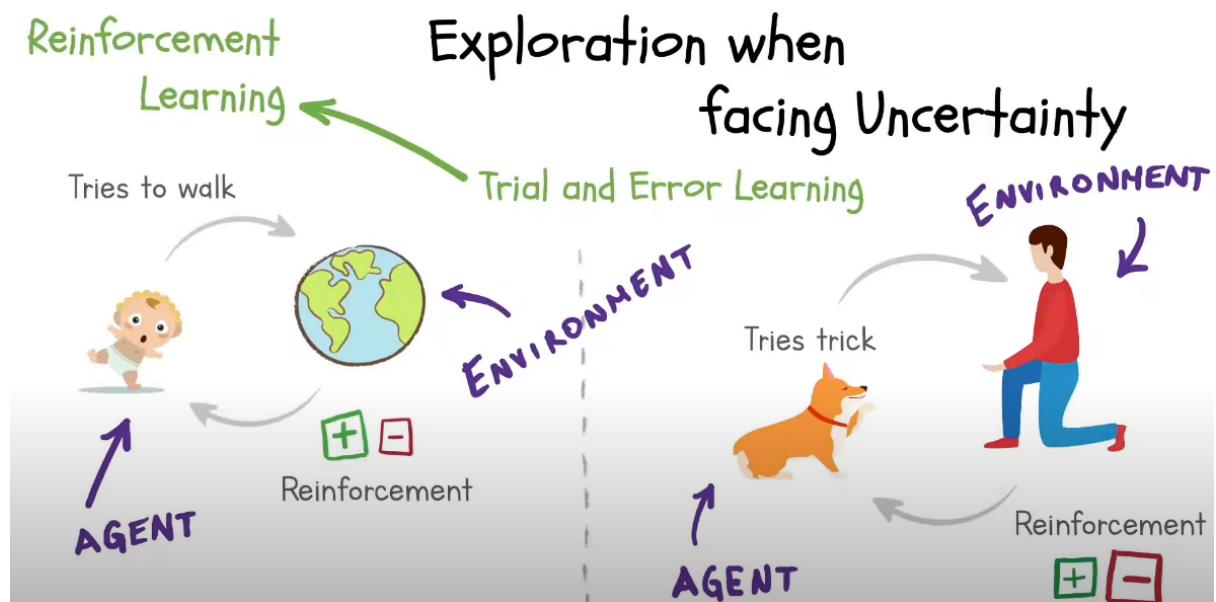
세 부분으로 구성된다.

1. 왜 추천에서 Exploration이 중요한가? 왜 Bandit이 필요한가?
2. 클래식한 Bandit 접근법
3. SOTA Bandit Application

Bandit - 카지노의 슬롯 머신



일상 생활에서 부터 Exploration 과 Exploitation 의 딜레마를 많이 접하는데 음식점을 고르는 것에 예를 들어서 두 개를 설명하고자 함  
좋아하는 것을 고수 할 것인가? (Exploitation )  
새로운 것을 해볼 것인지 ( Exploration )



### 아가의 관점에서 볼 때, 걸음마를 할 때

아가는 많은 것을 시도하고 서로 다른 타입의 액션을 Explore. 넘어지면 다치게 된다. 마침내 아기가 걷는 방법을 배우게 되면 행복 해진다. (positive reinforcement)

## 강아지에게 뭔가 가르칠 때, 새로운 재주를

강아지는 베끼게 되고, 강아지는 앓거나 구르거나 발을 들게 된다. 어떤 명령이든 Explore. 강아지는 내가 원하는 게 뭔지 모르고 trial & error 과정을 계속하게 된다. 강아지의 행동이 어떤 명령에 맞아 보상을 받게 될 때 까지

## 경험을 통해 학습하는 것을 Reinforcement Learning

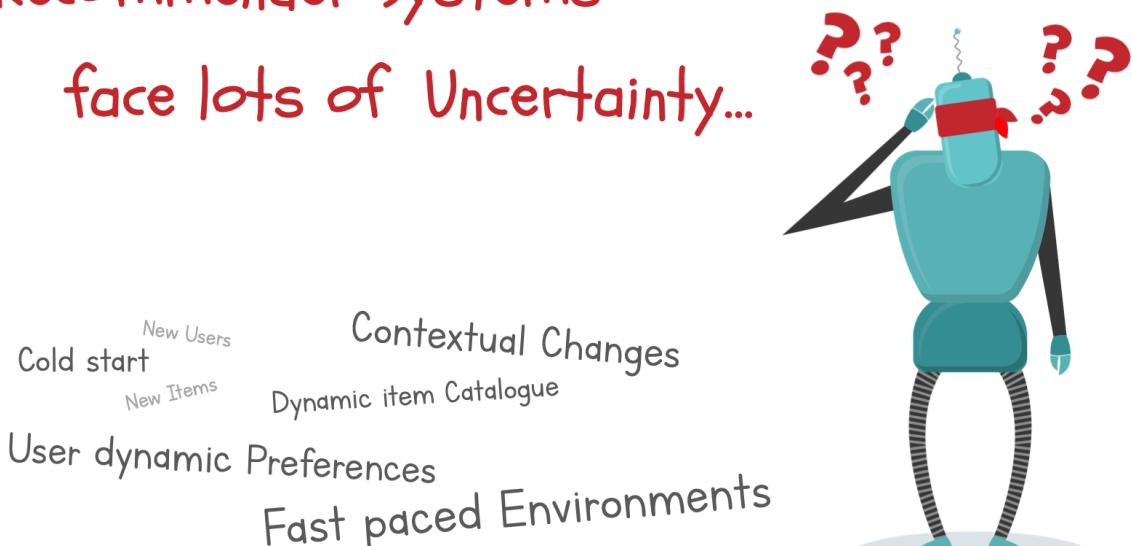
컴퓨팅 환경에서는 Agent(아기, 강아지)가 Environment(세상, 주인)과 상호작용을 한다. Action을 함으로 그림 Environment는 좋은 Action에 Positive 보상을 Negative 피드백을 나쁜 Action을 주고 Action 자체가 Agent를 학습 시키지는 않는다.

