



RecSys 2020 Tutorial: Introduction to Bandits in Recommender Systems(2)

이전에 우리가 공부한 것을 요약하자면

K actions/arms a

$R_a \sim$ unknown stochastic distribution

Action Value Function

$\hat{Q}_t(a)$ estimates $\mathbb{E}[R_a]$



$$\hat{Q}_t(a) = \hat{\mu}_a = \frac{r_1 + r_2 + \dots + r_{n_a}}{n_a}$$

Frequentist

Action Selection Strategy

Use $\hat{Q}_t(a)$ to make decisions

Switching

Exploit

Explore

$$a_t = \arg \max_a \hat{Q}_t(a)$$

$a_t =$ random action

Optimism in the Face of Uncertainty

$$a_t = \arg \max_a \left(\hat{Q}_t(a) + \text{measure of uncertainty} \right)$$

Exploit

Explore

Probability Matching

Use $\hat{Q}_t(a)$ to define $p_t(a)$

Select a_t according to $p_t(a)$

Exploit

Explore

K 개의 액션과 암 ← 각각은 우리가 모르는 분포를 가짐

R_a 를 우리가 모르는 각각의 분포라고 용어로 정의

액션 밸류 함수 Q 를 사용해서 결정하는데 사용한다.

Q 는 이 분포에서 얻을 수 있는 최대 보상을 추정 (우리는 최대 보상 얻는 것을 목표로 하기 때문) (샘플 평균으로 초기화 한다.)

Optimism in the Face of Uncertainty 가 UCB 다.

여기서 R_a 에서 사전 지식이 필요하게 된다. (이전 과정에서는 고려하지 않았음)

K actions/arms a $R_a \sim$ unknown stochastic distribution

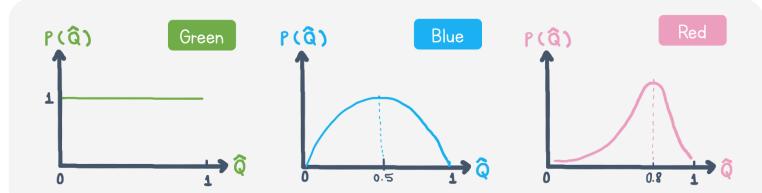
Bayesian Bandit

Action Value Function

$\hat{Q}_t(a)$ estimates $\mathbb{E}[R_a]$

Bayesian
Express our uncertainty about $\hat{Q}_t(a)$ with:

$p(\hat{Q}_t(a))$



Action Selection Strategy

Use $\hat{Q}_t(a)$ to make decisions

베이지언 접근법을 사용한다면, 우리는 Q 의 불확실성을 모델링 한다. 확률을 사용해서 Q 가 가질 수 있는 값의 세 가지 경우를 고려한다. Q 는 0~1 사이의 값을 가진다.

- 모든 암이 같은 1의 확률을 가진다.
- 암들이 0.5에 가까운 확률을 가진다. 확실하게 가지지는 않음
- 암들이 0.8의 확률을 가지는데 더 높은 확신을 가진다.

실제와 같은 분포를 가지지는 않고 불확실성을 가지지만, 그렇다고 믿는다.

Action Value Function

$\hat{Q}_t(a)$ estimates $\mathbb{E}[R_a]$

Bayesian
Express our uncertainty
about $\hat{Q}_t(a)$ with:

