# Artificial Intelligence Term Project

2021.6.18

# Team member

- 蔡昀達
- F08946007
-

# Motivation/Background

- In this course, we learned a lot of powerful methods such as MCTS, RL, DL.
- We can see that these action selection algorithms has increase the performance significantly as well as the complexity. Especially when it comes to bayesian methods.
- For tasks that requires a long action evaluation such as hyperparameter search and neural architectures search this does not matter. However, other tasks such as retrieval and recommendation that requires real-time response, this becomes the application bottleneck.

# Target Problem

- Main problem
  - How to achieve fast online real-time inference for decision making ?
- Consider the problem in contextual multi-arm bandit problem.
  - Given an action set A and context set C, how can we wisely select actions in order to maximize cumulative rewards (explore-exploit dilemma)
  - Algorithms such as UCB, Thompson sampling
  - Every time select an action
    - Calculate values for all (action, context) pair
    - Choose the best

$$a_t^* = \arg\max_{a \in A} P(r = 1 | x, a, \theta_t)$$

  - Inference complexity O(|A|)

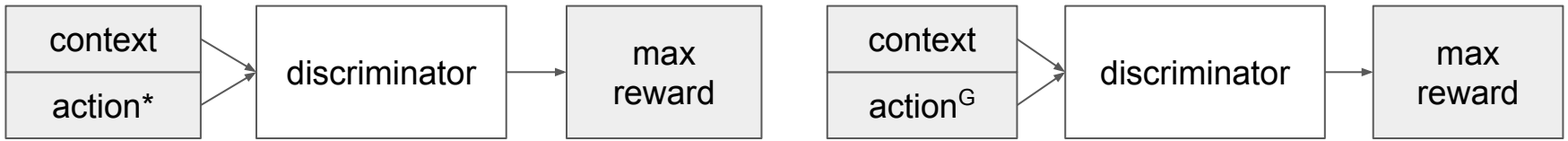
# Proposed Solution

- Main idea
  - Can we calculate the costly computation O(|A|) beforehand ?
  - Make Inference complexity O(1)
- Naive solution
  - Calculate everything and store
    - Time complexity O(|A|*|C|*N)
    - Space complexity O(|C|*N)
  - Approximation methods (MCMC, variational inference)
    - Space complexity O(|C|*N)
- Our solution
  - Generative adversarial network
    - Space complexity constant

# Proposed Solution

- Train a reward estimator


- Use the reward estimator as discriminator
- Given context, the discriminator will output the same (maximized) reward for both true optimal action* and generator output action$^{G}$.



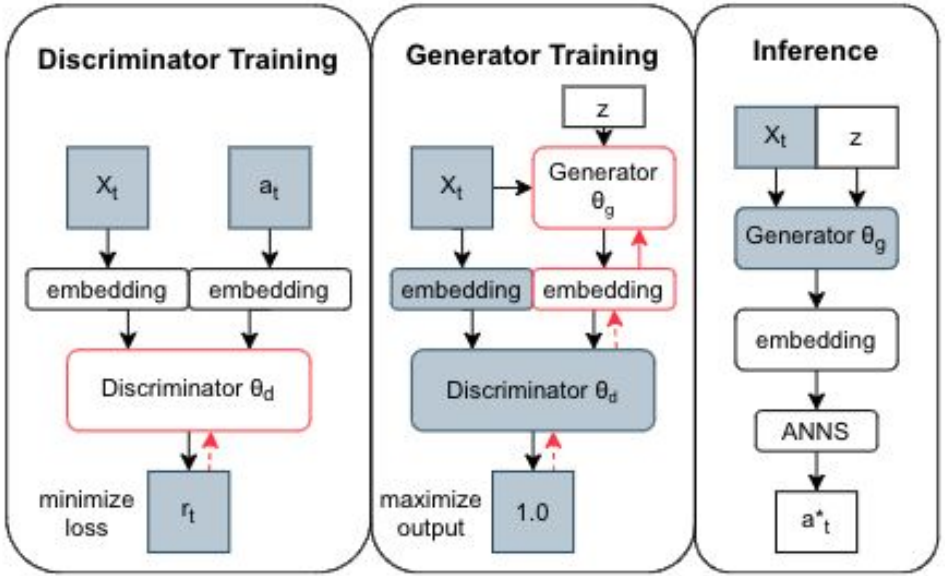
# Proposed Solution

- The generator learn the probability of the each action being argmax of the stochastic value function.
  - $a_t^* = \arg\max_{a \in A} P(r = 1 | x, a, \theta_t)$
- Constraint the discriminator to be trained with binary rewards and binary cross entropy loss
  - $\theta^* = \arg\max_{\theta} \sum_{i=1}^{T} r_i ln(\theta(x_i, a_i)) + (1 - r_i) ln(1 - \theta(x, a))$
  - The discriminator outputs estimated reward
  - The discriminator outputs the probability of being optimal action
- This results in an objective function exactly the same as GAN

$$\min_G \max_D \mathbb{E}_{a \sim p(a_t^*)} [\log(D(x_t, a, \theta_d))] + \mathbb{E}_{z \sim p(z),\, \theta_{d,t} \sim f(\theta_d)} [\log(1 - D(x_t, G(x_t, z), \theta_{d,t}))]$$

