A Contextual Bandit Approach to Personalized News Article Recommendation

https://www.miricanvas.com/v/14vqgb8

> Compared to machine learning in the more standard supervised setting, evaluation of methods in a contextual bandit setting is frustratingly difficult. Our goal here is to measure the performance of a *bandit algorithm* π, that is, a rule for selecting an arm at each time step based on the preceding interactions (such as the algorithms described above). Because of the interactive nature of the problem, it would seem that the only way to do this is to actually run the algorithm on "live" data. However, in practice, this approach is likely to be infeasible due to the serious logistical challenges that it presents. Rather, we may only have *offline* data available that was collected at a previous time using an entirely *different* logging policy. Because payoffs are only observed for the arms chosen by the logging policy, which are likely to often differ from those chosen by the algorithm π being evaluated, it is not at all clear how to evaluate π based only on such logged data. This evaluation problem may be viewed as a special case of the so-called "off-policy evaluation problem" in reinforcement learning (see, *c.f.*, [23]).

문제 배경 - 왜 Bandit 평가가 어려운가?

- 일반적인 지도학습(supervised learning)에서는 평가가 쉬움
    - 학습 데이터, 테스트 데이터를 나눠서

    - 모델의 성능을 수치로 바로 측정할 수 있음

→ 그러나 Contextual Bandit 문제에서는 평가가 매우 어려움

이유 : 상호작용적(interactive)환경

- Bandit 알고리즘은 매 순간 arm을 선택하고 → 그 보상(payoff)을 관찰함
- 알고리즘이 어떤 arm을 고르느냐에 따라 관찰할 수 있는 데이터가 달라짐
- “내가 선택하지 않은 arm에 대한 보상은 볼 수 없다”

→ 따라서:

- 알고리즘의 성능을 평가하려면
- 실제 시스템에 알고리즘을 적용해서 테스트 해야 할 것 처럼 보임

현실적인 문제 - Live Test가 힘든 이유

- 실제 서비스에서 새로운 알고리즘을 곧바로 적용하는 것은 어려움(트래픽 리스크, 운영 비용, 사용자 경험 손상 가능성)

→ 실험적 적용(live test)은 현실적으로 매우 부담스러움

오프라인 평가의 한계

보통 우리가 가진건 = 과거에 수집한 로그 데이터(logged data) = 다른 알고리즘이 arm을 고르고 그 arm만 관찰해 놓은 데이터

문제는:

- 로그 데이터는 다른 알고리즘(logging policy)이 선택한 arm들만 포함하고 있음
- 내가 평가하려는 알고리즘 π가 고를 arm들과는 다를 가능성이 큼
- π의 성능을 로그 데이터로 평가하기가 매우 어렵다

→ 이 문제는 RL(Reinforcement Learning)에서 말하는 Off-Policy Evaluation문제와 동일함

정리하자면 bandit알고리즘 평가는 과거데이터만으로 하기 어렵다. 과거 정책이 선택하지 않은 arm의 보상은 알 수 없기 때문

One solution is to build a simulator to model the bandit process from the logged data, and then evaluate π with the simulator. However, the modeling step will introduce *bias* in the simulator and so make it hard to justify the reliability of this simulator-based evaluation approach. In contrast, we propose an approach that is simple to implement, grounded on logged data, and *unbiased*.

Bandit 알고리즘을 평가하려면 live test 말고 다른 방법이 필요하다

흔한 해결책 – 시뮬레이터

한가지 해결책은 : 시뮬레이터(simulator)를 만드는 것

시뮬레이터가 무엇인가?

> 시뮬레이터는 현실에서 일어날 일을 가상으로 흉내내는 시스템

- bandit문제에서는 예를 들어:
  - 사용자가 뉴스를 클릭할지 말지를 예측해주는 가짜 시스템을 만드는 것.

- 과거 데이터를 이용해:
  - “이 사용자라면 이 기사를 클릭할 확률이 30%다.”
  - 이런 식으로 클릭 여부를 가상으로 생성해내는 게 시뮬레이터이다.