Prior $p(\hat{Q}_t(a))$

Observe data $D_a = \{r_1, r_2, r_3, \dots, r_{n_a}\}$

Bayes Rule

Action Selection Strategy

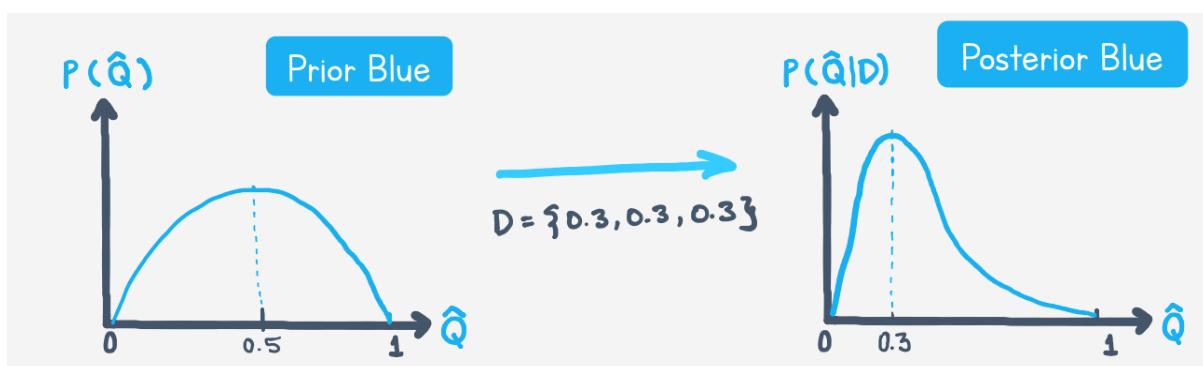
Use $\hat{Q}(a)$ to make decisions

$p(\hat{Q}_t(a) | D_a)$ Posterior

Posterior

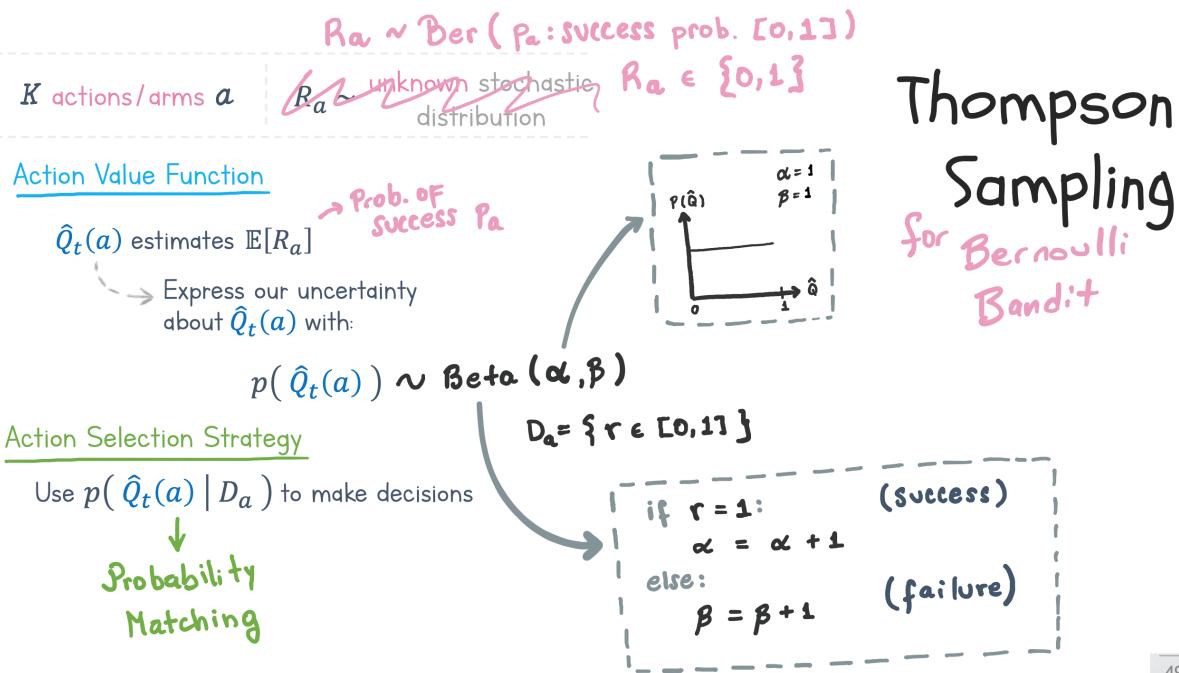
Likelihood

Prior



초기에는 정의하는 사전 확률 분포로 시작한다. 액션에 대한 보상이라는 샘플을 보고, 업데이트해서 사후 확률 분포를 구한다. 그리고 사후 확률 분포를 사용해서 다음 최적 액션을 선택하는데 사용한다. 사후 확률 분포는 다시 사전 확률 분포가 된다.

여기서 파생되는 것이 Thomson Sampling(Posterior Sampling)다.



