

추천 기술이 마주하고 있는 현실적인 문제들

#Recommender System

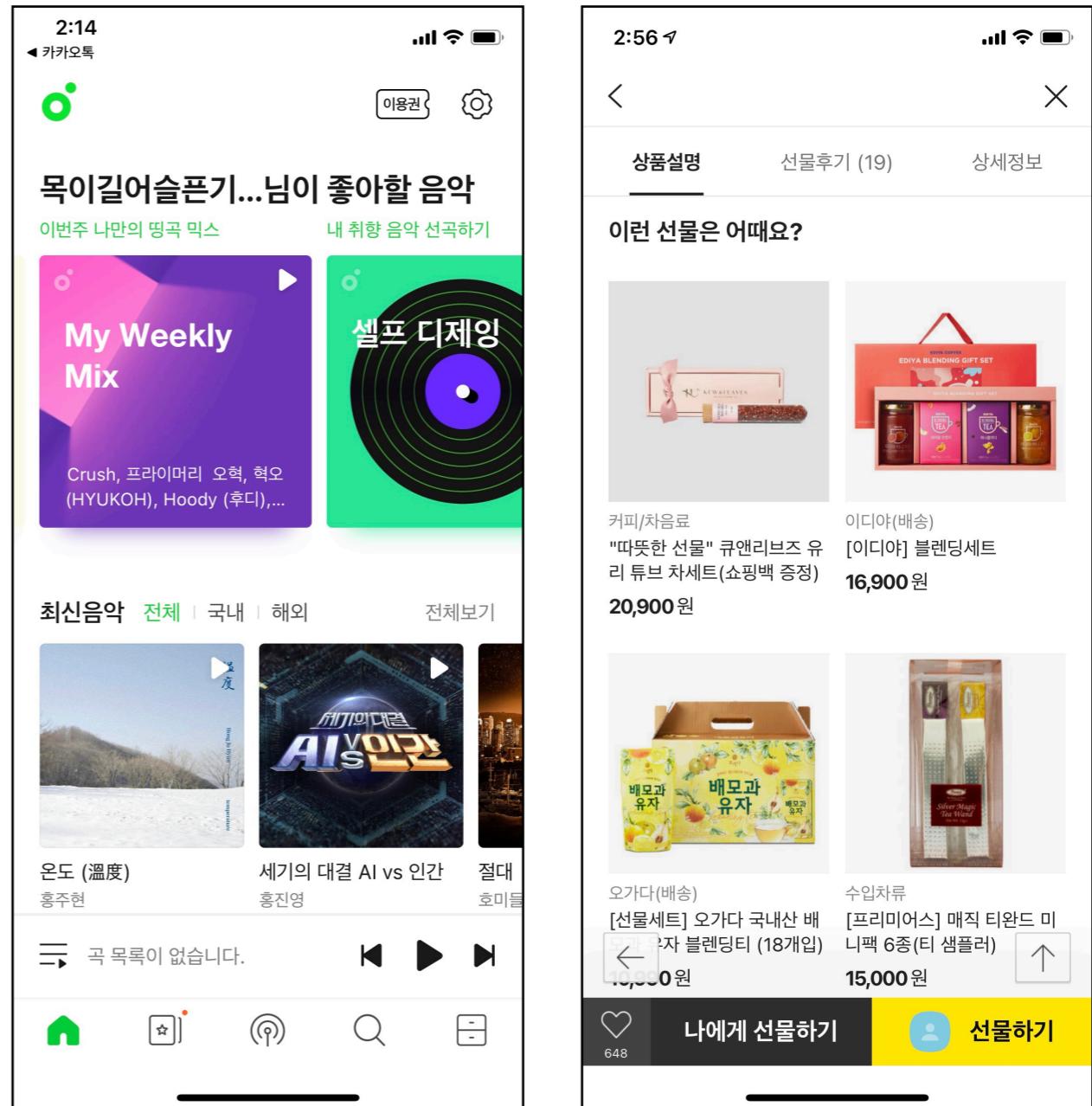
#Exploration #Inference #Offline Evaluation

