



# A Gentle Introduction to Recommendation as Counterfactual Policy Learning

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/87d15cab-d8ca-4123-83a2-cd664af526fb/BanditRecoCourse\\_part1.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/87d15cab-d8ca-4123-83a2-cd664af526fb/BanditRecoCourse_part1.pdf)

## Part 1. Recommendation via Reward Modeling

### 1.1 Classic vs Modern : Recommendation as AutoComplete vs. Recommendation as Intervention Policy

확률의 개념으로서 첫번째 뭘 봤는지, 두번째 뭘 봤는지에 따라서 세 번째 아이템을 뭘 봤는지를 확률  
가정한다면 다음에 어떤 아이템을 봤는지 예측하고 그 아이템을 추천한다면, 그 아이템의 클릭할 확률도 최대  
가 될 것.

Classic Approach - AutoComplete

## Next Item Prediction: Hold out $V_3$

$$P(V_3 = \text{phone A} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.27$$

$$P(V_3 = \text{phone B} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.24$$

$$P(V_3 = \text{rice} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.10$$

$$P(V_3 = \text{couscous} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.09$$

$$P(V_3 = \text{beer} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.30$$

What actually happened?  $V_3 = \text{phone B}$

Our model assigned:  $P(V_3 = \text{phone B} | V_1 = \text{beer}, V_2 = \text{phone A}) = 0.24$

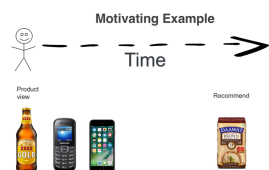
### Conditional Distribution

실제로 다음에 조회한 아이템에 대한 높은 확률을 부여하는 것이 Classic한 추천에서 좋은 것이라고 평가함  
다섯 가지 확률 변수에 대해서 계산한다. Evaluation Metric을 Recall @K를 사용하면, K=3에 실제 조회한  
아이템(Phone B)라고 했을 때 Evaluation

여기서 말하는 Issue는 사실 이런 건 추천이랑 관계가 없다고 말함. 이걸 사실 유저의 historical action 일 뿐  
이다. 그저 유저의 행동을 모방할 뿐이다. 실제로 아직 추천에서 좋을 지 나쁠지는 모른다.

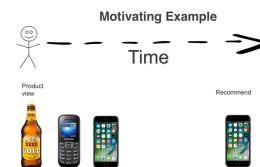
여기서 하는 모든 것에 Reward는 클릭으로 즉시 얻어지고 Action은 추천이다.

### The Real Reco Problem



$$P(C_4 = 1 | A_4 = \text{rice}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone B})$$

### The Real Reco Problem



$$P(C_4 = 1 | A_4 = \text{phone B}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone B})$$

그림과 같이 실제 추천에서 고려해야 할 문제는 C는 클릭으로 이루어진 문제다.

## What we have vs. what we want

$$\begin{aligned}
 P(V_4 = \text{phone A} | V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= 0.27 \\
 P(V_4 = \text{phone B} | V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= 0.28 \\
 \text{Our model: } P(V_4 = \text{rice} | V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= 0.12 \\
 P(V_4 = \text{couscous} | V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= 0.14 \\
 P(V_4 = \text{beer} | V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= 0.19
 \end{aligned}$$

Which is a vector of size  $P$  that sums to 1.

We want:

$$\begin{aligned}
 P(C_4 = 1 | A_4 = \text{phone A}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= ? \\
 P(C_4 = 1 | A_4 = \text{phone B}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= ? \\
 P(C_4 = 1 | A_4 = \text{rice}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= ? \\
 P(C_4 = 1 | A_4 = \text{couscous}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= ? \\
 P(C_4 = 1 | A_4 = \text{beer}, V_1 = \text{beer}, V_2 = \text{phone A}, V_3 = \text{phone A}) &= ?
 \end{aligned}$$

Which is a vector of size  $P$  where each entry is between 0 and 1.

과거 추천은 유저가 다음에 어떤 행동을 할지 예측하는 것에 중점을 두었는데 실제 유저의 응답을 함께 고려해야 한다.

과거에는 아이템 끼리 서로 제약을 해서 각 아이템에 대한 추천의 확률의 합이 10이 되었는데 여기서 정의하는 문제로 보면 추천을 해야할 아이템 끼리 서로 제약을 가지지 않는다.

