Review of LinUCB algorithm and summary of Ch7, 8 of White [2012]

Eunpyo Lee

Department of Statistics, Korea University

Feb 5, 2022

Outline

Motivation I

- Netflix landing page(Personalized recommendation system)
  - Question : Which series is best for certain customer? (based on contextual information)

Motivation II

- Examples of contexual information
  - User profile : Demograhics, Usage history, $\cdots$
  - Feature of environment : Day of week, time of the day, season, $\cdots$
  - Arm profile : Genre, length, $\cdots$

A Multi-armed Bandit Formulation I

Above example can be naturally modeled as a **multi-armed bandit problem with context information.** In trial $t = 1, 2, \cdots, T$,

- A_t be a set of arms(actions) in trial t, u_t is a current user at t.
- $x_{t,a}$ summarizes information of both the current user and arm $a \in A_t$, and will be referred to as the **context**.
- a_t be the arm chosen at t, $a_t \in A_t$.
- $r_{t,a}$: Reward(payoff) of $a \in A_t$ arm at tth trial. However, We can **only observe** r_{t,a_t}.

Triplet $(x_1, a_1, r_{1,a_1}), (x_2, a_2, r_{2,a_2}), \cdots, (x_{t-1}, a_{t-1}, r_{t-1,a_{t-1}})$ and x_t is what we have.

Question : How to choose best action a_t at t, given these observations?

A Multi-armed Bandit Formulation II

T -trial regret $R_{\mathbf{A}}(T)$:

$$R_{\mathbf{A}}(T) \stackrel{\text{def}}{=} \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right].$$

- a_t^* is the arm with maximum expected payoff at trial t.
- Total T-trial payoff of A is defined as $\sum_{t=1}^{T} r_{t,a_t}$.
- Similarly, we define the optimal expected T-trial pay*off* as $\mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right]$
- Our goal is to design A so that the expected total payoff above is maximized. That is to **minimize** $R_{\mathbf{A}}(T)$. That is to maximize $\sum_{t=1}^{T} r_{t,a_t}$.

LinUCB formulation I

Following Formulation from Li et al. [2010], **LinUCB** with disjoint linear models algorithm,

- **Linearity assumption** : Expected return of an arm a is linear in its d-dimensional feature $\mathbf{x}_{t,a}$ with some unknown coefficient vector $\boldsymbol{\theta}_a^*$,

$$\mathbf{E}\left[r_{t,a} \mid \mathbf{x}_{t,a}\right] = \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_a^*.$$

- Let $\mathbf{D}_a$ be a design matrix of dimension $m \times d$ at trial t, whose rows correspond to m training inputs (e.g., m contexts that are observed previously for arm a)
- $\mathbf{c}_a \in \mathbb{R}^m$ be the corresponding response vector (e.g., the corresponding m click/no-click user feedback).

LinUCB formulation II

- Applying ridge regression to the training data $(\mathbf{D}_a, c_a)$, where D_a be a design matrix of dimension $m \times d$ at trial t, whose rows correspond to m training inputs, and c_a be the corresponding response vector with length d, then estimated θ_a :

$$\hat{\boldsymbol{\theta}}_a = \left(\mathbf{D}_a^\top \mathbf{D}_a + \lambda \mathbf{I}_d\right)^{-1} \mathbf{D}_a^\top \mathbf{c}_a, \lambda \text{ is set to 1 for now.}$$

- It can be shown Walsh et al. [2012] that, with probability at least $1 - \delta$,

$$\left|\mathbf{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_a - \mathbf{E}\left[r_{t,a} \mid \mathbf{x}_{t,a}\right]\right| \leq \alpha \sqrt{\mathbf{x}_{t,a}^\top \left(\mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d\right)^{-1} \mathbf{x}_{t,a}}$$

for any $\delta > 0$ and $\mathbf{x}_{t,a} \in \mathbb{R}^d$, where $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$ is a constant.