Environment는 처음에는 불확실성을 가지고 있다.

- agent는 처음에는 모르기 때문에 exploration으로 학습 해야 한다.
- agent는 나중에는 학습한 것을 사용 해야 한다. 미래에 더 많은 보상을 받기 위해 지식을 exploit 해야 한다.

## Recommender Systems

face lots of Uncertainty...



추천 시스템이 직면하고 있는 많은 시나리오는 불확실성을 포함하고 있다. 현실에서는 흥미가 굉장히 다양한 요소를 통해 변화하고 빠르게 변화한다. 결정을 하기는 데이터가 부족하다.

- 추천 시스템 자체가 Agent가 되어 아이템을 추천하고 유저가 rating을 형태로 피드백을 돌려 준다.
- 추천 시스템의 궁극적인 목적은 positive reward feedback을 극대화 하는 것이다.
- 추천 시스템은 항상 기본적인 선택인 Exploration과 Exploitation에 직면하게 된다.

→ Exploration을 통해 얻은 지식으로 한 Exploitation으로 추천에 비유하는 Gamble 이 실패 할 수도 있지만, 미래를 위해 유저의 preference를 배운다고 볼 수 있다.



### For Explore New Things - Application

- Yahoo News
- Amazon Stream
- Spotify weekly
- Advertising Click
- Tourism

## Recap....

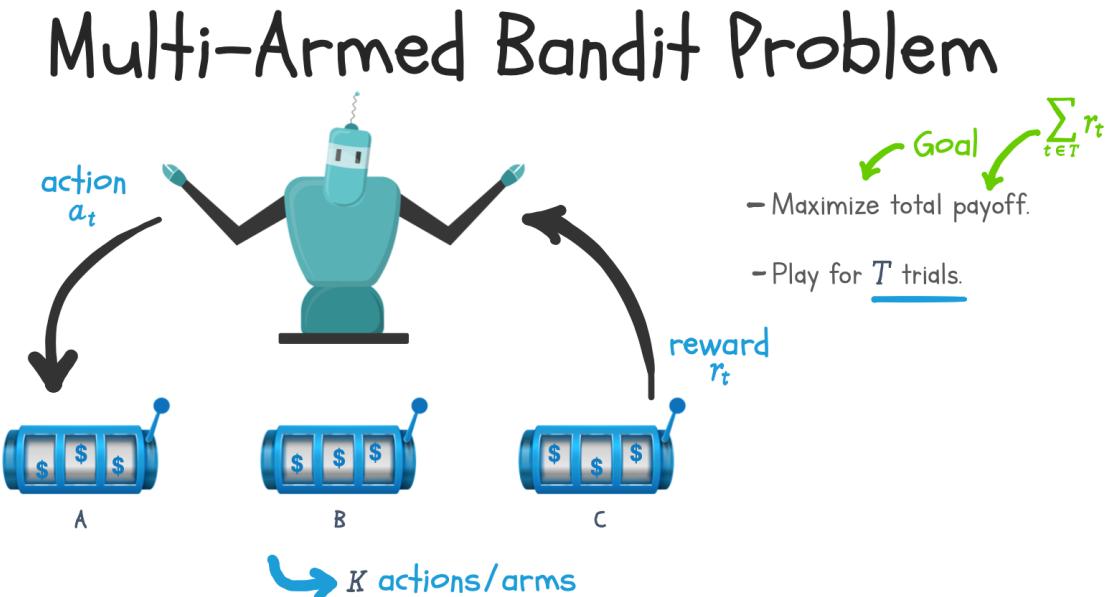
Balancing exploration and exploitation  
is important to learn in interactive  
sequential decision-making settings...

specially to face uncertainty!