카카오 추천팀

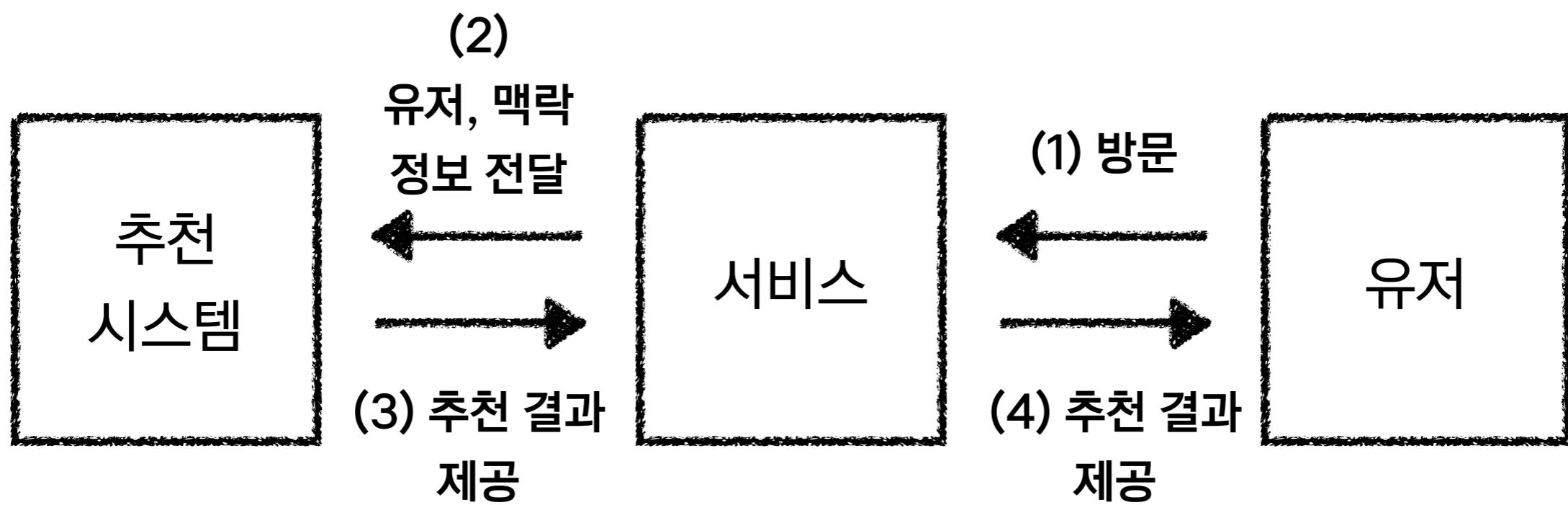
김성진 (nick.kim@kakao.com)

카카오 추천팀

- 카카오의 다양한 서비스에 추천 기능 제공
 - 멜론
 - 미디어다음 뉴스
 - 카카오톡 셀프
 - 카카오페이지
 - 선물하기
 - 픽코마
 - 브런치
 - ...