LinUCB formulation III

- The inequality above gives a reasonably tight UCB for the expected payoff of arm a.
- Let $\mathbf{A}_a \stackrel{\text{def}}{=} \mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d$, $b_a \stackrel{\text{def}}{=} \mathbf{D}_a^\top c_a$,
- At each trial t, choose

$$a_t \stackrel{\text{def}}{=} \arg\max_{a \in \mathcal{A}_t} \left(\mathbf{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_a + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}} \right),$$

LinUCB implementation I

- Line 4-7 : Initialization step.
- Line 8-9 : Calculate current UCB.
- Line 11 : Choose Arm a_t and Observe reward r_t.
- Line 12-13 : Update $\mathbf{A}_a = \mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d$, $b_a = \mathbf{D}_a^\top c_a$.

Algorithm 1 LinUCB with disjoint linear models.

0 : Inputs: $\alpha \in \mathbb{R}_+$
1: for $t = 1, 2, 3, \ldots, T$ do
2: Observe features of all arms $a \in \mathcal{A}_t : \mathbf{x}_{t,a} \in \mathbb{R}^d$
3: for all $a \in \mathcal{A}_t$ do
4: if a is new then
5: $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
6: $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
7: end if
8: $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
9: $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
10: end for
11: Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
12: $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
13: $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
14: end for

LinUCB implementation II

- Line 12-13 :

$$D_{new} = \begin{pmatrix} D_{old} \\ x_{new}^T \end{pmatrix}, c_{new} = \begin{pmatrix} c_{old} \\ c_{new} \end{pmatrix} \text{ by definition.}$$

$$\begin{aligned} A_{new} &= D_{new}^T D_{new} + I_d = \begin{pmatrix} D_{old}^T & x_{new} \end{pmatrix} \begin{pmatrix} D_{old} \\ x_{new}^T \end{pmatrix} + I_d \\ &= D_{old}^T D_{old} + x_{new} x_{new}^T + I_d = (D_{old}^T D_{old} + I_d) + x_{new} x_{new}^T \\ &= A_{old} + x_{new} x_{new}^T \end{aligned}$$

$$\begin{aligned} b_{new} &= D_{new}^T c_{new} = \begin{pmatrix} D_{old}^T & x_{new} \end{pmatrix} \begin{pmatrix} c_{old} \\ r_{new} \end{pmatrix} \\ &= D_{old}^T c_{old} + r_{new}\, x_{new} = b_{old} + r_{new}\, x_{new} \end{aligned}$$

LinUCB implementation III

For simplicity, Assume $A_t = \{a_1, \cdots, a_k\}$ for all $t = 1, \cdots, T$ (Arm pool does not change).

```
class LinUCB :
    def __init__(self, alpha, n_features) :
        self.alpha = alpha
        self.n_features = n_features

    def initialize(self, n_arms) :
        self.n_arms = n_arms
        self.A = np.array([np.identity(self.n_features)] * n_arms)
        self.A_inv = np.array([np.identity(self.n_features)] * n_arms)
        self.b = np.zeros((n_arms, self.n_features, 1))

    def select_arm(self,x) :
        theta = self.A_inv @ self.b
```

LinUCB implementation IV

```
        UCB = np.transpose(x) @ theta + \
              self.alpha * np.sqrt(np.transpose(x) @ self.A_inv @ x)
        return random.sample(list(np.where(UCB == np.max(UCB))[0]),1)[0]

    def update(self, chosen_arm, x, reward) :
        x = x.reshape(self.n_features,-1)

        # update A & A_inv
        self.A[chosen_arm] = self.A[chosen_arm] +x @ np.transpose(x)
        self.A_inv[chosen_arm] = np.linalg.inv(self.A[chosen_arm])

        # update b
        self.b[chosen_arm] = self.b[chosen_arm]+reward*x
        return
```

Benchmark Algorithms

- Five Algorithms are considered with varying hyper-parameters.
  - **LinUCB Algorithm**
    - Delta Range : $[0.01, 1]$
  - Epsilon-Greedy(Eps1) Algorithm
    - Epsilon Range : $[0, 1]$ are considered, where 0 indicates pure exploitation, 1 indicates pure exploration
  - Annealing Epsilon-Greedy(Eps2) Algorithm
    - Epsilon = $1/\log(t + 0.0000001)$, so that it can decay.
  - Softmax Algorithm
    - Temperature Range : $[exp(-5) \approx 0.0067, exp(5) \approx 148]$.
  - UCB1 Algorithm
    - Alpha : $[0, 5]$, here alpha is a value multiplied to the bonus.