REINFORCEMENT  
LEARNING

RECOMMENDATION  
SYSTEMS

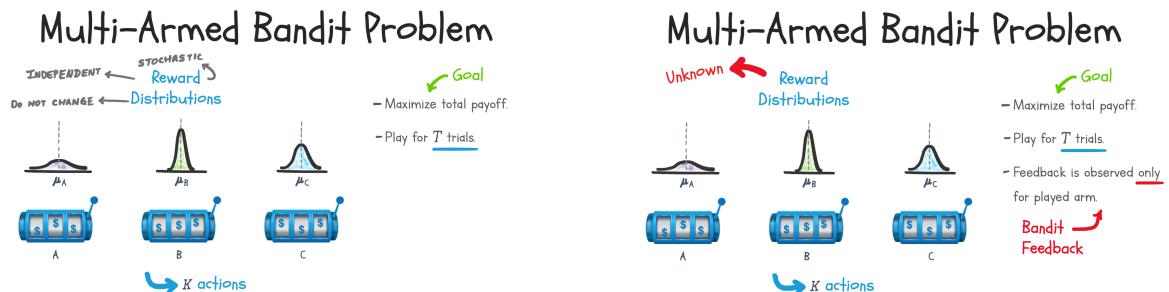
## Classic Bandit Approach - Multi-Armed Bandit Problem



간단한 방법이지만 강력한 방법이기도 하다. 카지노의 슬롯 머신에 비유되었다.

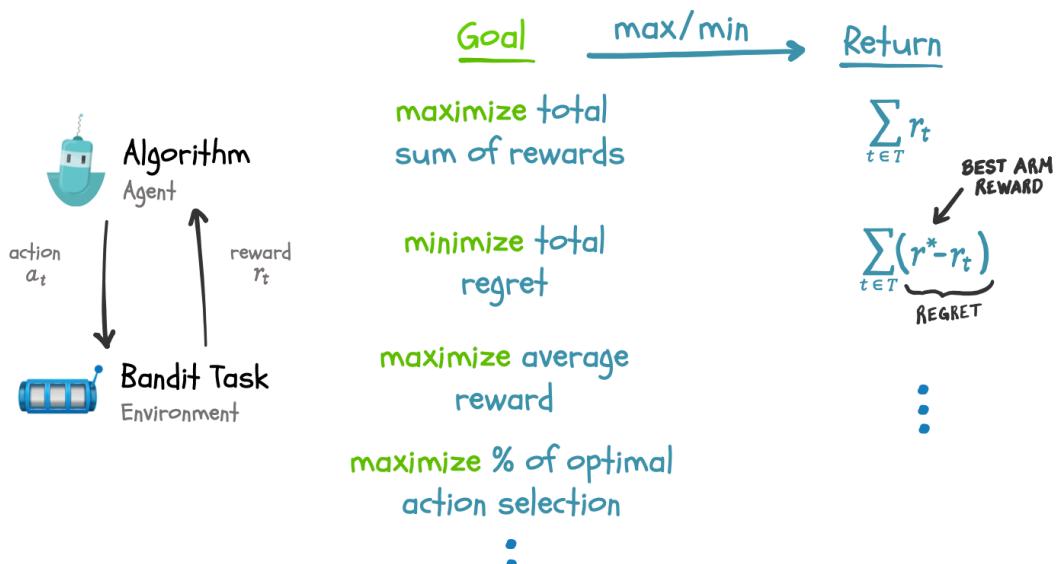
캠블러가 슬롯 머신 앞에 있고 캠블러는 어떤 머신을 각각 몇번 플레이 할지 계속 같은 머신을 사용할지 다른 머신으로 변경할지 정해야 한다.

goal of the bandit은 maximize the total sum of rewards ( Maximizing or Minimizing the 결과 )



- 고정된 수의 Arm 들은 stochastic(정확히 예측할 수 없다)
- reward는 각 Arm이 가지는 고정된 분포에서 나오고 분포는 변하지 않는다.
- 캠블러는 오직 플레이한 Arm에서 나온 결과로 알 수 있고
- 각각의 Arm은 독립적이다.