추천 시스템

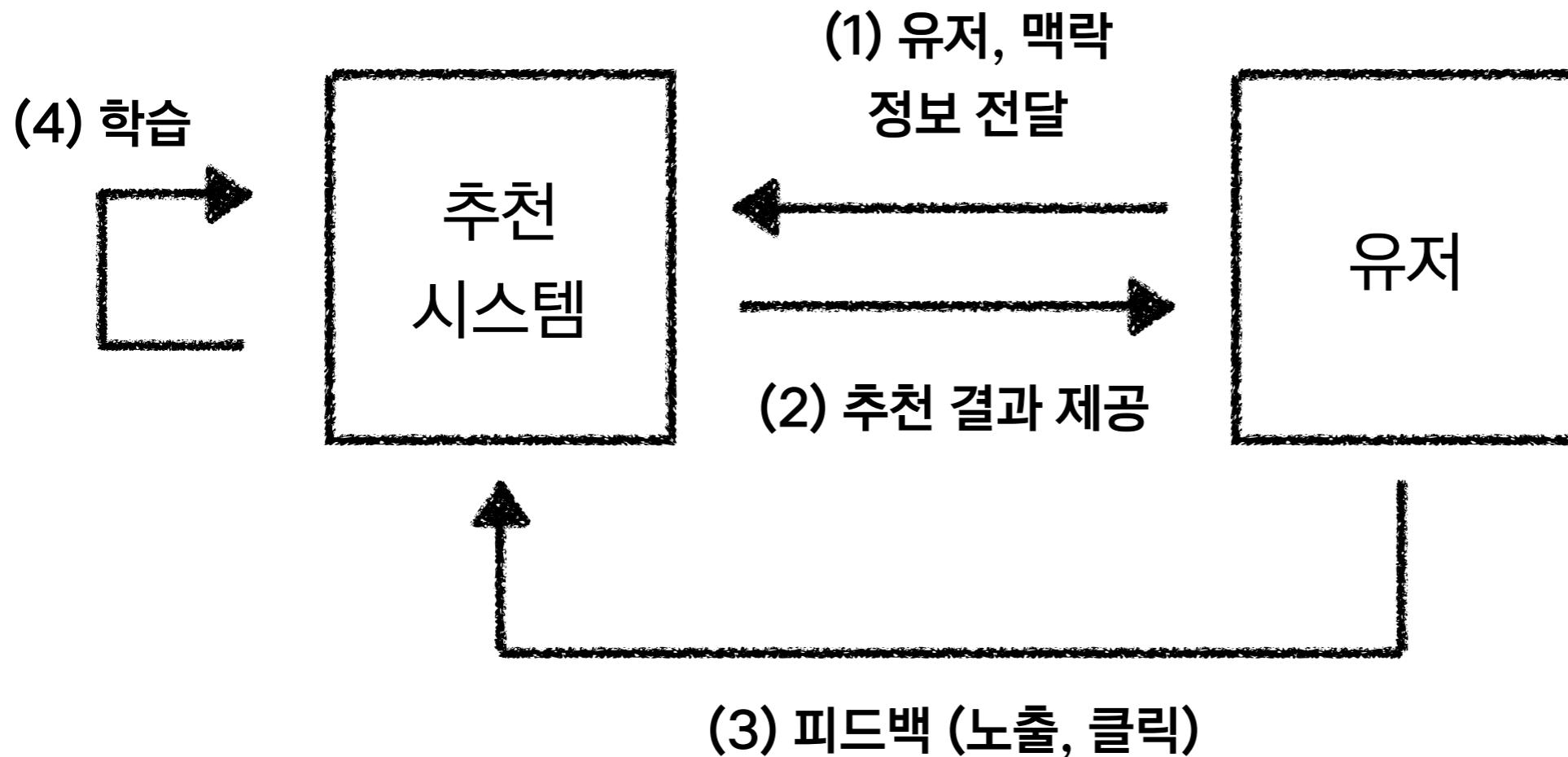


오늘 이야기

- 개인화 추천 시스템이 마주하게 되는 현실적인 문제들
- 연구 환경에서는 문제 정의 편의성, 명료성 등을 위해 생략되는 경우 많음.
- 하지만, 실서비스용 추천 시스템 성능에 큰 영향을 주는 이슈들