## Implicit Assumption

If:

$$P(V_n = a | V_1 = v_1 \dots V_{n-1} = v_{n-1}) > P(V_n = b | V_1 = v_1 \dots V_{n-1} = v_{n-1})$$

Assume:

$$\begin{aligned}
 P(C_n = 1 | A_n = a, V_1 = v_1 \dots V_{n-1} = v_{n-1}) \\
 > P(C_n = 1 | A_n = b, V_1 = v_1 \dots V_{n-1} = v_{n-1})
 \end{aligned}$$

Vast amounts of academic Reco work assumes this implicitly!

대 다수의 학회의 추천은 위와 같은 다음에 조회할 아이템  $a, b$ 에서  $a$ 가 더 높다면,  $a, b$ 를 추천했을 때도  $a$  추천의 클릭 확률도 높아질 것이라고 가정한다.

## Classical vs. Modern Recommendation

- **Classical Approach:** Recommendation as *Autocomplete*:
  - Typically leverage *organic* user behaviour information (e.g. item views, page visits)
  - Frame the problem either as *missing link prediction/MF* problem, either as a *next event prediction, sequence prediction* problem
- **Modern Approach:** Recommendation as *Intervention Policy*:
  - Typically leverage *bandit* user behaviour information (e.g. ad clicks)
  - Frame the problem either as *click likelihood* problem, either as a *contextual bandit/policy learning* problem

Organic - 추천 시스템 배치 전의 유저가 시스템과 상호 작용

Intervention - 사이에서 조정 및 중재

### 1.1.1. Advantages of the Classical Approach

#### 학계의 관점에서

- 이미 체계가 잡혀있는 framework
- 이미 많은 표준 데이터 셋과 메트릭
- publish가 쉽고 비교가 쉽다

#### 실제에서

- application을 유저가 사용함으로 데이터가 자연스럽게 얻어진다
- 초기 시스템으로 굉장히 좋다
- 단지 최고가 아닐 뿐이지 최악은 아니다

### 1.1.2. limitation of the Classical Approach

Classic한 문제 정의는 최고의 추천 정책이 유저 행동을 auto-complete 하는 것이라 가정 하는데 있다고 한다.

비즈니스 적인 관점을 볼 때, 사용 가능한 유저 피드백이 없는 초기에는 좋지만, 시간이 지날 수록 추천 최적화와 비즈니스는 멀어지게 된다

## Solving the Wrong Problem: are the Classic Datasets Logs of RecSys?

- MovieLens: no, explicit feedback of movie ratings [1]
- Netflix Prize: no, explicit feedback of movie ratings [2]
- Yoochoose (RecSys competition '15): no, implicit session-based behavior [3]
- 30 Music: no, implicit session-based behavior [4]
- Yahoo News Feed Dataset: **yes!** [5]
- Criteo Dataset for counterfactual evaluation of RecSys algorithms: **yes!** [6]

The Criteo Dataset shows a log of recommendations and if they were successful in getting users to click.

대표적인 추천 학습 데이터 셋에 진짜 추천 시스템의 로그가 들어간 게 아니라 explicit feedback만 담겨 있음  
criteo와 yahoo 를 제외하고는 ... evaluation에 적절하지 않다.

## Solving the Wrong Problem: do the Classical Offline Metrics Evaluate the Quality of Recommendation?

### Classical offline metrics:

- Recall@K, Precision@K, HR@K: How often is an item in the top k - no, evaluates next item prediction
- DCG: Are we assigning a high score to an item - no, evaluates next item prediction


### Online metrics and their offline simulations:

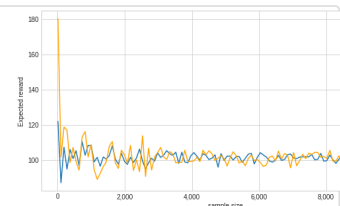
- A/B Test: i.e. run a randomized control trial live — yes, but expensive + the academic literature has no access to this
- Inverse Propensity Score estimate of click-through rate: - to be explained later. — yes! (although it is often noisy) [7]

Criteo 데이터셋이 Inverse Propensity Score를 가지고 만든 데이터셋임.

#### How To Use Counterfactual Evaluation To Approximate Online AB Test Results

In this article, I will explain a principled approach to estimate the expected performance of a model in an online AB test using only offline data. This is very useful to help decide which set of model enhancements that should be prioritized to be validated using online AB test.