# Proposed methodology

- Architecture
  - Neural network
- Approximate bayesian inference
  - Concrete dropout
  - Neural random feature mapping
- Approximate global maxima
  - Multi-start gradient ascent
- Approximate nearest neighbor
  - HNSW
- Notation
  - x → context
  - a → action
  - r → reward



# State of the art

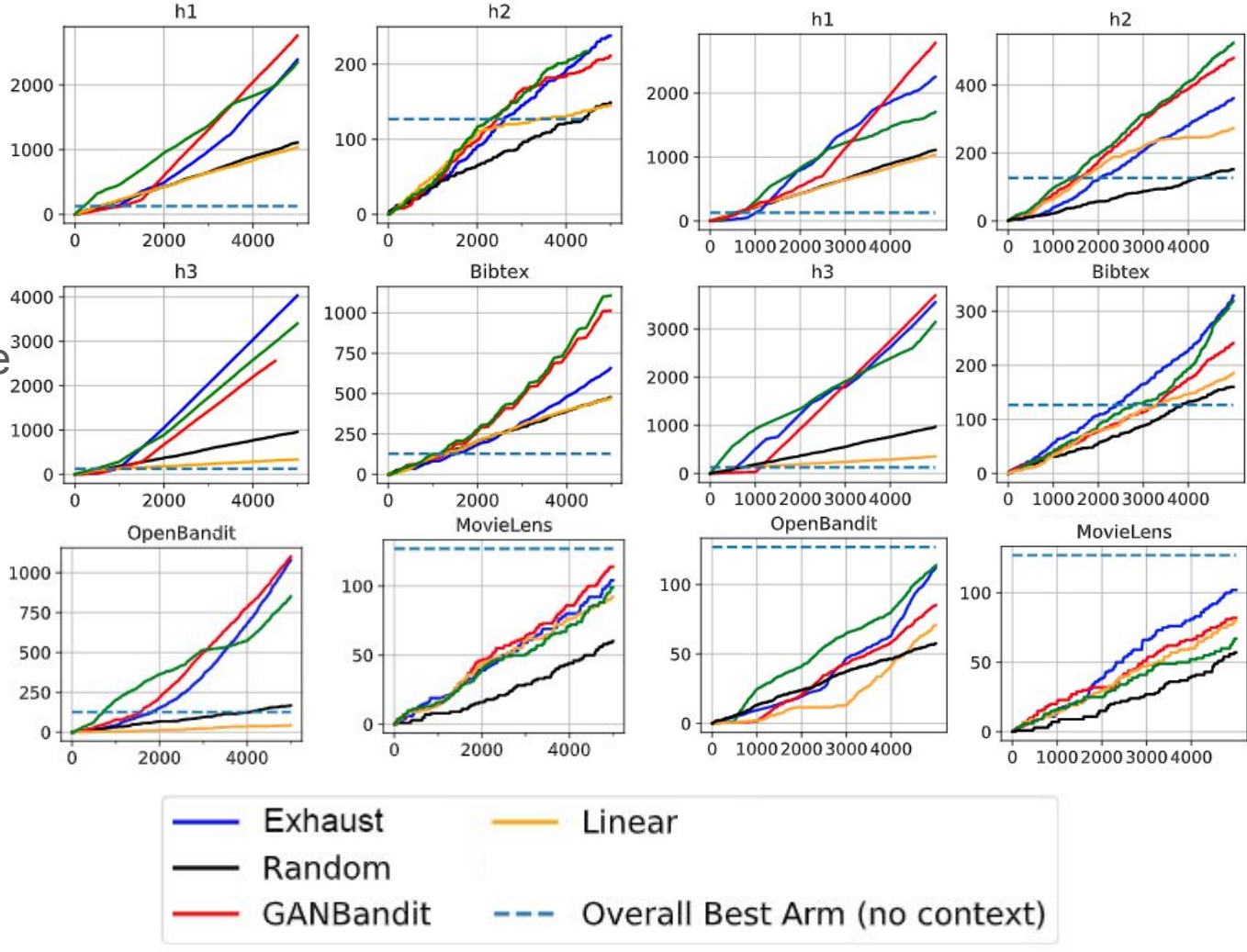
- Volumetric spanners[1]
  - uses a simple approach to select a subset of arms ahead of time. This solution is specialized for an efficient exploration only and can rule out a large number of good arms.
- QGLOC[2]
  - transforms the maximizing objective into quadratic form which then can be solved by using approximate maximum inner product search hashing.
  - QGLOC requires the objective function to be a distance or an inner product computation which can be satisfied by only a subset of models.