왜 쓰려고 할까

- π라는 알고리즘을 실제로 돌려보지 않고
- **미리 성능을 가상으로 알아보고 싶기 때문**

문제점

하지만 문제는 시뮬레이터를 만드는 과정(모델링 단계)에서 편향(bias)이 생길 수 있음

그래서 시뮬레이터를 이용한 평가 방법이 신뢰할 만하다고 말하기 어려움

→ 왜냐하면:

- 시뮬레이터는 현실이 아니고
- 결국 사람이 만든 가정(assumptions)으로 돌아가기 때문
  - “클릭 확률이 30%”라고 했지만
    - 진짜로는 10%일 수도 있고, 50%일 수도 있음
- 이렇게 잘못된 가정이 쌓이면
  - 시뮬레이터 평가가 **현실과 다를 위험(판향)**이 생긴다.

논문의 제안

> **구현하기도 간단하고**
>
> **과거 로그 데이터 자체를 그대로 쓰고**
>
> **편향되지 않은(unbiased)** 방법을 제안

In this section, we describe a provably reliable technique for carrying out such an evaluation, assuming that the individual events are i.i.d., and that the logging policy that was used to gather the logged data chose each arm at each time step uniformly at random. Although we omit the details, this latter assumption can be weakened considerably so that any randomized logging policy is allowed and our solution can be modified accordingly using rejection sampling, but at the cost of decreased efficiency in using data.

논문의 해결책 – 로그 데이터를 이용한 무편향 평가법

논문이 제안하는 평가법은

- 과거 로그 데이터만 가지고
- **편향 없이** π의 성능을 평가할 수 있는 방법

두가지 조건

- 각 이벤트들이 독립적이고 동일한 분포(i.i.d)를 따라야 한다
  - 즉, 과거 이벤트들끼리 서로 영향을 주지 않고 독립적이어야 함
- 과거 로그 데이터를 수집할 때 쓰인 logging policy가
  - 모든 arm을 매 시점마다 균등한 확률로(random uniform) 선택했다는 것
  - → 즉, 과거에 arm들을 무작위로 골랐다는 가정이 필요하다.

logging policy의 균등 선택 가정의 약화

위의 가정이 너무 빡센 것 같다면 논문에서는 “꼭 균등할 필요는 없고 arm마다 선택 확률이 달라도 된다”고 하고 있다.

→ 대신 **rejection sampling이라는 방법을 써야 한다.**

rejection sampling이 무엇인가?

쉽게 말해:

- 과거 로그 데이터 중에서
- 내가 원하는 분포에 맞는 데이터만 골라 쓰는 방법
- 하지만 문제는:
  - retain되는 데이터 수가 확 줄어든다.
  - 데이터 효율이 떨어짐 → **더 많은 로그 데이터가 필요**

More precisely, we suppose that there is some unknown distribution D from which tuples are drawn i.i.d. of the form $(\mathbf{x}_1, ..., \mathbf{x}_K, r_1, \ldots, r_K)$, each consisting of observed feature vectors and *hidden* payoffs for all arms. We also posit access to a large sequence of logged events resulting from the interaction of the logging policy with the world. Each such event consists of the context vectors $\mathbf{x}_1, ..., \mathbf{x}_K$, a selected arm a and the resulting observed payoff r_a. Crucially, only the payoff r_a is observed for the single arm a that was chosen uniformly at random. For simplicity of presentation, we take this sequence of logged events to be an infinitely long stream; however, we also give explicit bounds on the actual finite number of events required by our evaluation method.

Our goal is to use this data to evaluate a bandit algorithm π. Formally, π is a (possibly randomized) mapping for selecting the arm a_t at time t based on the history h_{t-1} of $t-1$ preceding events, together with the current context vectors $\mathbf{x}_{t1}, ..., \mathbf{x}_{tK}$.

평가 방법 개요

가정

- 어떤 분포 D가 존재:

이 분포 D로 부터 아래와 같은 형태의 튜플(tuple)들이 독립적으로 동일한 분포(i.i.d)로 생성된다고 가정함.

$$(x_1, ..., x_K, r_1, ..., r_k)$$

- $x_i = arm\ i$의 feature vector
- $r_i = arm\ i$의 실제 보상

→ 즉, 각 arm이 가진 특징들과 그 arm이 실제로 주는 보상 값들이 어떤 분포 D를 따르고 있다고 봄.