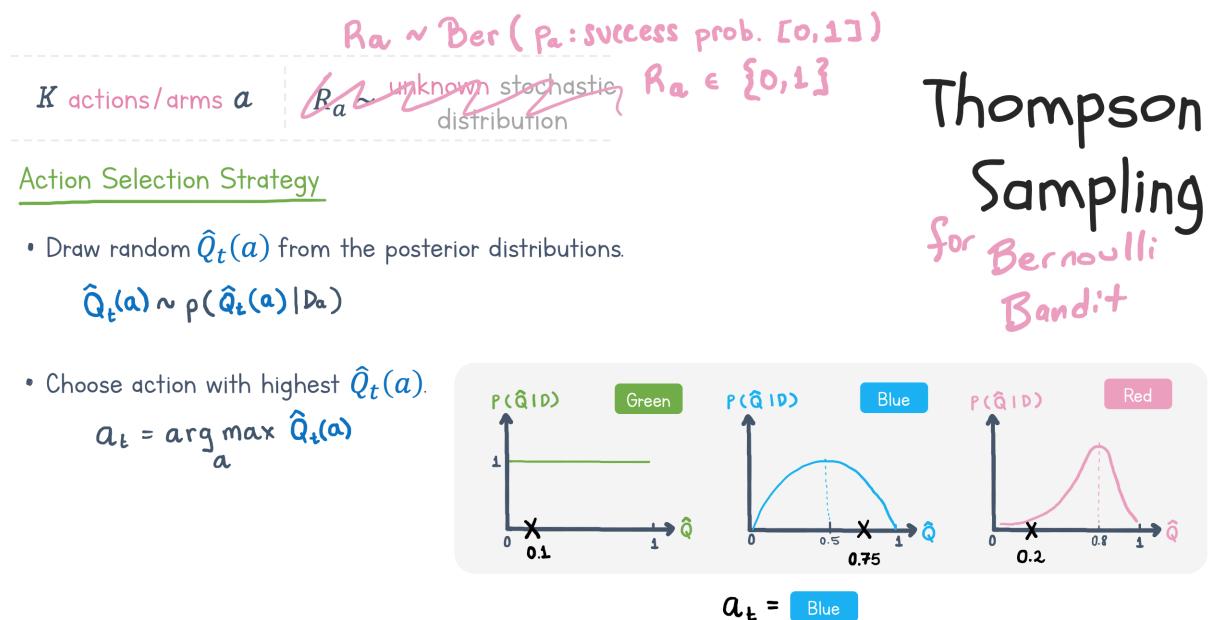
49

Thomson Sampling은 초기의 Bandit 알고리즘이라고 할 수 있다.

여기서는 보상(0과 1만 있는 보상)을 베르누이 분포를 사용해서 생성한다. 이 세팅에서는 분포를 베타 분포로 가정한다. 베르누이 분포와 베타 분포(α, β 를 가지는 분포)가 conjugate한 특성을 가지기 때문이다.

초기에는 알파(1일 확률), 베타(0일 확률)를 모두 1이라고 둔다. 관측 이전은 모든 Q가 1이 되게 된다.

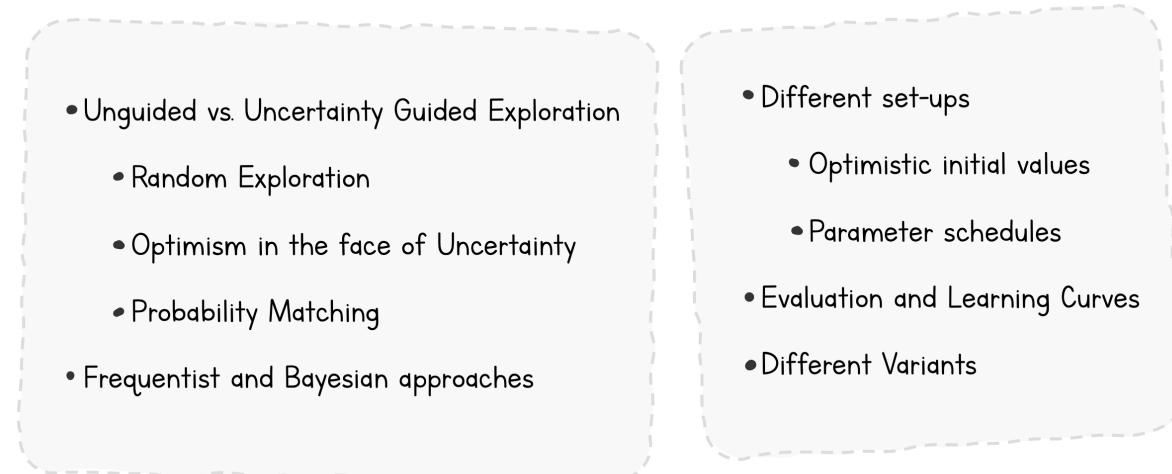
관측 후에 알파와 베타를 업데이트하고, 이에 따라 Q를 업데이트하고, 이걸 다음 최적 액션 선택에 사용한다.