 <https://medium.com/towards-artificial-intelligence/how-to-use-counterfactual-evaluation-to-approximate-online-ab-test-results-a1f29d6963a1>



기존 Recall/Precision의 offline test는 배포 전에 테스트를 한다는 장점이 있지만, 실제 비즈니스를 충분히 고려하지 못했다는 평가가 있다. 그래서 그것이 실제 성능을 보장하지 못한다는 해석을 할 수 있다.

기존 추천의 결과를 로깅된 추천 정책으로 보고 새로운 정책을 비교해서 A/B 테스트를 오프라인으로 진행하는 방법중 하나가 IPS 이다.

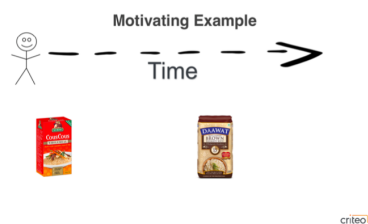
## Offline Evaluation that Predicts an A/B Test Result

Let's follow the previous notation ( $A$  action/recommended item,  $V$  organic item view/interaction) and denote  $\pi$  the recommendation policy (that assign probabilities to items conditionally on user past). Imagine we have a new recommendation policy  $\pi_t(A_n = a_n | V_1 = v_1, \dots, V_n = v_{n-1})$ , can we predict how well it will perform if we deploy it? We collected logs from a different policy:

$$\pi_0(A_n = a_n | V_1 = v_1, \dots, V_n = v_{n-1})$$

Let's examine some hypothetical logs...

## Offline Evaluation via IPS



Imagine the user clicks

$$\pi_t(A_2 = \text{rice} | V_1 = \text{couscous}) = 1$$

$$\pi_0(A_2 = \text{rice} | V_1 = \text{couscous}) = 0.01$$

The new policy will recommend “rice” the  $\frac{1}{0.01} = 100\times$  as often as the old policy.

1은 100%로 0.01은 1%로 100배 정도 차이가 생겨 분산을 커지게 만든다.

## Offline Evaluation IPS

Let's denote  $c_n$  the feedback on the  $n^{th}$  recommendation, i.e. 1 if the recommendation got clicked else 0.

Let's re-weight each click by *how much more likely the recommendation will appear under the new policy  $\pi_t$  compared the logging policy  $\pi_0$*

We will assume that  $\mathbf{X} = f(v_1, \dots, v_n)$ . Crafting  $f(\cdot)$  is often called "feature engineering".

$$\begin{aligned}\text{CTR estimate} &= \mathbb{E}_{\pi_0} \left[ \frac{c \cdot \pi_t(\mathbf{a}|\mathbf{X})}{\pi_0(\mathbf{a}|\mathbf{X})} \right] \\ &\approx \frac{1}{N} \sum_n^N \frac{c_n \pi_t(\mathbf{a}|\mathbf{X})}{\pi_0(\mathbf{a}_n|\mathbf{X})}\end{aligned}$$

로깅 정책  $\pi_0$ 과 비교해서 새로운 정책  $\pi_t$ 이 추천에 얼마나 더 나올 지 클릭할 때마다의 가중치를 적용해보면

This estimator is unbiased

$$\text{CTR estimate} = \mathbb{E}_{\pi_0} \left[ \frac{c \cdot \pi_t(\mathbf{a}|\mathbf{X})}{\pi_0(\mathbf{a}|\mathbf{X})} \right] = \mathbb{E}_{\pi_t} [c]$$

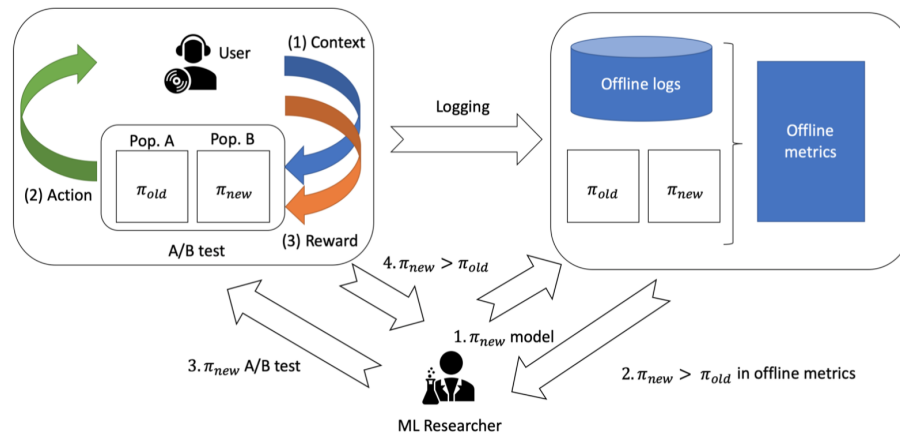
### IPS의 단점

- 우리는 클릭 될 때 만을 보고 있음
- 새로운 정책의 추천 결과가 이전 정책의 추천 결과와 많이 다를 때는, 가중치가 높아지게 된다 ( 샘플 에서 보기 드문 경우라는 사실을 보완해야 하기 때문에 ). 이 높아진 가중치는 추정기의 분산에 많은 영향을 끼치게 된다.