Exploration

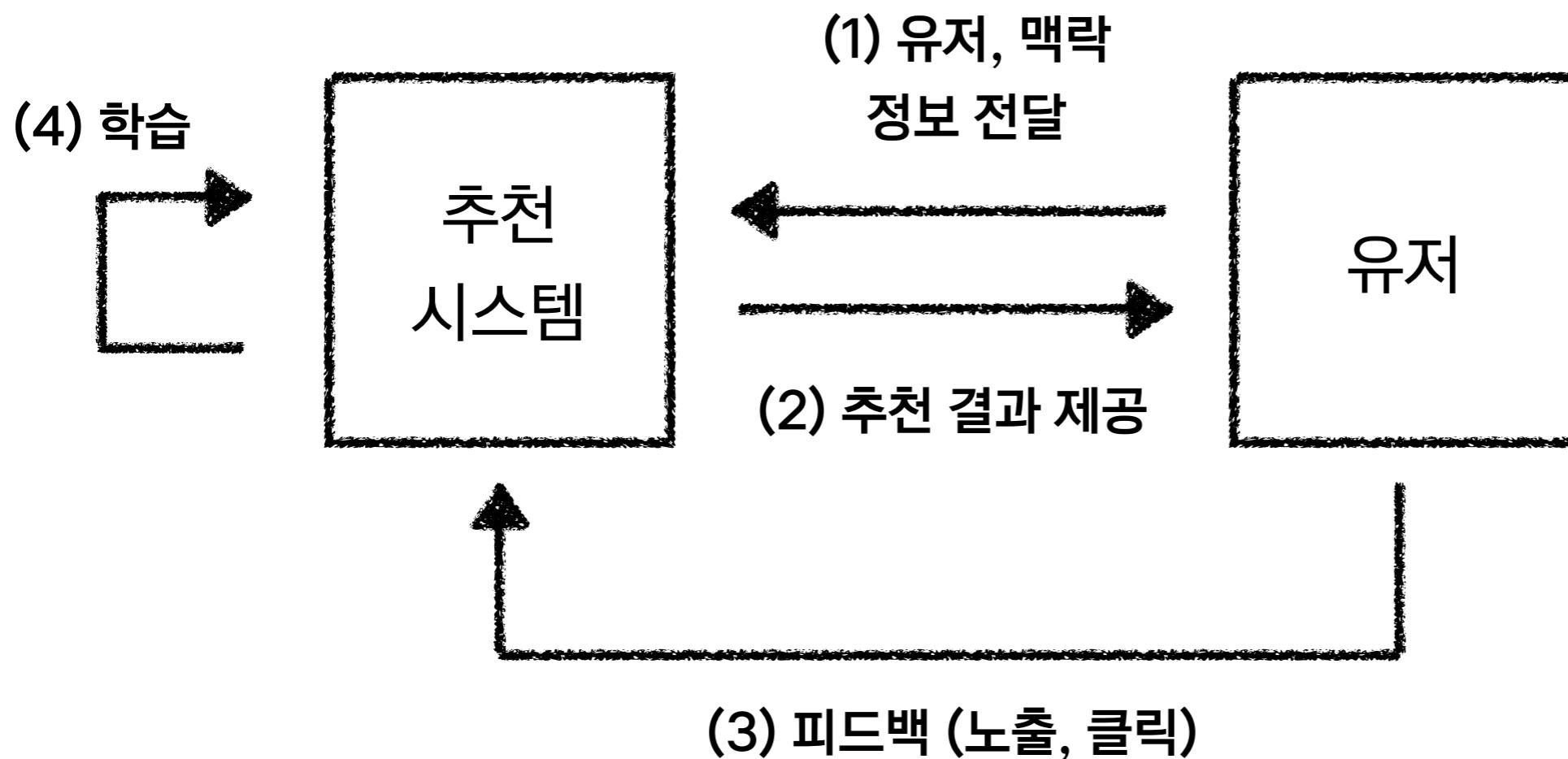
Feedback Loop



Bandit Feedback

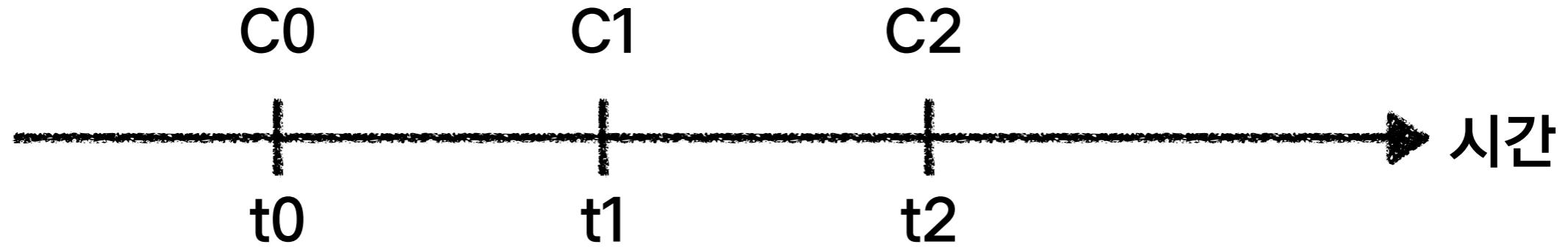
- 알고리즘이 선택한 행동으로 인한 결과 외에는 관측 할 수 없음.
- 추천 시스템이 갖는 주요 특징.
- (참고) Bandit Feedback이 아닌 경우 예시: 투자 (Full Feedback)

많은 경우, 자신이 서빙한 로그 위주로 학습



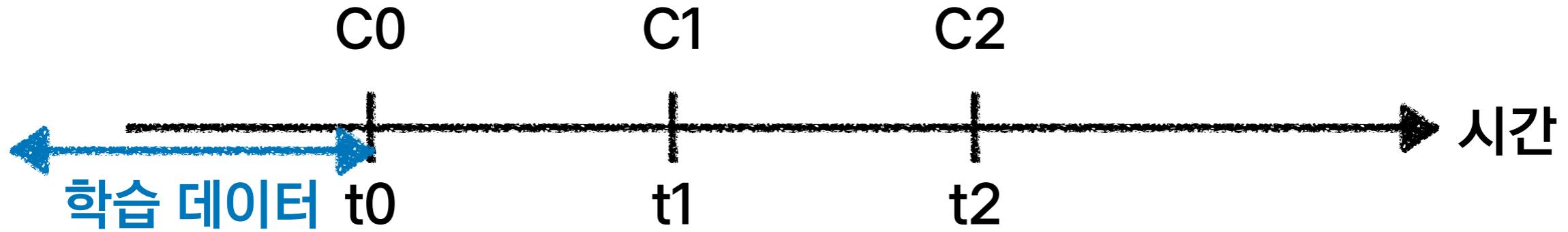
추천 시스템 학습 데이터

- 경우에 따라 서비스 전체 로그를 활용 할 수도 있음.
- 다만, 서비스에서 중요한 지면(예. 첫 진입 화면)의 추천을 담당하게 되면, 거의 대부분의 학습 데이터가 자신이 서빙한 피드백 로그가 됨.
 - 콘텐츠 서비스에서 추천의 역할이 커져가면서 자연스럽게 발생하고 있는 현상.
 - 오늘 이야기는, 자신이 서빙한 로그로만 학습할 수 있는 상황으로 한정해서 이야기함.



가정

- 시간은 $t_0 \rightarrow t_1 \rightarrow t_2 \dots$ 로 흘러간다.
- t_x 시점에 추천할 수 있는 콘텐츠 풀(집합)은 C_x .
 - 예를 들어, t_1 에서의 콘텐츠 풀은 C_1 .
 - \rightarrow 시간이 지남에 따라 콘텐츠 풀이 바뀐다. (오래된 콘텐츠가 제외되고, 신규 콘텐츠가 추가됨.)



가정

- 클릭률을 예측할 수 있는 알고리즘이 있음.
- t_0 시점까지의 로그 데이터로 해당 알고리즘을 학습하고 실서비스 배포.

질문

- t_0 이후에 등장한 신규 콘텐츠에 대해서는 어떻게 추천해야 할까?