Data set I

- Design mantrix and Reward matrix $(X_l, R_l),\ l = 1,\ 2,\ 3$ are simulated based on **IRIS**(Fisher [1936]), **music genre classification** datasets.
  - Generating Method :
    - Let (D_l, y_l) be the given design matrix and target vector, $f(\cdot)$ be the arbitrary classification algorithm.
    - Train $f(\cdot)$ with (D_l, c_l) and obtain predicted probability $\hat{\pi}(D_l) = (\hat{\pi}_1, \hat{\pi}_2, \cdots, \hat{\pi}_K)$. And let $R_l = \hat{\pi}(D_l)$.
  - Then (D_l, R_l) is now design matrix and reward matrix of K-armed bandit problem. That is, $r_d = (reward_{a_1,d}, reward_{a_2,d}, \cdots, reward_{a_K,x}) = \hat{\pi}(d)$ is considered as simulated reward for each arms $a_1, a_2, \cdots, a_K$ given context vector d.

Data set II

- (D_1, R_1) is generated from **IRIS**, and Logistic regression method.
- $(D_2, R_2), (D_3, R_3)$ is generated from **music genre classification**, and RandomForest and Logistic regression method.
- Characteristics of (D_l, R_l) :

	N, Sample size	d, # of covariates	K, # of arms	Proportions of best arm
(D_1, R_1)	150	4	3	0.33, 0.32, 0.35
(D_2, R_2)	5063	16	11	0.14, 0.09, $\cdots$, 0.09, 0.08
(D_3, R_3)	5063	16	11	$0.86, 0.12, 0.01, \cdots$

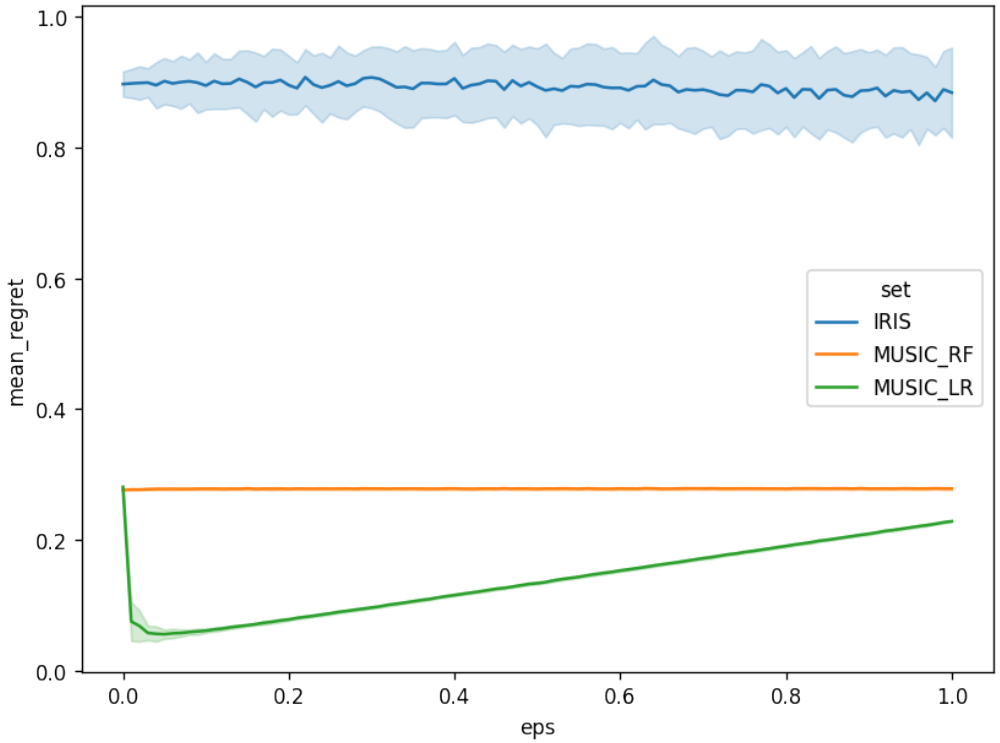
- R^2 when fitting linear regression D_l to each arms :
  - Set 1 : 0.93, 0.39, 0.79
  - Set 2 : All < 0.01
  - Set 3 : 0.56, 0.24, 0.85, 0.58, 0.58, 0.27, 0.43, 0.56, 0.96, 0.03, 0.75
- (D_4, R_4) : From (D_3, R_3), Leave only $k = 3$ arms with highest R^2.