- 로그 데이터는 과거 logging policy가 수집
  - 매번 랜덤하게 arm a 선택
  - 선택한 arm a의 feature vector x_a와 보상 r_a만 관찰됨
  - 이때 남은 기록은:

$$(x_1, ..., x_K, a, r_a)$$

  - arm들의 context
  - 선택된 arm a
  - 그 arm의 보상 r_a

→ 다른 arm들의 보상은 모름

평가 목표

- 새로운 알고리즘 π를 평가하고 싶음
- π는:
  - 과거 이벤트 h_{t-1}를 바탕으로
  - 현재 context $(x_{t,1}, ..., x_{t,K})$중 하나를 arm으로 선택

→ 우리가 하고 싶은 건:

- 과거 로그 데이터만으로
- π가 실제로 얼마나 잘할지 평가하는 것!

Our proposed policy evaluator is shown in Algorithm 3. The method takes as input a policy π and a desired number of "good" events T on which to base the evaluation. We then step through the stream of logged events one by one. If, given the current history h_{t-1}, it happens that the policy π chooses the same arm a as the one that was selected by the logging policy, then the event is retained, that is, added to the history, and the total payoff R_t updated. Otherwise, if the policy π selects a different arm from the one that was taken by the logging policy, then the event is entirely ignored, and the algorithm proceeds to the next event without any other change in its state.

평가 방법 – 핵심 아이디어

> "과거 로그에서 π와 같은 arm을 고른 이벤트만 뽑아 평가하면,
>
> π를 실제로 서비스에서 돌린 것과 **똑같은 결과**가 나온다."

과거 로그

이벤트	logging policy가 고른 arm	관찰된 보상
①	arm2	클릭 O
②	arm1	클릭 X
③	arm3	클릭 O

π의 선택

- π도 arm2를 선택한다면:
  - **이벤트 ①** 사용 가능 → retain
- π가 arm1을 선택했다면:
  - 이벤트 ①은 못 씀 → skip
- π가 arm3을 선택했다면:

- **이벤트 ③** 사용 가능 → retain

> π와 과거 policy가 같은 arm을 고른 이벤트만 모아 평가한다.
>
> 다르면 그냥 버린다.

제안하는 정책 평가 방법(policy evaluator)은 algorithm 3에 나와있음

Algorithm 3 Policy_Evaluator.

Inputs: $T > 0$; policy π; stream of events
$h_0 \leftarrow \emptyset$ {An initially empty history}
$R_0 \leftarrow 0$ {An initially zero total payoff}
for $t = 1, 2, 3, \ldots, T$ **do**
  repeat
    Get next event $(\mathbf{x}_1, ..., \mathbf{x}_K, a, r_a)$
  until $\pi(h_{t-1}, (\mathbf{x}_1, ..., \mathbf{x}_K)) = a$
  $h_t \leftarrow \text{CONCATENATE}(h_{t-1}, (\mathbf{x}_1, ..., \mathbf{x}_K, a, r_a))$
  $R_t \leftarrow R_{t-1} + r_a$
end for
Output: R_T/T

이 방법은 아래 두가지를 입력으로 받음

- 평가하려는 정책(알고리즘) π
- 평가에 사용할 좋은 이벤트(good events)의 개수 T

  → 여기서 good events란 π가 선택한 arm과 과거 logging policy가 선택한 arm이 우연히 일치한 이벤트를 말함.

- 로그 데이터를 한 줄씩 읽어옴
- 평가하려는 알고리즘 π에게 "이 context에서 어떤 arm을 고르겠냐"고 물어봄

- 만약 π가 logging policy가 실제로 선택했던 arm a와 **같은 arm을 선택**했다면:
  - 이 이벤트를 **retain (채택)** 한다
  - payoff r_a를 π의 평가 성과로 누적
- π가 다른 arm을 선택했다면:
  - 이 이벤트는 **버린다(무시)**

왜 이게 무편향일까?

- logging policy가 각 arm을 **동일한 확률(1/K)**로 선택했으므로:
  - π가 어떤 arm을 선택했든
  - π와 logging policy가 **같은 arm을 선택할 확률은 1/K**
- 즉:
  - retain된 이벤트들은
    - 분포 D에서 π가 직접 선택한 arm으로부터 온 것과 동일한 분포를 가진다.