“신규 콘텐츠를 추천 할 수 있는가?”는 핵심이 아니다

- 단순히, “신규 콘텐츠를 추천 할 수는 있다.” 만으로는 이 문제를 해결 할 수 없음.
- 신규 콘텐츠 중 인기 있는 콘텐츠를 빠르게 찾아내서 그것을 추천하는 것이 가능해야 함.

Exploration Exploitation Tradeoff

- Multi Armed Bandit (MAB)

(예시) Epsilon Greedy Algorithm

- epsilon의 확률로 -> 무작위 콘텐츠 서빙
- $1 - \text{epsilon}$ 의 확률로 -> Optimal 콘텐츠 서빙

MAB 알고리즘: 이론과 현실 차이

Exploration 관점에서의 주요 이슈

- 시간이 지남에 따라 빠르게 콘텐츠 풀(Action Space)이 바뀜.
- 시간이 지남에 따라 각 콘텐츠의 기대 보상(Expected Reward)이 바뀜.

그 외 이슈

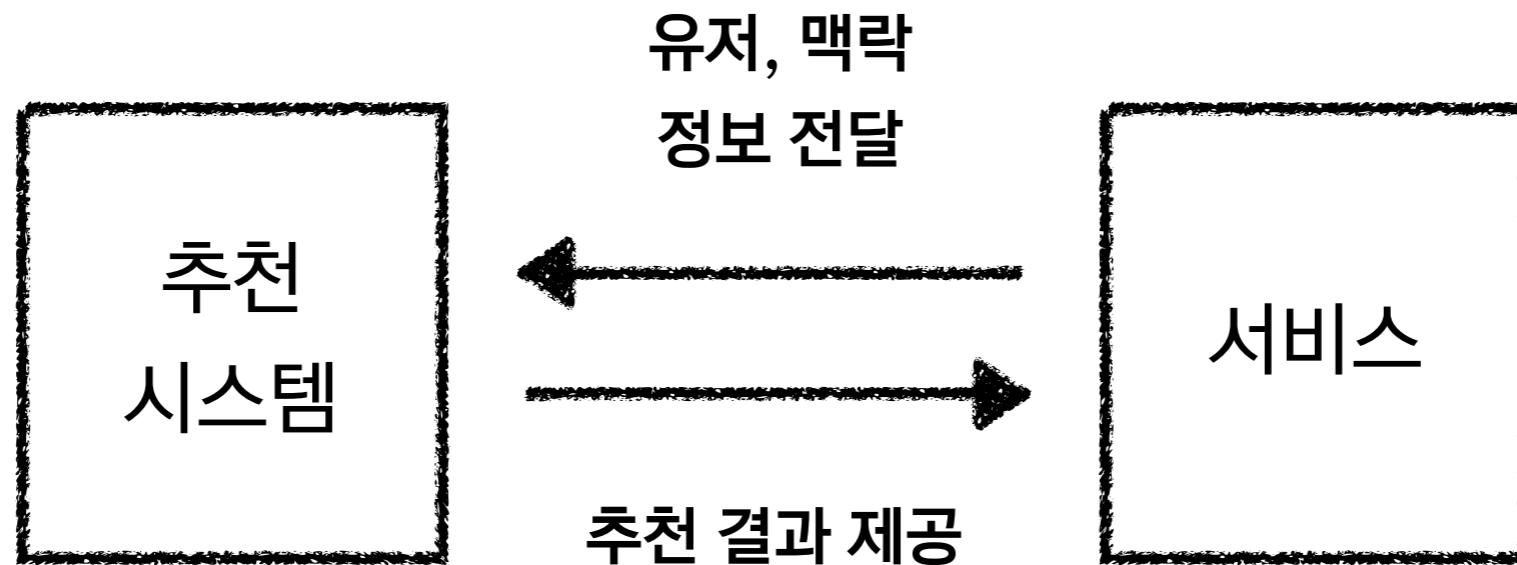
- Single Play vs. Multiple Plays
- 즉각적인 피드백 반영

Exploration: 정리

- 개인화 추천 알고리즘은 Stand alone 상황에서 안정적인 성능을 낼 수 있어야 함.
- 학습 데이터에 있는 콘텐츠만 추천 가능한 알고리즘이라면, 신규 콘텐츠를 서빙하고 그에 대한 피드백 데이터를 확보할 수 있는 Exploration 로직/전략이 필요함.
- Multi Armed Bandit 알고리즘을 이용하여 Exploration Exploitation Tradeoff 문제를 풀 수 있으나, 이론과 현실의 차이는 여전히 존재.