톰슨 샘플링은 사후 분포를 가장 쉽게 적용할 수 있는 방법이다

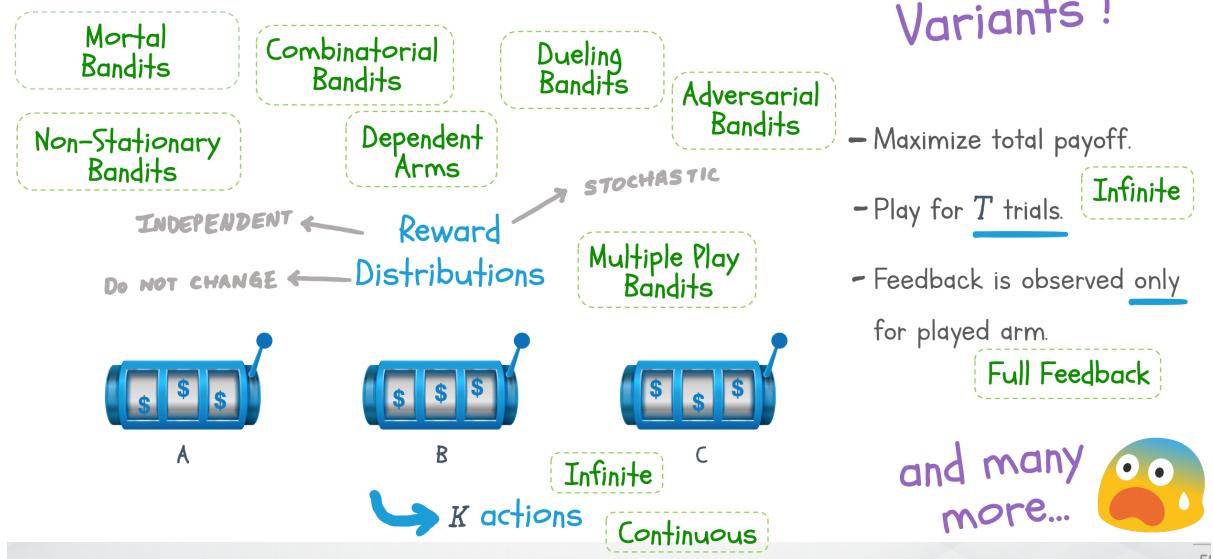
- 각 Q의 사전 분포를 랜덤 샘플링 해 중 가장 좋은 Q를 다음 최적 액션으로 한다

→ 그래서 0.1, 0.75, 0.2 를 얻었다. 따라서 blue를 선택하게 된다.



Multi-Armed Bandit Problem

Variants !

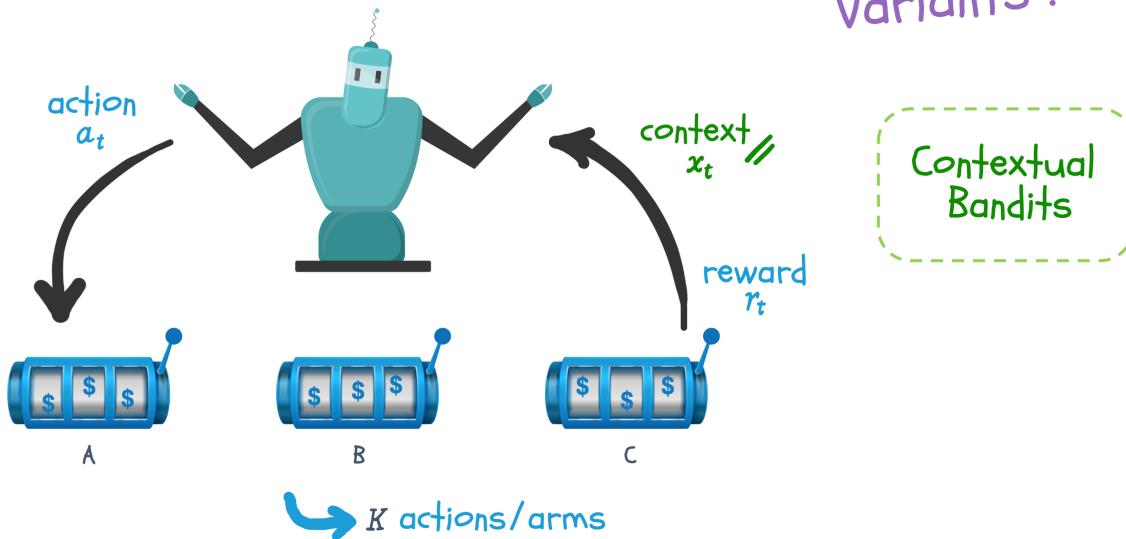


다양한 제약이 있기 때문에 Bandit 문제는 가정이 달라져야 하고 대안들이 필요하다.

