강화학습 기반 디지털 광고 노출 정책 최적화

A71042 이상강 https://github.com/JinsaGalbi/RL-Project/blob/main/RL_project_final.ipynb

프로젝트 주제 및 목표

프로젝트 주제

디지털 광고 시스템에서 **어떤 사용자에게 광고를 노출할지(treatment)**를 강화학습으로 결정하여, 전환(conversion)·방문(visit)은 극대화하고 광고비용은 최소화하는 **최적의 정책(policy)**을 학습.

프로젝트 목표

1. Contextual Bandit 환경을 구성하고 reward 구조를 정의.
2. Policy Gradient, DQN, A2C 알고리즘을 적용하여 최적 정책을 탐색.
3. Hyperparameter 변화가 정책 및 성능에 미치는 영향을 분석.
4. Seed 변화에 따른 평균 성능 및 신뢰구간을 산출하여 안정성을 평가.

데이터셋 소개

데이터 특징

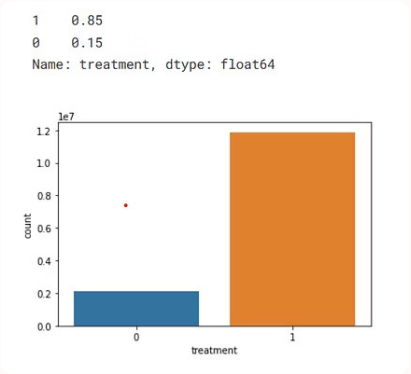
- Criteo AI Lab uplift modeling dataset
- kaggle에서 발췌한 랜덤화된 광고 실험(RCT) 데이터

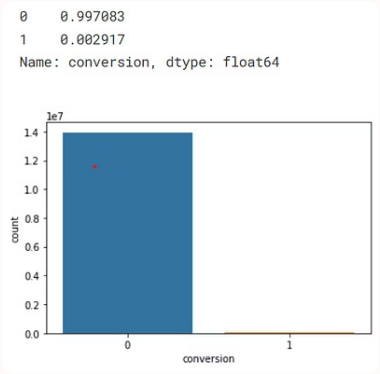
주요 컬럼

- **Feature (12개):** f0 ~ f11
- **treatment:** 광고 노출 여부 (0/1)
- **visit:** 광고 후 방문 여부 (0/1)
- **conversion:** 광고 후 전환 여부 (0/1)

도메인 특성

- treatment 비율 약 85%
- conversion 비율 약 0.2%
- conversion이 발생하면 reward 규모가 매우 큼 → reward(long-tail) 분포로 인해 정책 학습이 민감





데이터 전처리 과정

01

Train / Validation 분리

Train : Validation 8:2로 분리

Feature에 대한 Standard Scaling 수행

02

Reward 초기 구조

$$R = r_{\text{conv}} \cdot I_{\text{conv}} + r_{\text{visit}} \cdot I_{\text{visit}} \cdot (1 - I_{\text{conv}}) - c_{\text{ad}} \cdot I_{\text{ad}}$$

Where:

- R: reward
- r_conv: 전환보상
- I_conv: 전환여부 (0/1)
- r_visit: 방문보상
- I_visit: 방문여부 (0/1)
- c_ad: 광고비용
- I_ad: 광고여부 (0/1)

※ 전환되었을 경우에는 전환보상만, 방문만 했을 경우에는 방문 보상만 부여

03

문제 발생

- reward의 평균이 **+0.07으로 양수**
- treat가 이득인 고객은 **4.7%뿐**인데
- 전체 reward는 treat할 때 평균적으로 플러스 → Policy Gradient와 A2C가 Always treat 정책으로 수렴

Reward 재설계

문제 핵심

- treat가 항상 평균적으로 이득 → feature 기반 의사결정이 이루어지지 않음
- **RL 알고리즘이 "모두 treat(모든 고객에게 광고)" 정책을 최적이라고 판단**