# How to measure success?

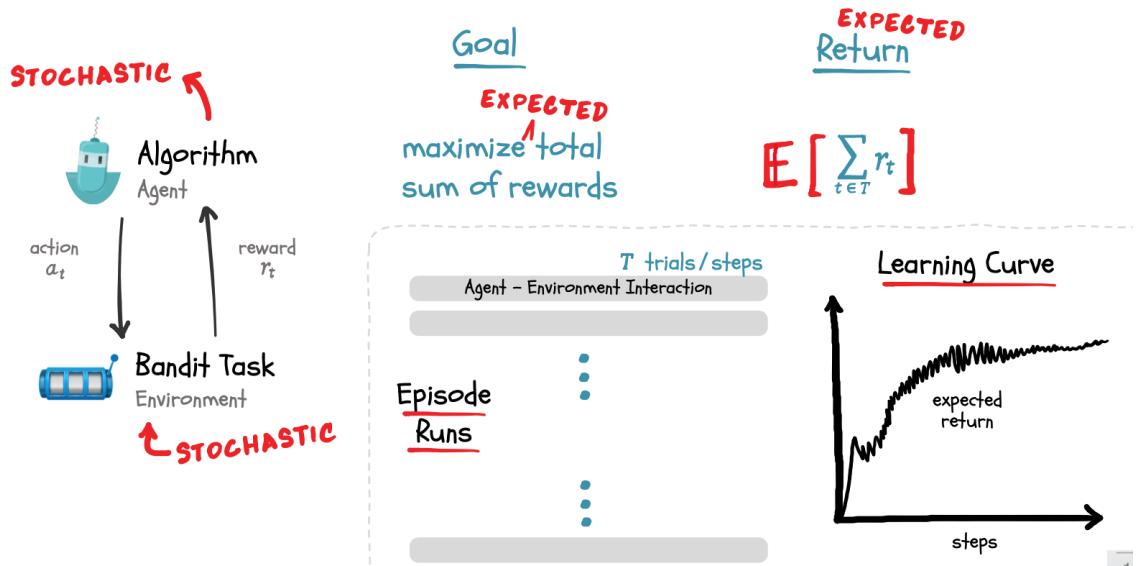


Goal 은 다음과 같이 될 수가 있다.

- 보상의 총 합을 극대화 하는 방향
- 보상의 평균을 극대화 하는 방향
- 총 regret를 최소화 하는것 Min( Best Arm Reward - Reward )
- 최적의 액션 선택 확률을 극대화 하는 방향

다른 Arm들과 비교해서 최적의 Action을 Return하는 Arm을 찾아야 한다.

# How to measure success?



Bandit Approach에서는 결정을 하는 것이 Randomness과 결합 될 필요가 있는데

- Agent가 Bandit 알고리즘과 연결이 되고
- Environment가 Bandit Task과 상호 작용을 한다.

문제가 Stochastic하기 때문에 각 반환되는 값의 예상되는 기댓값을 측정해야 한다.

### 알고리즘을 Evaluating을 할때,

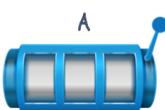
Agent와 Environment가 구성되어 있는 특정 수 만큼의 에피소드를 t 만큼이나 특정 시간 동안을 진행 해야 한다.

에피소드의 예상 되는 결과를 모아 t-trial을 steps의 축으로 하는 plot으로 나타내는데 learning curve라고 한다.

# Let's Play !



Always pays \$5/play



1	\$5
2	\$5
3	\$5
4	\$5
5	\$5
6	\$5
7	\$5
8	\$5



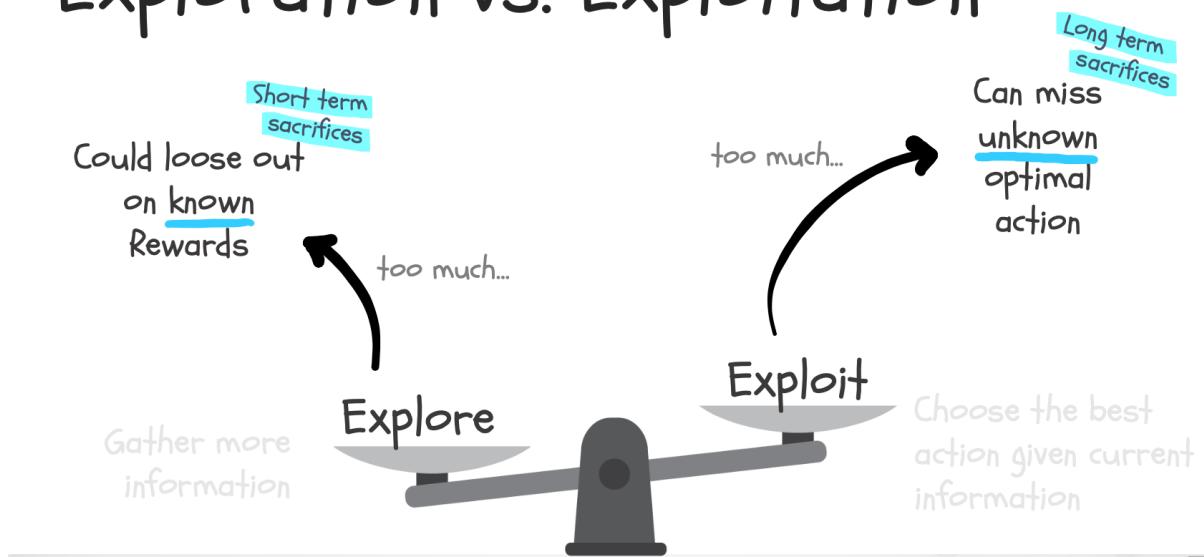
Pays \$60  
%50 of the  
time  
(~\$30/play)

단기적인 관측으로 보았을 때 A 가 더 좋아서 잘못된 Exploitation 전략을 사용했다...

하지만, 장기적인 관점으로 보았을 때, 사실은 슬롯 B가 더 좋다.