# Experiment

- Dataset
  - Artificial dataset
    - $h_1(x) = x\cos(x^T)a + 0.25(x^T a)$ $h_2(x) = 10(x^T a)^2$ $h_3(x) = cos(3x^T a)$
  - Real world dataset
    - OpenBandit[3]
    - Movielens[4]
    - Bibtex[5]
- Baselines
  - Random
  - Exhaust TS, UCB
  - Linear TS, UCB
  - GAN + TS, UCB
- Objective
  - Speed up inference time
  - Reduce approximation error

# Results

- Linear model fails
- Approximation error is acceptable



# Run time results

| TS | Inference | Train |
|---|---|---|
| Exhaust | 4.03 s | 9.12 s |
| Linear | 1.31e-4 s | 8.16 s |
| GAN | 1.56e-4 s | 31.49 s |

| UCB | Inference | Train |
|---|---|---|
| Exhaust | 32.09 s | 8.41 s |
| Linear | 7.36e-4 s | 9.16 s |
| GAN | 3.01e-4 s | 52.93 s |

# Limitation

- This method requires a lot of parameters fine tuning
  - Five optimization process
    - reward estimator training
    - generator training
    - approximate global maxima
    - approximate bayesian inference
    - Approximate nearest neighbor
  - Pairwise coordinate optimization

# Reference


1. Hazan, Elad, and Zohar Karnin. "Volumetric spanners: an efficient exploration basis for learning." The Journal of Machine Learning Research 17.1 (2016): 4062-4095.
2. Jun, Kwang-Sung, et al. "Scalable generalized linear bandits: Online computation and hashing." arXiv preprint arXiv:1706.00136 (2017).
3. Saito, Yuta, et al. "Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation." arXiv preprint arXiv:2008.07146 (2020).
4. Harper, F. Maxwell, and Joseph A. Konstan. "The movielens datasets: History and context." Acm transactions on interactive intelligent systems (tiis) 5.4 (2015): 1-19.
5. Bhatia, et al.The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html (2016)

# Q&A

1. By action what do you mean inside your model? Is it like the action inside a reinforcement learning model?
    a. Yes, it is like action in reinforcement learning. Although the experiments in this projects run on bandit algorithms, it can also apply to reinforcement learning, mdp, hmm, bayesian optimization kind of algorithm with slightly different assumptions.
2. Can you give an example of practical application in real life?
    a. For example, in a recommendation application, whenever a user request comes in, we have to choose an item from millions of items and recommend to the user in real-time. With such a large action (item) space, it is very hard to run in real-time, so we proposed this solution to speed up the inference process. So as mentioned on page 5, our method only benefits applications that requires real-time action selection with immediate feedback. Most algorithm that involves bayesian inference or multi-modality objective directly assume the action evaluation time is much longer than action selection time (such as hyperparameter search and neural architectures).
3. Do you have any criteria about the artificial dataset which acceptable to train the model?
    a. Since the inference complexity issue only arise in non-linear models (curse of dimensionality), we aims at solving non-linear reward functions with no structure assumption. That is also the strength that neural network has which it theoretically can fit to any distribution.
4. Can you explain more about the dataset? Speed up on what kinds of tasks?
    a. We want to speed up the task of selecting actions. No matter what learning process, reinforcement learning, mdp, bandit, bayesian optimization we require to evaluate the action given the state to choose the best action to perform. Most of the time we don't care about how much time an agent needs to select an action. But in the case when we have a large action space but requires real-time response, it becomes the bottleneck. It is hard to explain more about the dataset because this method can apply to any learning task so what kind of dataset really does not matter.
5. How is this method better than the algorithms optimized for the specific problems themselves?
    a. I don't really understand this question. On page 8, I mentioned that no matter what kind of algorithm we use to model the q-value (which can be any kind of neural networks, gaussian process, gaussian mixture models, MCTS, etc), we have to calculate the value for every (state, action) pair in order to find the best. We can apply approximation methods here to solve the argmax objective of course, as mentioned on page 5, but it still take time and space. The main novelty of our method is that we use a generator and the GAN architecture to shift the inference computation to training time. This does not contradict to what algorithm we use and can generally be applied to any learning task. More specifically, each algorithm we used in every single component in the experiment listed on page 13 is already efficient approximation methods.

If there is any further question, feel free to contact bb04902103@gmail.com

# Implementation code

https://github.com/j40903272/AI2021