- Multiple Play Bandits - 동시에 여러 개의 앰을 당겨야 하는 제약
- Non-Stationary Bandits - 보상의 분포가 시간에 따라 바뀌는 제약

- Adversarial Bandits - 고유한 확률 분포 아닌 다른 모든 암과 경쟁 관계 제약

Multi-Armed Bandit Problem Variants !



추천의 경우에는 Contextual Bandit이라는 제약 조건을 가지는 Bandit.

각 암의 보상이 Context에 따른다. 아이템에 따른 보상(Rating 이 보상이 될수 있다)이 유저에 의존한다.

$$K \text{ actions/arms } a \\ x_t \text{ context } x \in \mathbb{R}^d$$

$$R_a(x_t) = f_a(x_t) + \varepsilon \\ = x_t^\top \theta_a + \varepsilon \\ (\varepsilon - \text{noise/error}) \\ \theta_a \in \mathbb{R}^d$$

Action Value Function

$\hat{Q}_t(x_t, a)$ estimates $\mathbb{E}[R_a | x_t]$

• Least-Squares ridge regression

• Prediction error (uncertainty)

• Training data

$$\hat{Q}_t(x_t, a) = x_t^\top \hat{\theta}_a$$

$$\hat{\theta}_a = (D_a^\top D_a + I_d)^{-1} D_a^\top C_a$$

$$S(x_t, a) = \sqrt{x_t^\top (D_a^\top D_a + I_d)^{-1} x_t}$$

Contextual Bandits

With
Linear
Rewards

LinUCB

$$D_a = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}_{m \times d} \quad C_a = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}_{m \times 1}$$

$$a_t = \arg \max_a (\hat{Q}_t(x_t, a) + \alpha \cdot S(x_t, a)) \\ \downarrow \text{Controls Exploration}$$

Action Selection Strategy

Use $\hat{Q}_t(x_t, a)$ to make decisions

↳ UCB

보상이 Context X에 의해서 생성 된다. 입력 되는 Context X에 따라 R_a 가 노이지한 보상을 생성한다.

Contextual Setting에 대한 구체적인 접근법인 Linear UCB에 대해 소개한다

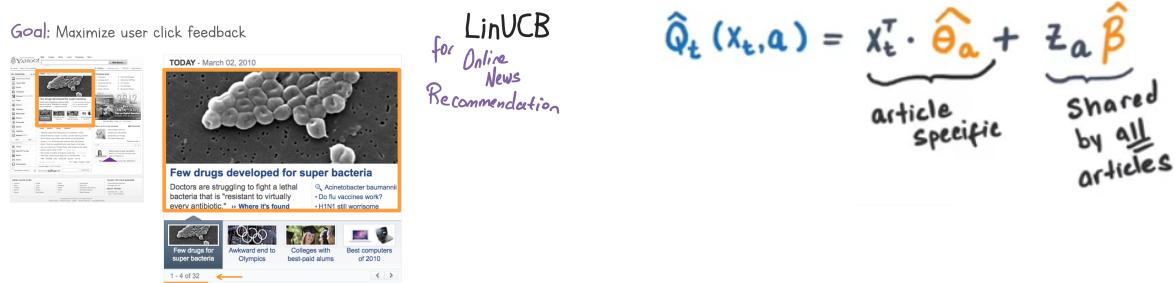
Context X는 D 차원의 벡터이다. R_a 는 선형 함수로 보상을 Context 벡터 x 와 파라미터 벡터 θ_a 간 선형 관계로 생성한다.

모든 암은 고유한 상관 계수인 파라미터 벡터 θ 를 가진다. 주어진 선형 보상에 대한 사전 지식을 가지고 Q도 선형 함수라고 이해할 수 있다. θ 를 추정해서 암에 대한 보상을 추정한다. linear UCB의 최적 액션을 선택 전략은 UCB 전략과 같다.

Q를 사용해서 upper bound가 exploitation과 결합해서 Q에 대한 불확실성을 측정한다. 이것을 S라고 정의한다. 파라미터 α 를 사용해서 exploration을 조절한다.