해결 전략: **Reward Re-centering**

$$R = r_{conv} \cdot I_{conv} + r_{visit} \cdot I_{visit} \cdot (1 - I_{conv}) - c_{ad} \cdot I_{ad}$$

$$R_{new} = R_i - \text{mean}(R)$$

효과

- **treat**의 전체 평균 **reward = 0**
- **treat** 시 이득인 고객만 **reward > 0**
- 기존의 **Always treat** 정책 붕괴
- **feature** 기반 선택적 **treat** 정책이 학습 가능해짐

강화학습 환경 설계

환경 형태

- **Contextual Bandit**
- 각 episode = 1 step
- transition 없음, γ(감가율) 의미 없음

구성 요소

- **state:** 사용자의 feature vector (f0~f11)
- **action:** 0(광고 X), 1(광고 O)
- **reward:**
  - action=0 → 0
  - action=1 → reward_if_treat[state]

환경의 특성

deterministic reward를 사용하지만 평균이 0으로 교정되어 정책이 state-dependent하게 변함

exploration이 중요한 환경

알고리즘 설명: **Policy Gradient, DQN, A2C**

Policy Gradient (REINFORCE)

확률적 정책 π(a|s; θ)를 직접 학습

Gradient 업데이트: θ ← θ + α * log π(a|s) * reward

Advantage normalization으로 안정성 강화

Entropy regularization으로 exploration 유지

장점: direct optimization → 빠르게 좋은 정책을 형성, deterministic bandit에서 매우 안정적

단점: variance가 큼 → seed 변경 시 변동 가능

DQN (Deep Q-Network)

Q(s,a)를 신경망으로 근사

Replay buffer + target network 기반

epsilon-greedy exploration

bandit 문제에서는 bootstrap이 필요 없어서 불리

특징: action-value function을 직접 학습, exploration 정책이 성능에 큰 영향, reward imbalance에 매우 민감

📌 이 프로젝트에서 **PG/A2C 대비 낮은 성능**을 보임

A2C (Advantage Actor-Critic)

Actor: π(a|s)

Critic: V(s)

Advantage = reward - V(s)

Policy Gradient + Value Learning 결합 알고리즘

장점: PG보다 variance가 매우 낮음, reward re-centering 환경에서 특히 안정적, exploration을 entropy로 제어 가능

Hyperparameter 설정

Policy Gradient

- hidden_dim = 64 / 128
- lr = 1e-3 / 3e-4
- entropy_coef = 0.01 / 0.001
- seed = 0, 1, 2

DQN

- lr = 1e-3
- epsilon decay 스케줄
- batch size, replay size

A2C

- hidden_dim = 64 / 128
- entropy_coef = 0.01 / 0.001
- value_coef = 0.5
- lr = 1e-3 / 3e-4

📌 각 알고리즘 내 하이퍼파라미터를 조합하여 4번씩 학습. 이후 최적 하이퍼파라미터로 3번의 시드로 학습.