Inference

추천 결과 제공

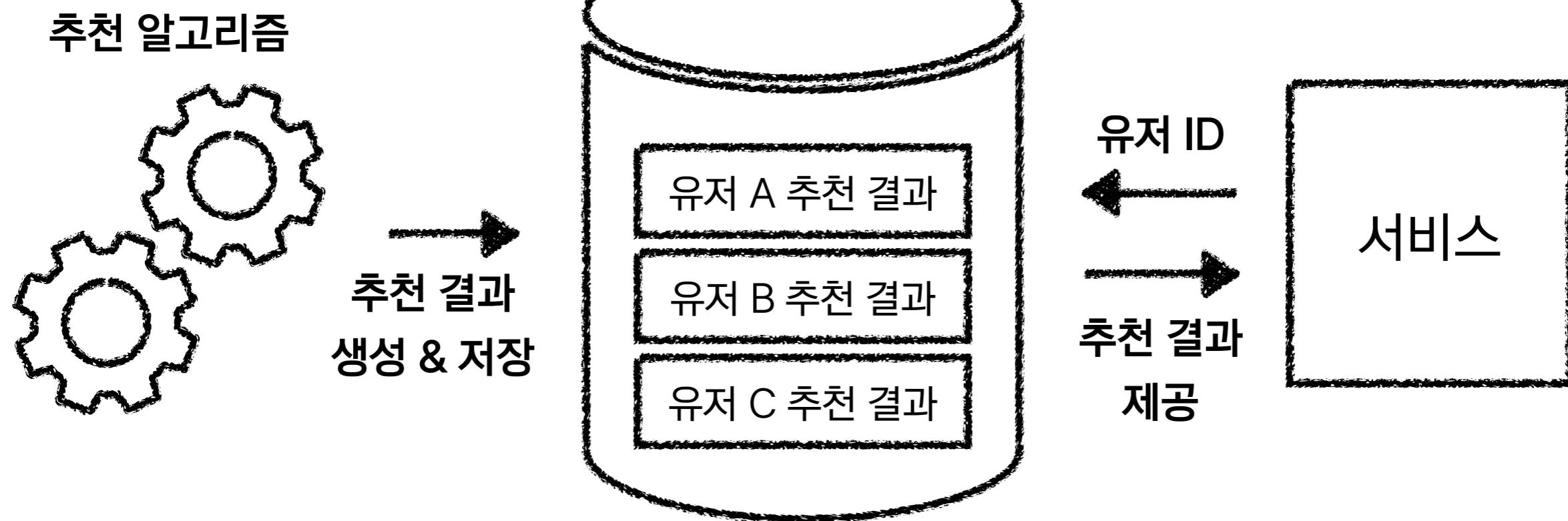


- 추천 결과 제공, 몇 초 안에 만들어줘야 할까?
 - -> 서비스마다 차이가 있지만, 대부분 수십 ms 수준.
 - (추천 결과 만드는 시간, 서비스 로직 적용, 본 것 빼기 등을 모두 포함한 시간.)

대표적인 추천 결과 제공 방식 2가지

- 저장소 기반 (Storage base)
- 실시간 추론 (Real time Inference)

저장소 기반 (Storage base)