As IPS is unbiased, it works better than any other biased estimator in general. However, IPS suffers a **high variance** if the random variable  $\frac{\pi_t(a|x)}{\pi_0(a|x)}$  in the denominator is very big as  $\pi_t(a|x)$  is big while  $\pi_0(a|x)$  is small. Because we can estimate  $V$  only once as we have one sample, the high variance makes our estimate far from the true value, which could make a little bit biased estimator with low variance better than IPS.

따라서, IPS는 실제와 관련된 질문에 대답할 수 있지만, logging 정책과 많이 다른 선택을 새로운 정책이 선택했을 때, 분산이 커지는 문제가 있어서 극적인 결과보다 더 적은 결과를 얻는 단점이 있다

## Recommendation as Supervised Learning and A/B Testing



1. 과거 유저 활동 데이터로 새로운 모델을 학습한다.
2. 오프라인 테스트를 하고 A/B 테스트를 할지 결정한다
3. A/B 테스트 한다
  - 결과가 좋으면 → 사용한다.
  - 결과가 안 좋으면 → 원인을 파악하고 1번으로 간다
4. 전 과정을 반복한다.

## RecoGym을 보기 전에 의문?

우리는 강화 학습을 직접 하려고 한다.

게다가, 지도 학습 프레임워크를 사용해서 강화 학습을 하려고한다.

표준 데이터 셋으로는 추천 시스템의 특성을 explore할 수 없다.

→ 그럼 근데 어떻게 강화 학습 알고리즘을 오프라인 테스트 하는가?

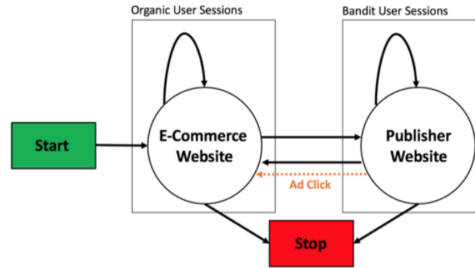
## OpenAI Gym

- Problem : RL 알고리즘을 비교해주는 공통 tool set이 부족
- Python API 의 형태로 RL 알고리즘을 성능 비교를 할 수 있다
- Environment - 문제 정의, Agents (RL 알고리즘)



## Introducing RecoGym

**RecoGym:** An OpenAI compatible environment that simulates user behavior (both organic and bandit, e.g. its reaction to recommendation agents)



- 새로운 추천 정책을 가상 환경에서 online evaluation을 하게 한다
- 전체 관점에서 ( organic , bandit, organic + bandit ) 유저의 피드백을 볼 수 있다.
- 파라미터를 조정해서 원하는 경우를 interpolate 해서 환경을 만들 수 있다.
- 환경을 시뮬레이션 해서
- <https://arxiv.org/pdf/1808.00720.pdf>

### Algorithm 1: A simple Simulator

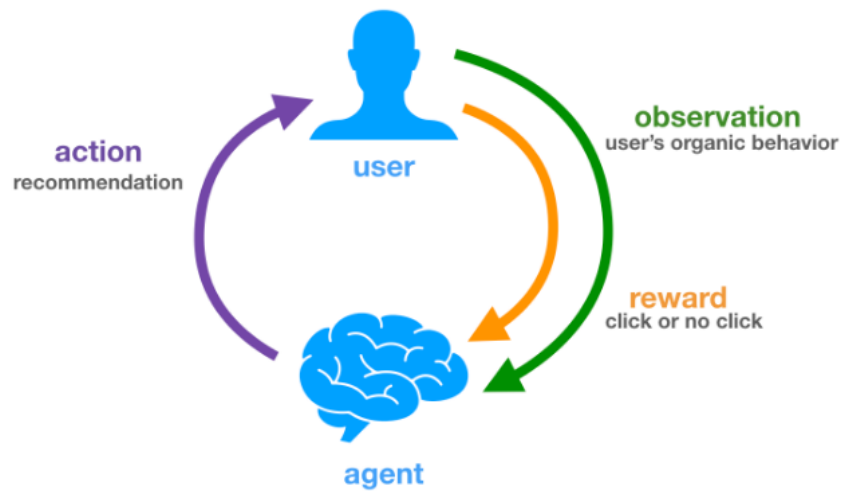
**Input** :  $S \in \mathcal{R}^{3 \times 3}$  transition matrix between organic and bandit,  $\Gamma \in \mathcal{R}^{P \times K}$  organic embeddings,  $\mu_\Gamma$  organic popularity  $\beta \in \mathcal{R}^{P \times K}$  bandit embeddings,  $\mu_\beta$  non-personalised ctr contribution  $f(\cdot)$  monotonic increasing function accounting for ad fatigue  $m, U$  number of users,  $P$  number of products.