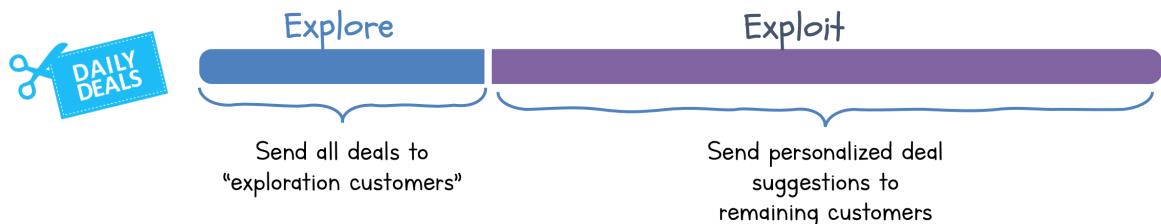
linUCB는 각 암에 대해 선형 회귀로 예측 모델을 만든다. 각 암에 대해 관측한 데이터가 학습 데이터로 관측된 context들이 input feature, 관측된 보상이 label로 한다.

D_a 가 context feature, C_a 가 Context와 함께하는 수집된 보상(label)이다.

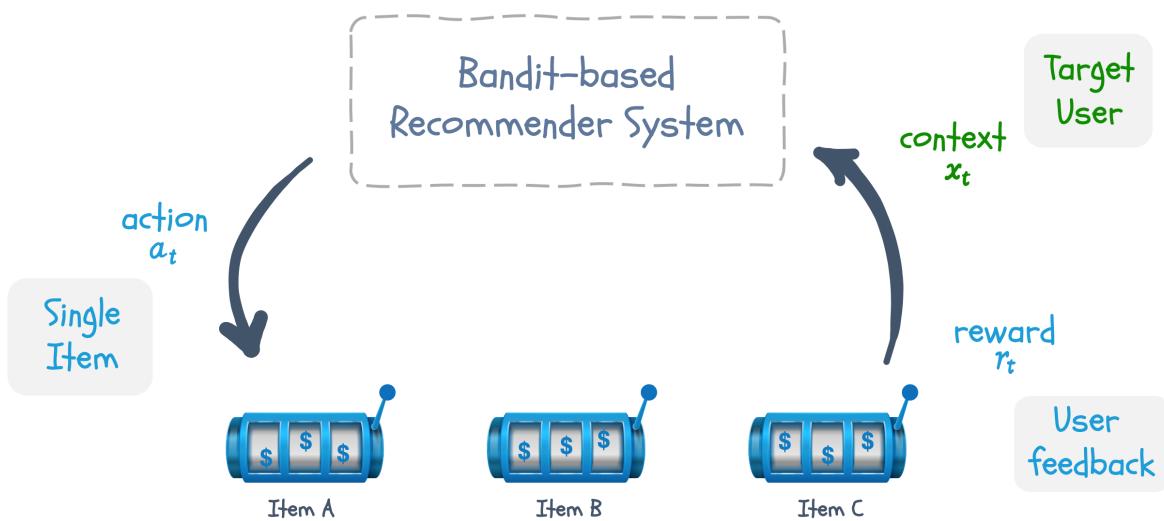


야후에서 Cold Start문제가 있었다. LinUCB를 사용해서 새로운 기사에 대해 어떤 뉴스를 홈페이지의 메인 기사로 사용할지 결정하는데 사용했다. (LinUCB with disjoint linear models), 유저를 세그멘테이션(입력 피처로)하고 뉴스 클릭 했는지 (보상)

Explore then Exploit meets Recommender Systems (RecSys)