실험 셋업

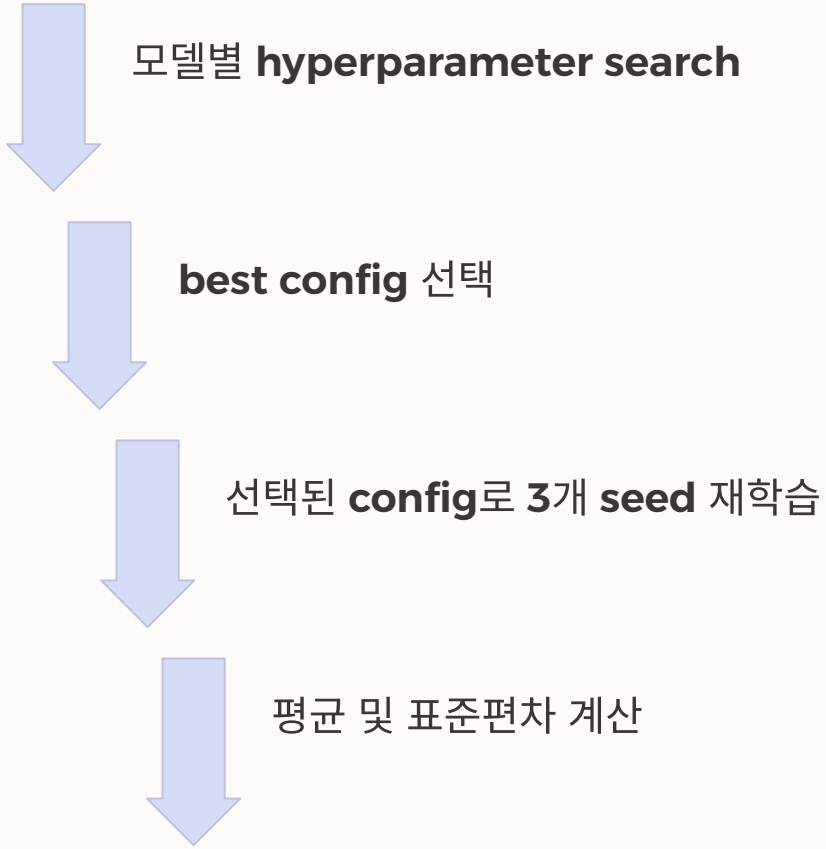
실험 환경

- Python 3.10
- Jupyter Notebook
- PyTorch
- CPU 기반 학습 (bandit 특성상 빠름)