**Output**: Sequence of organic and bandit events

```

1 for  $u \in 1..U$  do
2    $t \leftarrow 0$ 
3    $z_{u,0} \leftarrow \text{organic}$ 
4    $r_{u,0} \leftarrow \text{undef}$ 
5    $c_{u,0} \leftarrow \text{undef}$ 
6    $\omega_{u,0} \sim \mathcal{N}(0_{K \times 1}, I_K)$ 
7    $v_{u,0} \sim \text{Categorical}(\text{softmax}(\Gamma \omega_{u,0}))$ 
8   while  $z_{u,t} \neq \text{stop}$  do
9      $t \leftarrow t + 1$ 
10     $\omega_{u,t} \sim \mathcal{N}(\omega_{u,t-1}, \sigma_\omega^2 I_K)$ 
11    if  $c_{u,t-1} \neq 1$  then
12       $z_{u,t} \sim \text{Categorical}(S_{z_{u,t-1}, \text{organic}}, S_{z_{u,t-1}, \text{bandit}}, S_{z_{u,t-1}, \text{stop}})$ 
13    else
14       $z_{u,t} = \text{organic}$ 
15    end
16    if  $z_{u,t} = \text{organic}$  then
17       $v_{u,t} \sim \text{Categorical}(\text{softmax}(\Gamma \omega_{u,t} + \mu_\Gamma))$ 
18       $r_{u,t} \leftarrow \text{undef}$ 
19       $c_{u,t} \leftarrow \text{undef}$ 
20    end
21    if  $z_{u,t} = \text{bandit}$  then
22       $r_{u,t}$  is generated from the policy
23       $c_{u,t} \sim \text{Bernoulli}([f(\beta \omega_{u,t} + \mu_\beta)]r_{u,t})$ 
24    end
25  end
26 end
  
```

## Getting Started




유저의 organic 행동을 observe 하고, Action이라는 추천을 한다. 그리고 유저의 추천에 클릭 피드백을 받고

시간이 흐를 수록 보상을 최대화 한다. → 새로운 시스템의 Goal.


- 1 Offline Learning**  
fixed policy chooses **action**,  
environment reveals **observation** and **reward**  
agent learns from data
- 2 Online Learning**  
agent chooses **action**,  
environment reveals **observation** and **reward**  
agent continues learning

Google Colaboratory

 <https://colab.research.google.com/github/criteo-research/bandit-reco/blob/master/notebooks/0.%20Getting%20Started.ipynb>




Google Colaboratory

 <https://colab.research.google.com/github/criteo-research/bandit-reco/blob/master/notebooks/1.%20Organic%20vs%20Bandit%20Best%20Ofs.ipynb>



criteo-research/bandit-reco

Permalink GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together. Millions of developers and companies build, ship, and maintain their software on GitHub -

 <https://github.com/criteo-research/bandit-reco/blob/master/notebooks/Solutions/1.%20Organic%20vs%20Bandit%20Best%20Ofs.ipynb>



[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/07768814-164d-4cdd-a4f2-ee3048e2c1a7/0.\\_Getting\\_Started.ipynb\\_-\\_Colaboratory.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/07768814-164d-4cdd-a4f2-ee3048e2c1a7/0._Getting_Started.ipynb_-_Colaboratory.pdf)

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ea10fb1d-cf15-4fe4-818a-84788893c581/1.\\_Organic\\_vs\\_Bandit\\_Best\\_Ofs.ipynb\\_-\\_Colaboratory.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ea10fb1d-cf15-4fe4-818a-84788893c581/1._Organic_vs_Bandit_Best_Ofs.ipynb_-_Colaboratory.pdf)