→ 이 방법으로 평가한 성능은 실제 π를 live로 운영했을 때 얻을 성능과 **같은 기대값**을 가진다.

왜 두 방법이 같다고 할 수 있을까?

1. 실제로 π를 서비스에 띄워서 **T번 arm을 선택해서 평가**하는 것
2. 과거 로그 데이터에서
   - **우연히 π와 같은 arm**을 고른 이벤트만 골라 평가하는 것

→ 이 둘은 **통계적으로 같은 결과를 줌.**

예시)

- 가정: arm이 4개라서 k = 4
- π가 arm2를 고른다고 해보자.
- 과거 로그는 무작위로 arm을 골랐다
- arm 2를 고를 확률 → 1/4 = 25%

- 그러면 과거 로그 중에서 logging policy가 우연히 arm2를 고른 이벤트만 retain하면 그 데이터는 실제로 π가 arm2를 골라서 얻은 데이터와 **같은 의미**가 된다.

THEOREM 1. *For all distributions D of contexts, all policies π, all T, and all sequences of events h_T,*

$$\Pr_{Policy\_Evaluator(\pi,S)}(h_T) = \Pr_{\pi,D}(h_T)$$

where S is a stream of events drawn i.i.d. from a uniform random logging policy and D. Furthermore, the expected number of events obtained from the stream to gather a history h_T of length T is KT.

This theorem says that *every* history h_T has the identical probability in the real world as in the policy evaluator. Many statistics of these histories, such as the average payoff R_T/T returned by Algorithm 3, are therefore unbiased estimates of the value of the algorithm π. Further, the theorem states that KT logged events are required, in expectation, to retain a sample of size T.

평가 효율성

- retain 확률은 1/k
- arm 개수 K가 클수록 retain 확률은 낮아져서
  - 필요한 로그 데이터의 양이 많아질 수 있음
- 하지만 무편향성이 보장되기 때문에 매우 강력한 평가법

> $Theorem 1$. **과거 로그에서 π와 arm이 일치한 이벤트만 모으면, 실제적으로 π를 운영했을 때랑 동일한 평가 결과를 얻을 수 있고 평균적으로 KXT 개의 로그가 필요하다.**

예: arm이 4개라면 (K=4)

- π를 평가하기 위해 T = 100개의 “good” 이벤트를 얻고 싶다.
- 그럼 평균적으로 얼마나 로그가 필요할까?
  - K×T=4×100=400K × T = 4 × 100 = 400K×T=4×100=400
- 즉, 로그를 400개 살펴봐야
  - π와 arm이 일치한 이벤트가 **100개 정도** retain된다.

PROOF. The proof is by induction on $t = 1, \ldots, T$ starting with a base case of the empty history which has probability 1 when $t = 0$

under both methods of evaluation. In the inductive case, assume that we have for all $t - 1$:

$$\Pr_{\text{Policy\_Evaluator}(\pi,S)}(h_{t-1}) = \Pr_{\pi,D}(h_{t-1})$$

and want to prove the same statement for any history h_t. Since the data is i.i.d. and any randomization in the policy is independent of randomization in the world, we need only prove that conditioned on the history h_{t-1} the distribution over the t-th event is the same for each process. In other words, we must show:

$$\Pr_{\text{Policy\_Evaluator}(\pi,S)}((\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K}, a, r_{t,a}) \mid h_{t-1})$$
$$= \Pr_{D}(\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K}, r_{t,a}) \Pr_{\pi(h_{t-1})}(a \mid \mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K}).$$

“Policy Evaluator로 얻은 데이터가, 실제로 π가 운영될 때 얻은 데이터와 **확률적으로 똑같다**는 것을 보이겠다.”

증명

Step1- base case

- t=0일 때
- history가 비어 있는(empty) 경우이다
- 이때는 두 평가 방법 모두에서 빈 history의 확률이 1이다.

Step 2 - 귀납가정

- t-1번째까지 history는 둘이 같다고 같다고 가정
- 즉, t-1까지는 아래가 성립한다고 가정한다:
  - Policy Evaluator로 얻은 history h_{t-1}의 확률이
  - 실제로 π가 분포 D에서 만든 history h_{t-1}의 확률과 같다

$$\Pr_{\text{Policy\_Evaluator}(\pi,S)}(h_{t-1}) = \Pr_{\pi,D}(h_{t-1})$$

Step3 - t번째 이벤트를 보자.