하지만 장기 전략은 그 동안의 희생이 많이 들어가게 된다.

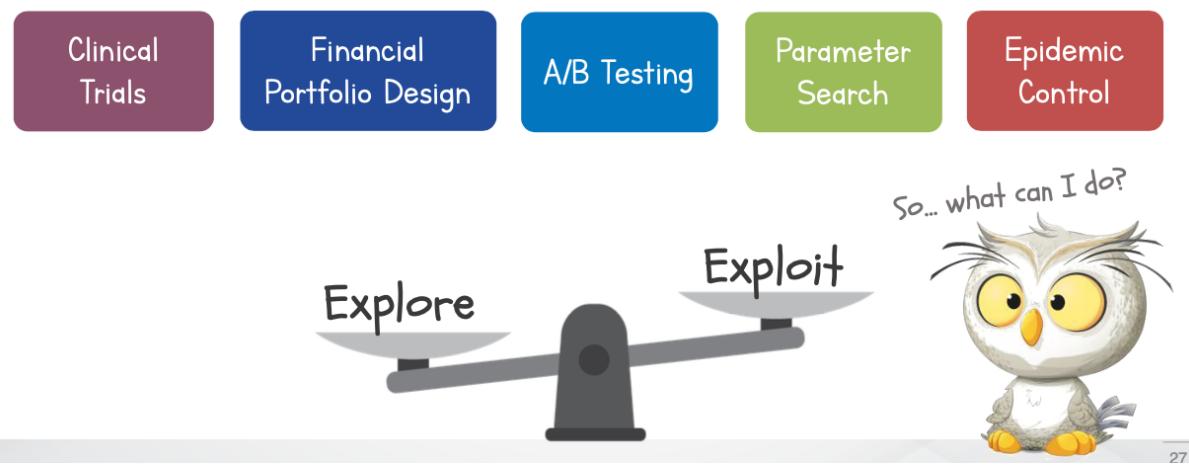
# Exploration vs. Exploitation



⇒ Exploration Exploitation Strategy 를 잘 세워야 하는데 Balance가 중요하다.

- Exploration - 이미 알려진 보상을 얻을 기회를 놓칠 수 있다 ( Short term sacrifice )
- Exploit - 모르는 최적의 액션을 놓칠 수 있다. ( Long term sacrifice )

# Exploration vs. Exploitation



## Clinical Trials

어떤 치료제를 먼저 테스트를 할 것인가?

## Financial Portfolio Design

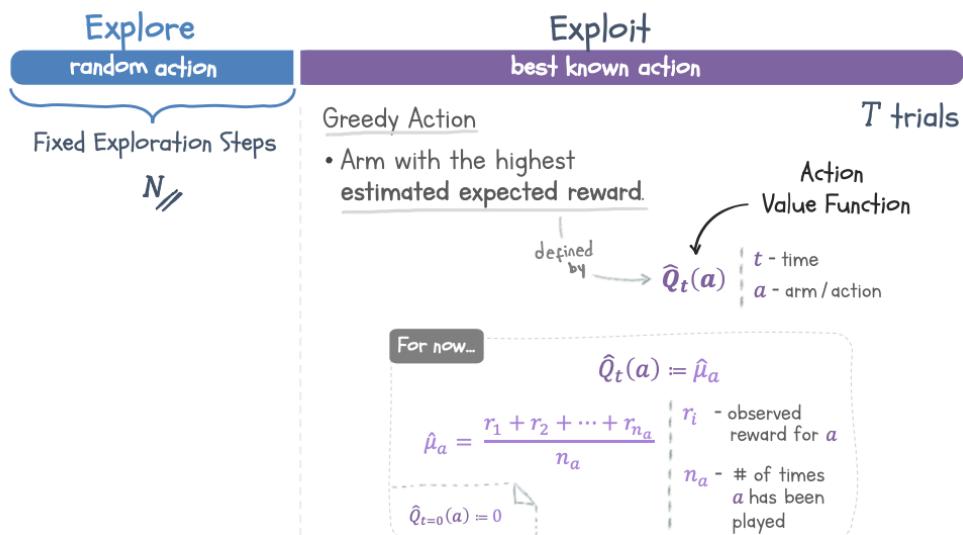
어떻게 자산을 배분할 것인가? 높은 수익을 가져 올 수도 있지만 꽤나 Risk가 있다.

## A/B Testing

비슷한 시스템에서 어떤 버전이 유저가 더 선호할 것인가

## Parameter Search & Epidemic Control

# Explore then Exploit



우선 exploration 과정을 가진다.

- 각 Arm에 대해 어떤 타입의 보상을 우리에게 줄지 테스트 한다.
- 그리고 가장 좋은 한 Arm에 고정(stick)한다.