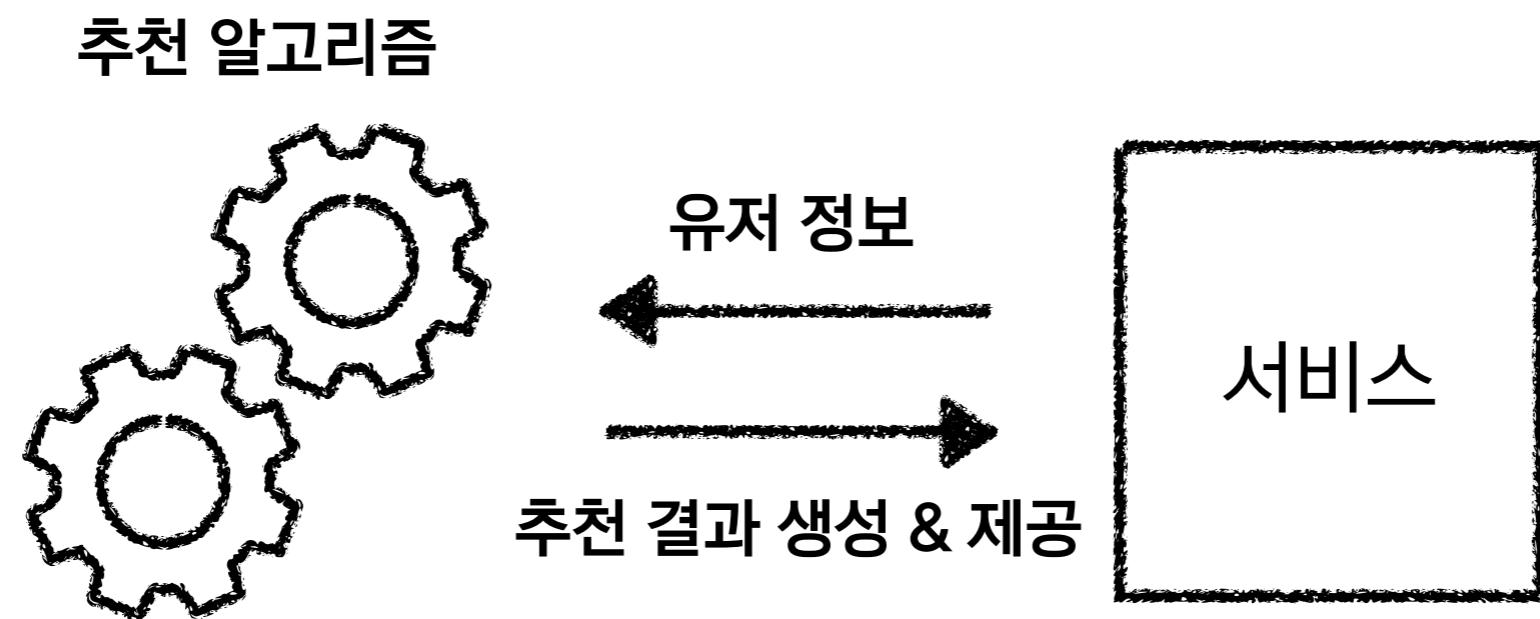


저장소 기반 (Storage base)

추천 결과 생성과 제공이 분리된 형태

- 장점
 - 추천 시스템 개발/운영이 효율적이다.
 - 추천 시스템 설계시 추천 알고리즘 특성에 영향을 받지 않는다.
 - 추천 시스템 응답 속도가 빠르다. (저장소에서 꺼내는 비용은 매우 저렴하므로)
- 단점
 - 유저가 추천 결과를 요청했을 때 일정 수준의 시간차가 있는 추천 결과를 제공할 수밖에 없다. 시간차를 줄이려면 저장소 쓰기(Write) 부하가 증가한다.

실시간 추론 (Real time Inference)



실시간 추론 (Real time Inference)

추천 결과를 요청 받은 시점에 추천 결과를 생성해서 제공함.

- 장점
 - 유저가 추천 결과를 요청한 시점에 맞는 최적의 추천 결과를 제공할 수 있다.
- 단점
 - 시스템 개발/운영 비용이 저장소 기반 대비 크다.
 - 빠른 응답 속도, 안정적인 성능을 내기 위한 최적화 비용이 크다.
 - 새로운 추천 알고리즘 실험 비용이 크다.

실시간 추론: 카카오 추천팀에서는?

- 서비스로직을 제외한 추천 API 응답 속도는 10ms ~ 20ms 이하로 개발.
- Deep Learning Real Time Inference
 - C++ Custom 구현을 통해 TensorFlow Serving 대비 속도 개선
 - Generalization vs. Performance Tradeoff

Inference: 정리

- 저장소 기반이 시스템 운영/개발 효율이 높음.
- 실시간 추론 방식은 ‘실시간’ 자체로 인해 얻는 가치가 큼.
- 실시간 추론을 응답속도 요구 사항에 맞춰 개발하기 어려운 알고리즘은 사용처가 크게 제한됨.
- 별도의 C++ Custom API 구현을 통해 응답 속도를 크게 개선시키는 것도 가능.
 - 다만, 개발 비용은 큼.
 - TensorFlow Serving 처럼 일반화의 강점을 갖는 프레임워크로 빠르게 성능 실험을 해보고, 성능이 입증되면 Custom 구현을 고려해보는 접근도 좋음.

Offline Evaluation

핵심 지표 (Key Metric)