- 그리고 우리는 t번째 이벤트가 추가된
  - 어떤 history h_t에 대해서도
- 동일한 식이 성립함을 증명하려고 한다.
- 데이터가 **i.i.d. (독립적이고 동일 분포)**이고

- 정책 π의 무작위성과 세상의 무작위성은 서로 독립이기 때문에

- 따라서 우리는 아래만 증명하면 된다:
  - h_{t-1}가 주어졌을 때
  - 두 과정에서 t번째 이벤트의 분포가 **동일하다**는 것

- 현실 세계에서는:
  1. 분포 D가 context와 보상을 만든다.
  2. π가 그 context를 보고 arm을 고른다.

- Policy Evaluator에서는:
  1. 로그에서 무작위로 뽑은 context와 arm, 보상이 온다.
  2. π가 context를 보고 **같은 arm을 고르면** retain

→ 결국 retain된 이벤트의 확률은:

$$\Pr_{\text{Policy\_Evaluator}(\pi,S)}((\mathbf{x}_{t,1},...,\mathbf{x}_{t,K},a,r_{t,a}) \mid h_{t-1})$$
$$= \Pr_{D}(\mathbf{x}_{t,1},...,\mathbf{x}_{t,K},r_{t,a}) \Pr_{\pi(h_{t-1})}(a \mid \mathbf{x}_{t,1},...,\mathbf{x}_{t,K}).$$

- context와 보상이 D에서 만들어질 확률 × π가 그 arm을 고를 확률

왜 두 과정이 같을까?

- Policy Evaluator가 retain하는 이벤트의 확률은 항상:1/K
- π가 고른 arm과 logging policy가 우연히 같을 확률도 1/K

→ 그래서 retain된 이벤트 분포는:

- **π가 직접 arm을 고른 것과 완전히 똑같다.**

왜 retain 확률이 1/K인가?

- 과거 로그가 arm3를 선택할 확률 = 1/K
- π도 arm3 고르는 순간만 retain하기 때문

Since the arm a is chosen uniformly at random in the logging policy, the probability that the policy evaluator exits the inner loop is identical for any policy, any history, any features, and any arm, implying this happens for the last event with the probability of the last event, $\Pr_D(\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K}, r_{t,a})$. Similarly, since the policy π's distribution over arms is independent conditioned on the history h_{t-1} and features $(\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K})$, the probability of arm a is just $\Pr_{\pi(h_{t-1})}(a|\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,K})$.

Finally, since each event from the stream is retained with probability exactly $1/K$, the expected number required to retain T events is exactly KT. □

logging policy는 arm을 무작위로 고름

- 과거 로그를 만든 logging policy는:
  - arm1, arm2, arm3 ... arm K를
  - 모두 똑같은 확률(1/K)로 선택한다.

→ 그래서 Policy Evaluator가:

- 과거 로그를 하나 읽어오면
- π가 고른 arm과 **같은 arm**이 선택될 확률은 항상:

$$1/K$$

π도 arm을 선택

- π는 context를 보고 arm을 고른다.
- π가 arm a를 고를 확률은:

$$\Pr_{\pi}(h_{t-1})(a \mid x_{t,1}, \ldots, x_{t,K})$$

즉:

- π가 과거 로그의 arm 선택과 **같은 arm을 고를지 여부**는 π 자체의 정책에 달려 있다.

retain되려면 두 조건이 맞아야 한다

1. 과거 로그가 arm a를 선택해야 하고
2. π도 arm a를 선택해야 한다.

→ 그런데 로그가 arm a를 선택할 확률은 항상 1/K

→ π가 arm a를 고를 확률은 π가 context를 보고 결정

retain 확률은

- 평균적으로 이벤트 retain 확률은 1/K
- π가 잘 고르는 arm에 따라 조금 달라지지만,
- **전체 평균적으로는 retain 확률이 1/K**

필요한 로그 개수

- “good 이벤트” T개가 필요하다면?
- 평균적으로:

$$K \times T$$

개의 로그 이벤트를 읽어야 한다.