평가 지표 **(Evaluation Metric)**

1. 평균 reward
2. action=1 비율
3. feature bin 별 treat 정책 분석
4. seed별 mean & std

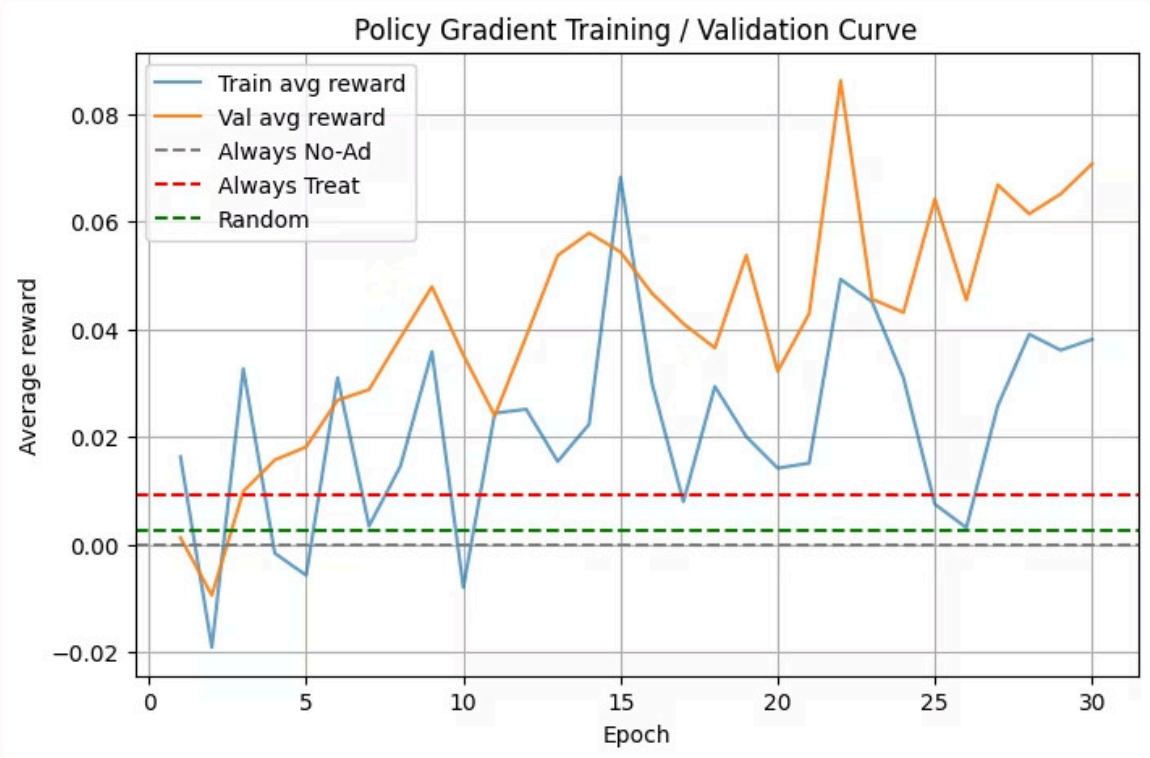
프로토콜



Policy Gradient 결과

Hyperparameter 결과

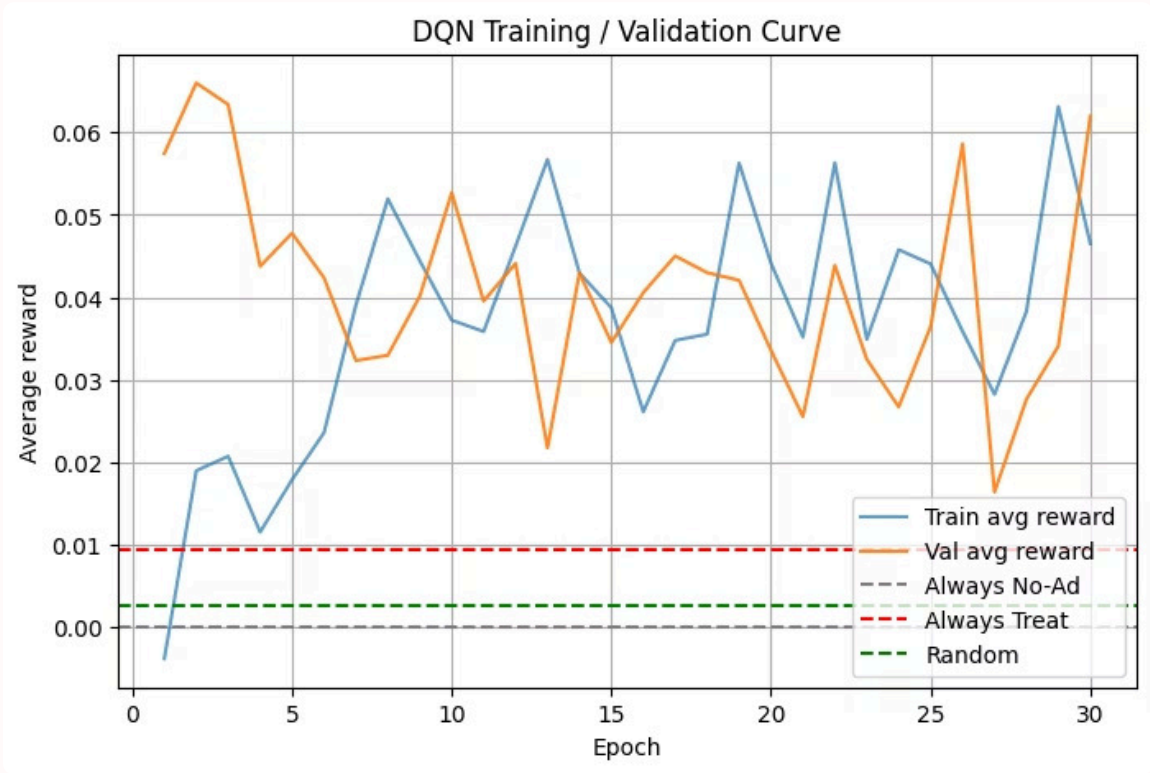
- best config: hidden_dim=64, lr=1e-3, entropy=0.01
- validation mean reward 약 0.0568
- seed 3개 평균 reward 안정적
- 초반에는 0 근처에서 진동하지만, epoch이 증가할수록 완만하게 상승
- 후반부에는 0.04~0.06 구간에서 안정적으로 수렴