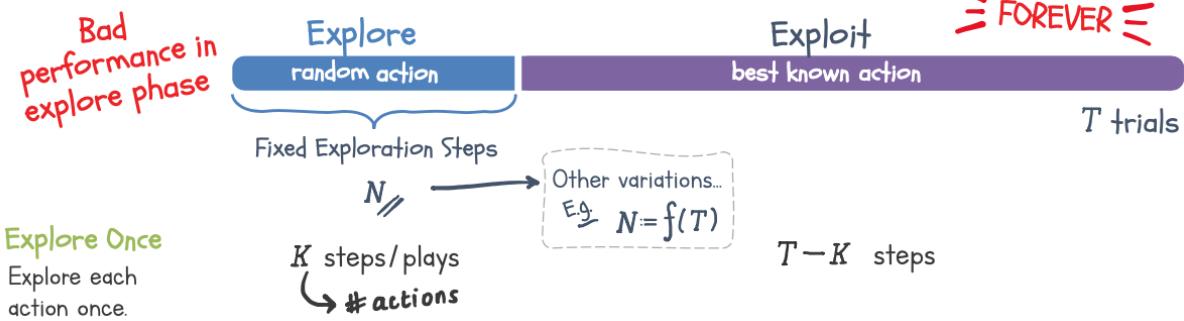
$n$  번의 과정으로... 그 후에 exploit 을 한다.

$n$  번 중에서 각 arm에 대해 랜덤으로 수행하고 Arm의 Action에 대해 우리의 knowledge를 업데이트한다.

보상을 극대화하는 것이 목적으로 action value 함수  $q$ (어떤 action을 선택할지 결정 하는 함수)를 사용한다.

$q$ 는 각 arm에 대한 관측한 결과를 바탕으로  $q$ 를 업데이트한다. ( 각 arm 들 대해  $n_a$ 번의 수행에 대한 평균)

# Explore then Exploit



## Epsilon First

Exploration depends on the size of the time horizon.

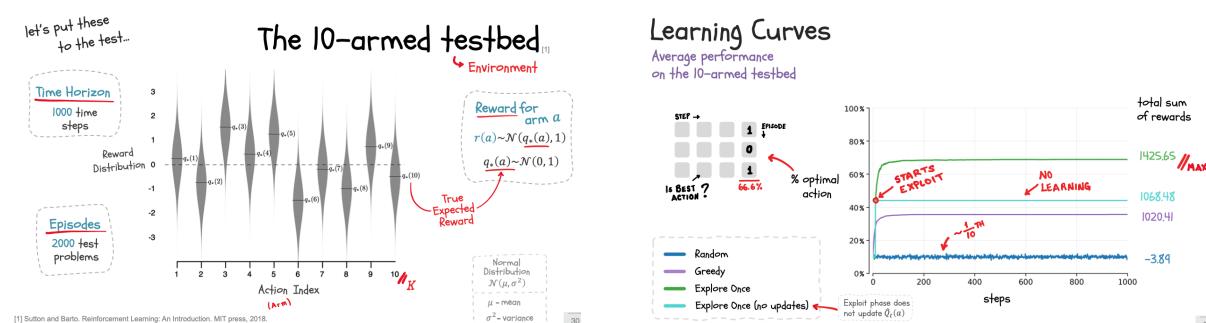
$$\begin{aligned} & \epsilon \cdot T \text{ steps} \\ & 0 \leq \epsilon \leq 1 \quad \text{time horizon} \\ & (1-\epsilon) \cdot T \text{ steps} \end{aligned}$$

n을 정하는 다양한 방법이 있다.

여기에도 이슈가 있는데 Exploration에서 굉장히 나쁜 성능을 보이거나 충분히 Exploration을 하지 않았다면.

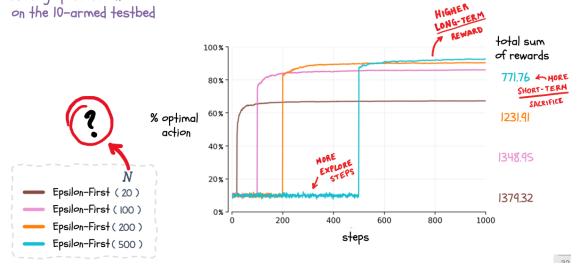
영원히 그 Sub-optimal(차선)에 계속해서 걸려 있을 수 도 있다는 점이다.

$\epsilon - first$  알고리즘은 : 처음에  $\epsilon$  만큼은 exploration 하고 이후로는 exploitation

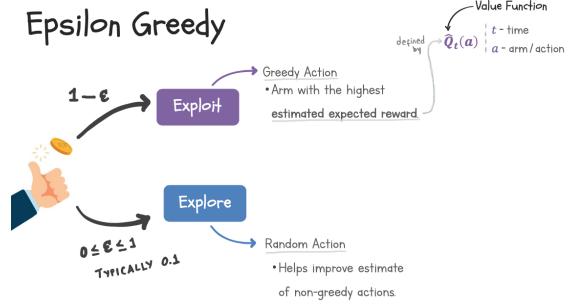


## Learning Curves

Average performance on the 10-armed testbed



## Epsilon Greedy



## Epsilon Greedy Policy

1. 코인이 있다고 가정한다. 코인이 biased 코인이다.
2. 1-epsilon 퍼센트의 시간에서 epsilon 확률 만큼 explore 한다.