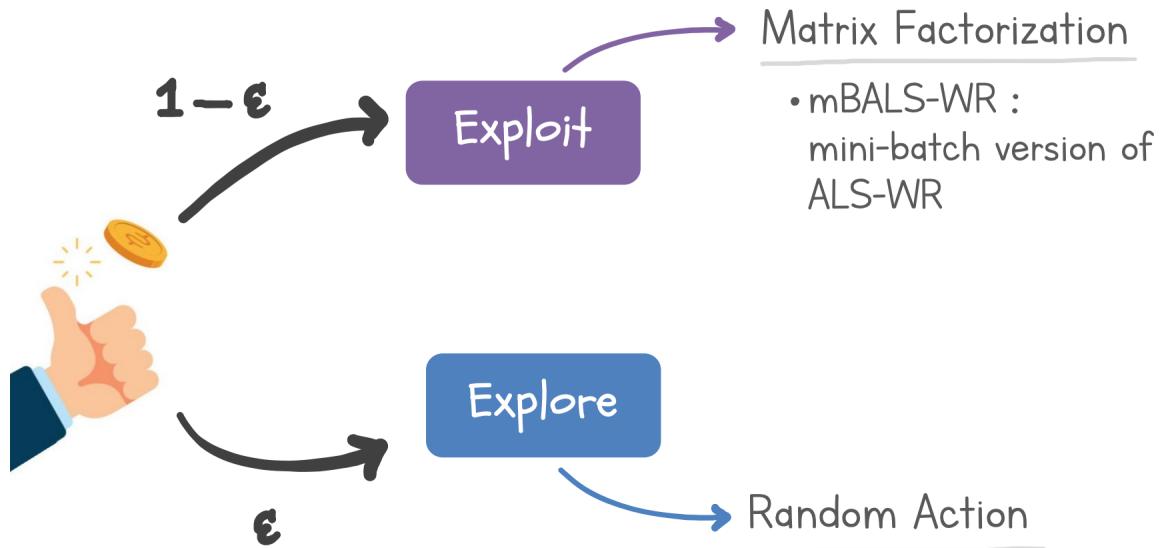


Daily Deal : 전자 쿠폰에 대해 cold start 문제가 있었다. 피드백을 주는 것을 좋아하는 고객들에 대해 Explore를 하고 나머지 유저 들에게는 개인화된 추천을 했다.



arm이 추천할 아이템, action은 아이템 추천, context는 유저 피처, reward는 아이템 추천에 대한 고객의 피드백이라고 생각한다.