DQN 결과

Hyperparameter 및 학습 결과

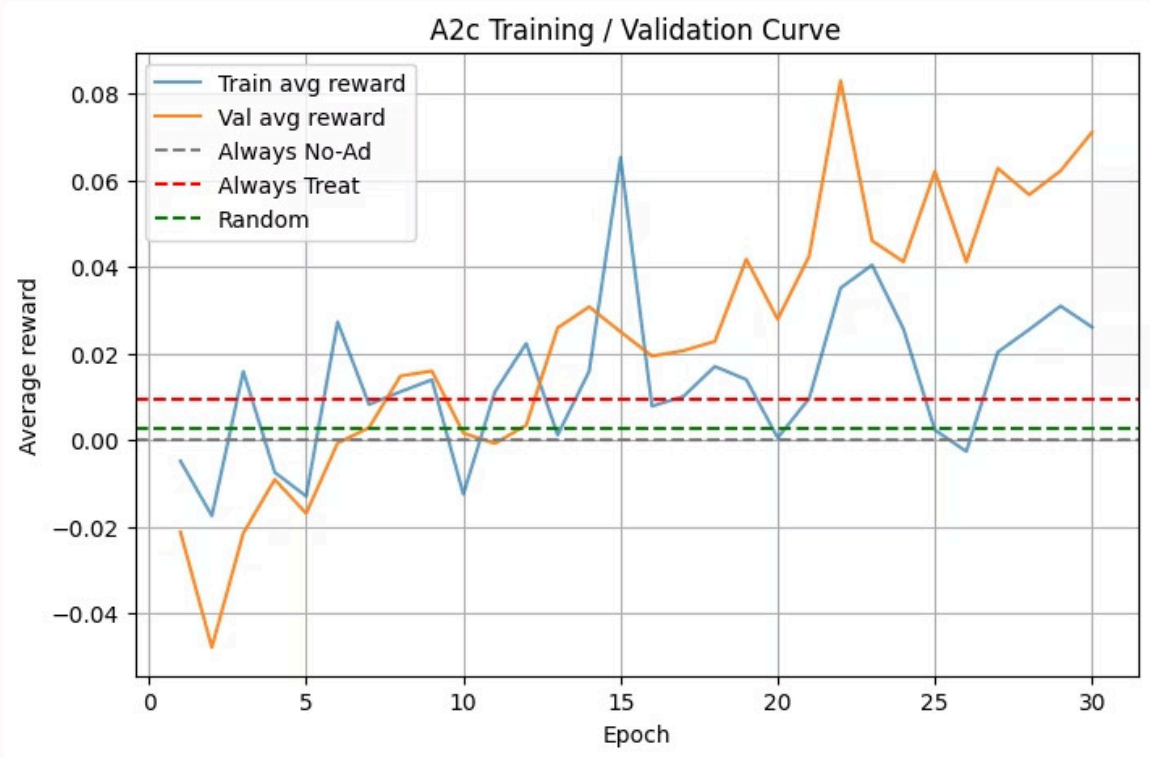
- best config: hidden_dim=64, lr=1e-3, epsilon_start=1.0, epsilon_end=0.05, epsilon_decay_steps=50000, batch_size=256
- seed 3개 평균으로 평가
- 초반부터 0.04~0.06 수준으로 꽤 높게 나오지만
- 전반적으로 0.03~0.06 사이에서 **상승보다는 진동**하는 형태



A2C 결과

Hyperparameter 및 학습 결과

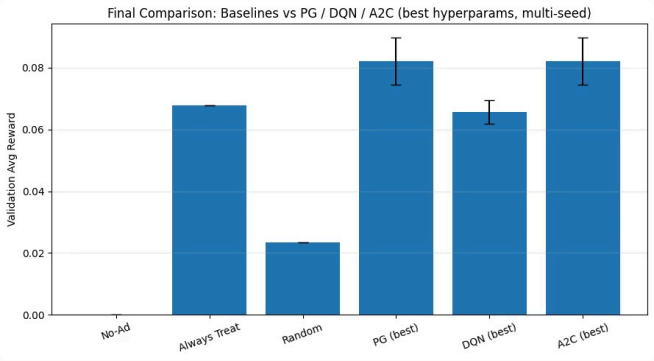
- best config: hidden_dim=64, lr=1e-3, entropy=0.01
- mean reward ≈ 0.082 (PG와 유사한 성능)
- variance가 PG보다 낮음 → 더 일관적
- 초기에 다소 불안정하지만, 10 epoch 이후 꾸준히 상승
- 0.05~0.07 구간에서 안정적으로 유지되며 PG보다 약간 우위