## Learning Curves

Average performance on the 10-armed testbed

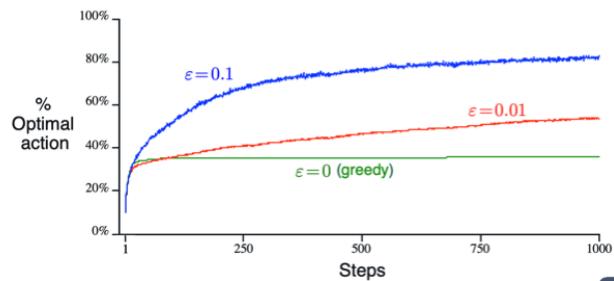
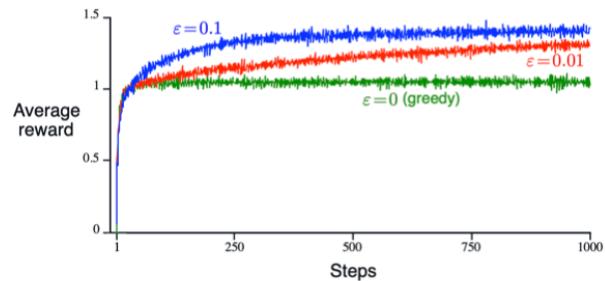
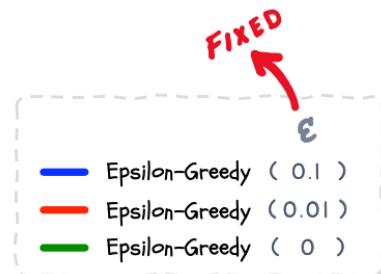
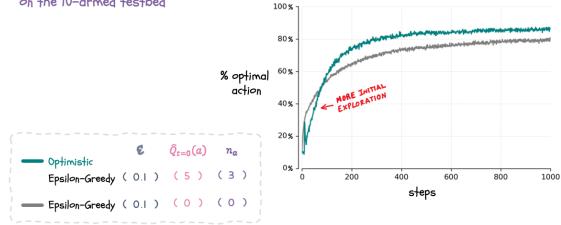


Fig 2.2. [1]

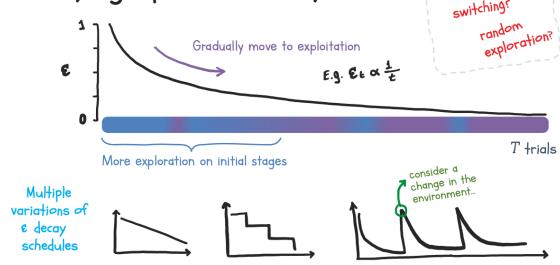
Epsilon이 올라갈 수록 더 빨리 Optimal Action을 빨리 찾을 수 있게 된다. Epsilon이 작다면, 앞에 지정한 1000번의 시도 내에서는 Optimal Action을 찾을 수 없을 수 있다.

## Optimistic Initial Values

Average performance  
on the 10-armed testbed



## Decaying Epsilon Greedy



Decaying Epsilon, 우리의 추천 시스템은 계속해서 흥미가 변경 될 수 있기 때문에 Decaying도 다양하게 가져가야 할 수도 있다

## Suppose...

$$p_t(a) \propto \frac{\hat{Q}_t(a)}{\sum_{b \in \mathcal{A}} \hat{Q}_t(b)}$$

probability of choosing arm  $a$  at time  $t$

set of all arms

## Boltzmann Exploration

Choose action  $a$  with probability:  
PROBABILITY MATCHING

$$p_t(a) = \frac{e^{\frac{\hat{Q}_t(a)}{\tau}}}{\sum_{b \in \mathcal{A}} e^{\frac{\hat{Q}_t(b)}{\tau}}}$$

$\tau$  - temperature

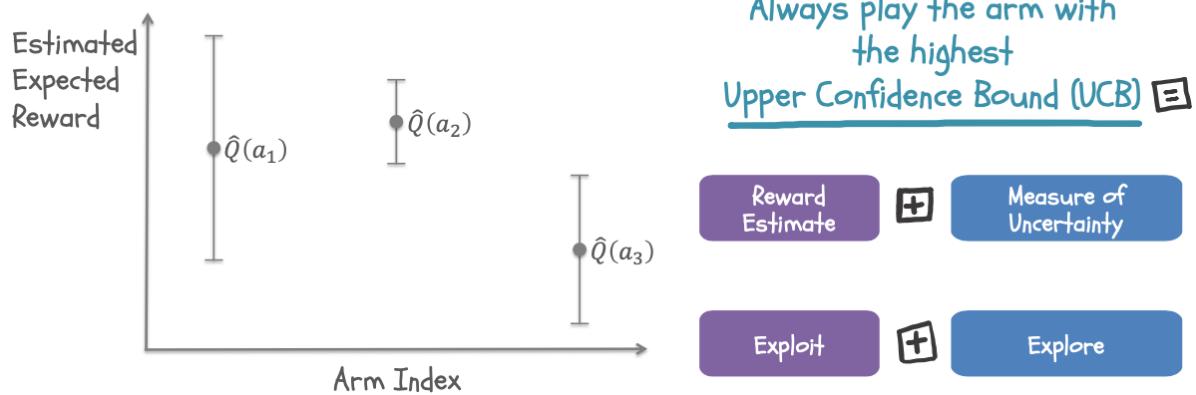
However...  
how to pick  $\tau$ ?  
how to pick  $\epsilon$ ?  
What if we have unlucky initial experiences?