Epsilon Greedy + RecSys

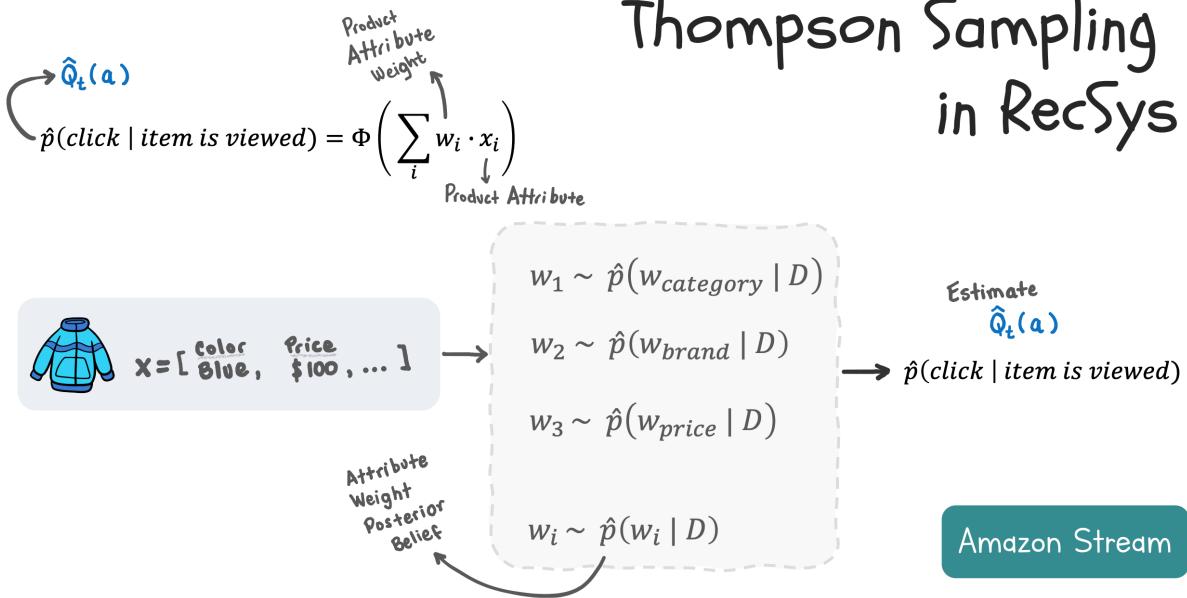


Upper Confidence Bound in RecSys



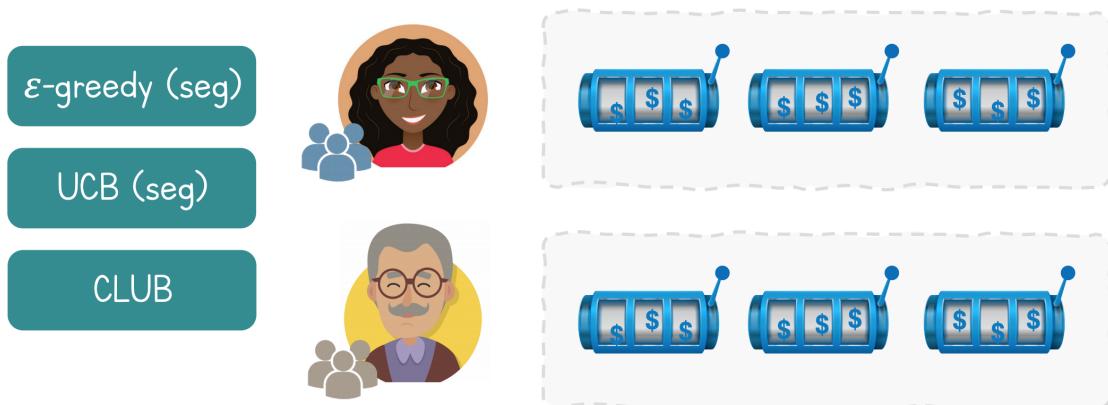
Matrix Factorization과의 결합

Thompson Sampling in RecSys



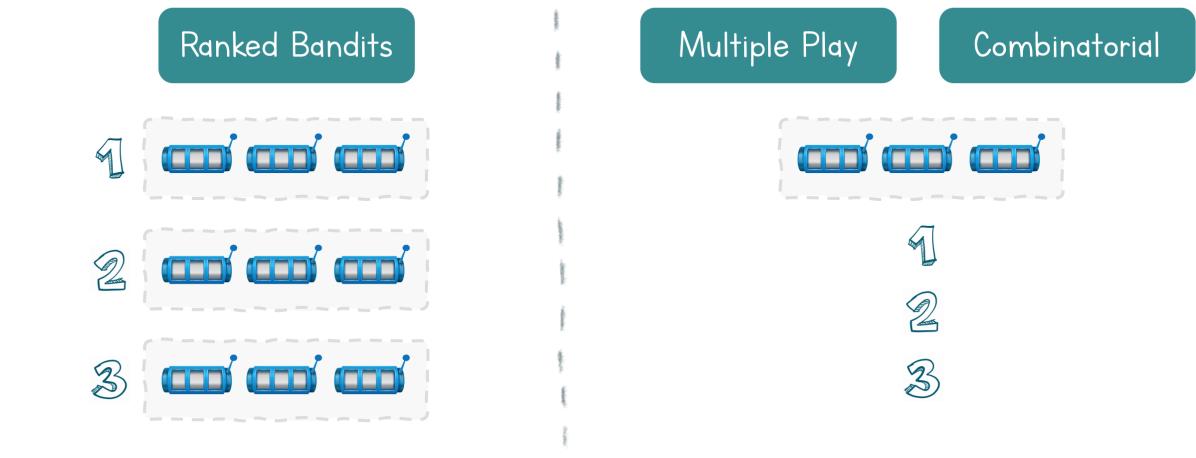
Amazon Stream에서 사용한 Thomson Sampling 기반 추천. 아이템 피처에 대해서 클릭율을 추정한다. 모델이 아이템의 각 피처 가중치를 학습하는데 얼마나 클릭율에 기여하는가에 대해서 다. 각 가중치 자체에 대해 학습할 뿐만 아니라 각 가중치에 대해 불확실성까지 함께 학습한다. 더 많은 데이터가 있으면 불확실성을 함께 학습하기 때문에 학습하는 가중치가 점점 더 확실해진다.

Can we have more
than one bandit?



다중 밴딧 인스턴스를 하나의 솔루션을 사용할 수 있다. 유저들을 세그멘테이션이나 클러스터링하고 각 밴딧을 클러스터마다 학습한다.

What about recommending lists?



밴딧으로 랭킹 리스트의 순위 마다 실행한다.

→ 첫번째 랭킹에 대한 밴딧 하나 두번째 랭킹에 대해 밴딧 하나

아니면 밴딧을 탑다운 구조로 만들어서 첫번째 밴딧이 원가를 잘하는 밴딧을 고르고 골라진 밴딧이 그 아이템에 대해 밴딧 결과를 생성하는 방식으로 한다.