모델 간 최종 비교

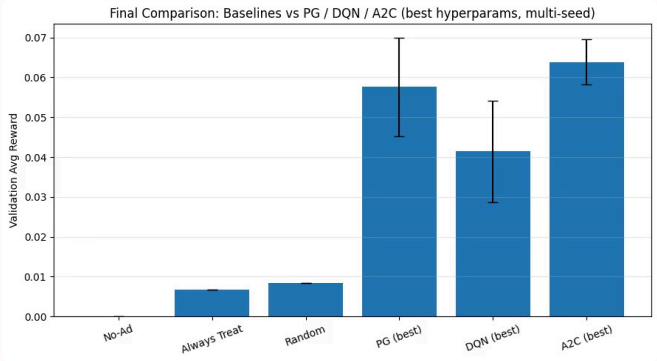
Reward 재설계 이전 결과

Policy Gradient와 A2C가 모두 action=1로 같은 정책을 채용함



Reward 재설계 이후 결과

모든 acion=1로 하지 않게 되면서 알고리즘의 결과가 서로 상이해짐



모델 간 최종 비교 표 **(Reward** 재설계 후**)**

세 알고리즘 모두 reward 재설계 이후 **“아무나에게 다 보내는 정책”보다 훨씬 좋은 정책을 학습하며 A2C가 가장 우수한 성능을 보임**

모델	Mean Reward	Std	정책 특성 요약
Policy Gradient (PG)	0.057612	0.012309	reward re-centering 후 feature 기반 선택적 treat 정책 학습
DQN	0.041447	0.012722	bandit 환경에서 exploration 의존도가 높아 성능 변동이 큼
A2C	0.063815	0.005666 (가장 안정적)	PG 대비 variance 낮고 안정적으로 높은 reward 기록

정책 해석 **(Explainability)**

방법

- 각 feature(f0~f11)를 5분위로 구간화하여 treat 확률 계산
- 모델 간 treat 행동 비교

관찰 결과

- PG/A2C는 특정 feature 범위에서 treat 비율이 높거나 낮게 나타남 (f0,f2,f6,f8,f9)
- reward 재설계 덕분에 **state-dependent policy가 성공적으로 학습됨**
- DQN은 treat 비율 변화 폭이 상대적으로 적음

구간별 정책 확인 그래프

f2_bin	pg_action	a2c_action	dqn_action
(-0.771, -0.165]	0.306667	0.311167	0.140
(-0.165, 1.228]	0.290500	0.298000	0.130
(1.228, 2.031]	0.290000	0.290000	0.134

f0_bin	pg_action	a2c_action	dqn_action
(-1.3119999999999998, -0.368]	0.2890	0.29275	0.13375
(-0.368, 0.582]	0.3085	0.31600	0.13950
(0.582, 0.962]	0.3125	0.31050	0.13600
(0.962, 1.323]	0.3015	0.30950	0.14100

f6_bin	pg_action	a2c_action	dqn_action
(-4.056, -0.844]	0.287909	0.291378	0.130823
(-0.844, 0.0411]	0.302796	0.313548	0.140215
(0.0411, 0.63]	0.306013	0.314410	0.139066
(0.63, 0.975]	0.300373	0.294776	0.135821

f8_bin	pg_action	a2c_action	dqn_action
(-4.714, -0.718]	0.297069	0.300050	0.135618
(-0.718, 0.176]	0.289276	0.289276	0.129351
(0.176, 0.671]	0.305067	0.311210	0.139908

f9_bin	pg_action	a2c_action	dqn_action
(-0.403, 0.0278]	0.300938	0.306905	0.139082
(0.0278, 6.259]	0.296255	0.292342	0.126328

토의 및 결론

결론 요약

- reward 재설계 전에는 Always treat(모두 광고) 정책 발생
- reward re-centering 이후 안정적으로 비선형 정책 학습이 가능
- Policy Gradient와 A2C가 bandit 환경에서 강력한 성능
- DQN은 탐색·reward 구조의 영향을 크게 받음

보완 및 개선사항

1

현재 reward는 완전히 결정적(deterministic)하여 일부 알고리즘(DQN)이 안정적 학습 불가

→ reward를 확률 기반(stochastic) 구조로 확장

2

현재는 1 step이라는 한계가 존재 → Sequential MDP 환경으로 전환하여 long-term 광고 전략 고려

3

향후 Policy Gradient와 A2C의 단점을 보완한 알고리즘인 PPO(Proximal Policy Optimization) 적용

감사합니다