$\tau$ 라는 온도 계수를 softmax 함수와 함께 사용해서 Exploration과 Exploitation의 밸런스를 정한다.

# Upper Confidence Bound Policy

Optimism in face of uncertainty

Uncertainty Guided Exploration



각 Arm이 예상되는 보상을 신뢰 구간과 함께 가지고 그것의 Upper Bound를 가지고 action을 선택하는 것이다.

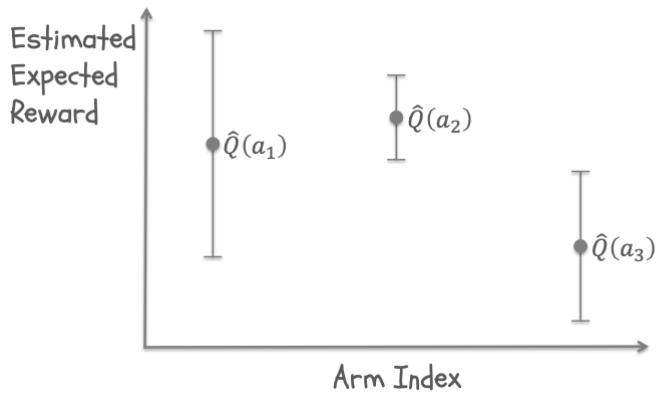
적게한 행동 → 샘플이 작음 → 신뢰 구간이 넓음 → 가치가 낮아도 선택

# Upper Confidence Bound Policy

Optimism in face of uncertainty

UCB1

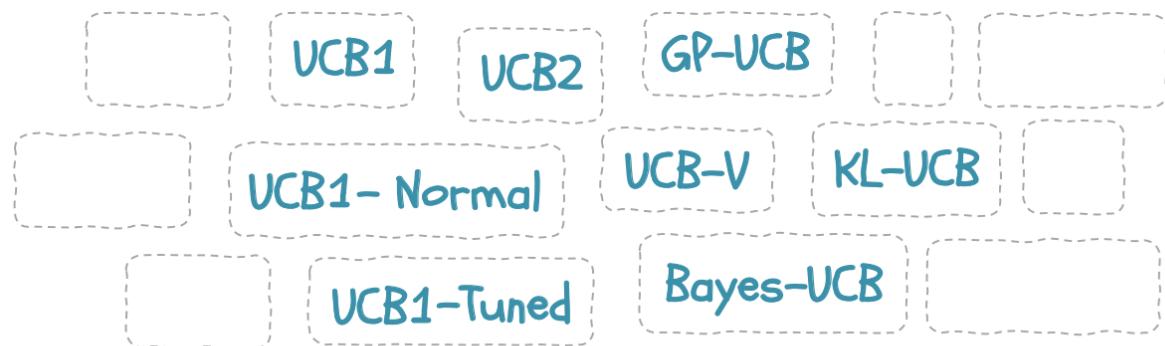
$N$  - Total # of played arms  
 $n_{a_j}$  - Total plays for arm  $a_j$



UCB1은 우선 먼저 Arm들을 한번 해봐야 한다.

Exploration Factor가 있어서 특정 Arm이 계속해서 선점되지 않도록 만든다.

# UCB Variants



## Multi-Armed Bandit (2) - Action Value Methods

큰 제목은 action value methods입니다. 즉 행동가치함수 기반 행동을 취하는 전략의 모음입니다. 기본적으로 action value methods는 행동가치가 큰 쪽으로 행동을 취합니다. 행동가치함수가 \$t\$ 시점에서

↳ <https://yjo.tistory.com/21?category=882868>

```
initialize, for a=1 to k:  
    Q(a) ← 0  
    N(a) ← 0  
op forever:  
    A ← argmaxa Q(a) , with probability 1 – ε  
    a random selection, with probability ε  
    R ← bandit(A)  
    N(A) ← N(A) + 1  
    Q(A) ← Q(A) + 1 / N(A) [R – Q(A)]
```

## The Multi-Armed Bandit Problem and Its Solutions

The multi-armed bandit problem is a classic example to demonstrate the exploration versus exploitation dilemma. This post introduces the bandit problem and how to solve it

↳ <https://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html#ucb1>