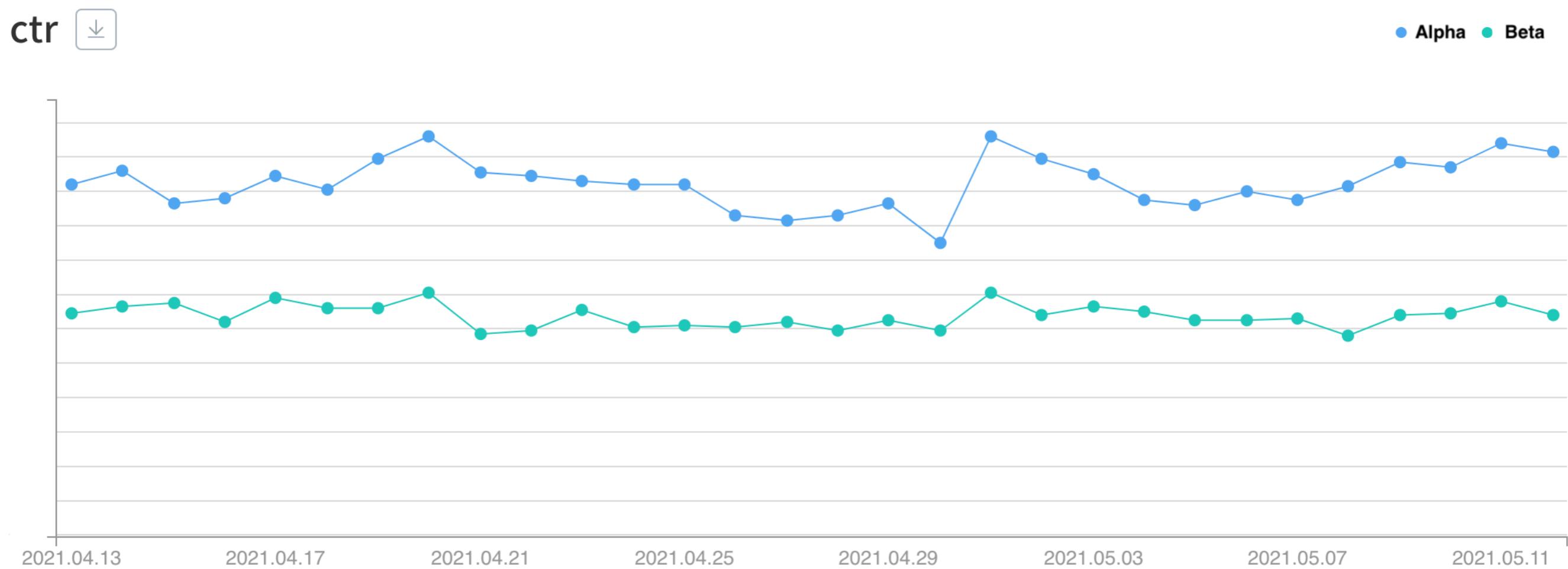
- 서비스에 추천 시스템을 적용할 때, 핵심 지표(Key Metric)를 정한다. 추천 시스템은 해당 지표를 최적화하는 방향으로 행동한다.
- 핵심 지표 예시
 - Click Through Rate (클릭률)
 - Conversion Rate (전환률)

서비스 입장에서 “추천 시스템 도입 효과” 판단 기준

- 대조군 설정: 추천 시스템을 도입하지 못하는 상황에서 사용할 수 있는 최고의 로직.
 - 예시: 인기 콘텐츠 순으로 추천, 에디터 선정 기반 추천
- 대조군 대비 추천 알고리즘이 핵심 지표에서 어느 정도 앞서는지 정량적으로 측정한 결과가 추천 시스템의 성과가 됨.
- 예시: 대조군 대비 클릭률 20% 상승

추천 알고리즘의 우열은 어떻게 판단할까

- 온라인 AB Test 활용



전통적인 오프라인 지표 (NDCG, Recall, ...)

- 이 지표들에서 우수했다는 점이, 온라인 AB Test에서의 우수함을 보장하지 못한다.
- 이유
 - Logging Policy의 Serving Bias
 - 노출(Impression)에 대한 정보가 평가 데이터에 포함되어 있지 않다.
 - Stand alone 상황에서의 성능을 확인하기 어렵다.

그러면, 매번 온라인에서 실험해야 할까?

- 하지만, 온라인 실험은 비용이 크다.
- Off-policy Evaluation이 필요하다.

Off-Policy Evaluation

- Policy A의 히스토리 데이터를 이용해서 Policy B의 성능 평가를 하는 것이 목표.
- Inverse Propensity Score
- Doubly Robust
- Direct Method
- Bias, Variance Tradeoff

Uniform Random Serving Data

- 구할 수만 있으면 유용하게 쓸 수 있음. Logging Policy의 Serving Bias가 없기 때문.
- 현실적으로 Uniform Random 으로 추천 결과를 내보내는 것은 불가능함. (퀄리티 이슈, Search Space 이슈)
- Logging Policy가 Randomize 된 방식이라면 Rejection Sampling을 통해 Uniform Random Distribution에서 생성된 데이터를 모사 가능.*
 - Rejection Sampling으로 인해 데이터양이 크게 줄어드는 단점도 존재함.

* Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms, Lihong Li et al. , WSDM 2011
kakao

Offline Evaluation: 정리

- 추천 알고리즘간 성능 비교는 온라인 AB 테스트를 통해 진행한다.
- 전통적인 추천 시스템 오프라인 지표들(NDCG, Recall, ...)은 온라인 AB 테스트에 서의 우열 여부를 보장하지 못한다.
- 새로운 로직 실험이나 Hyper Parameter Tuning이 필요할때마다 매번 온라인 테스트를 하는 것은 비용이 크다.
- Off-Policy Evaluation이 필요하다.
- Uniform Random Data를 모사해내는 방법도 가능하다.

요약

요약

- 개인화 추천 알고리즘은 Stand alone 상황에서도 안정적인 성능을 낼 수 있어야 한다. 새로 추가되는 콘텐츠 중 최적의 콘텐츠를 빠르게 찾아낼 수 있어야 한다.
- 추론 속도가 느린 알고리즘은 실서비스 적용이 어렵다.
- 온라인 실험 비용은 크기 때문에, Off-Policy Evaluation을 통해 온라인 실험 비용을 낮출 필요가 있다.

감사합니다