Setting

- Experiment 1 : Repeat 100 simulations of 5 algorithms with their hyper-parameters varying on selected grid on each **IRIS, MUSIC RF, MUSIC LR** data sets.
- Experiment 2 : Repeat 100 simulations of 5 algorithms with their hyper-parameters fixed according to Experiment 1.
  - EPS1 : Epsilon is set to 0.05
  - Softmax : Temperature is set to 0.05
  - LinUCB : Delta is set to 0.1
  - However, EPS2, UCB1 are simulated following basic settings from the book.

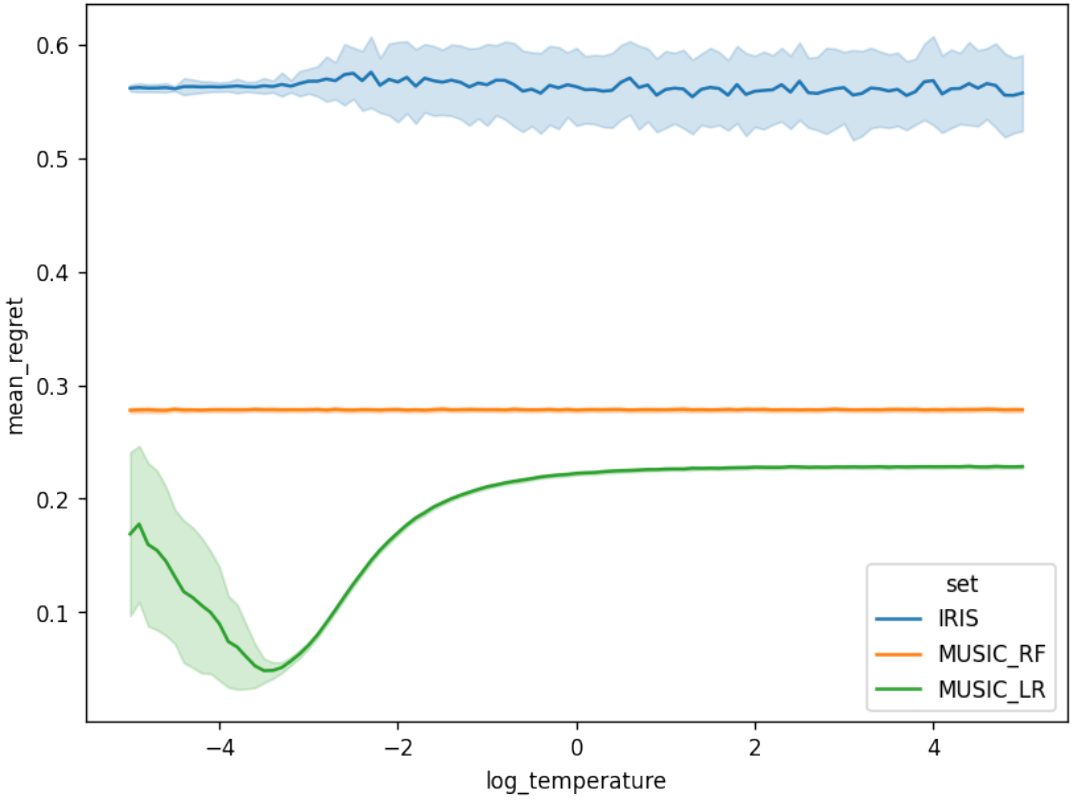
Experiment on hyper-parameter I

- Epsilon-Greedy(Eps1) Algorithm
  - No significant difference on first two datasets. However, on last one, regret drop from 0.3 to 0.05 then climb back to 0.3.



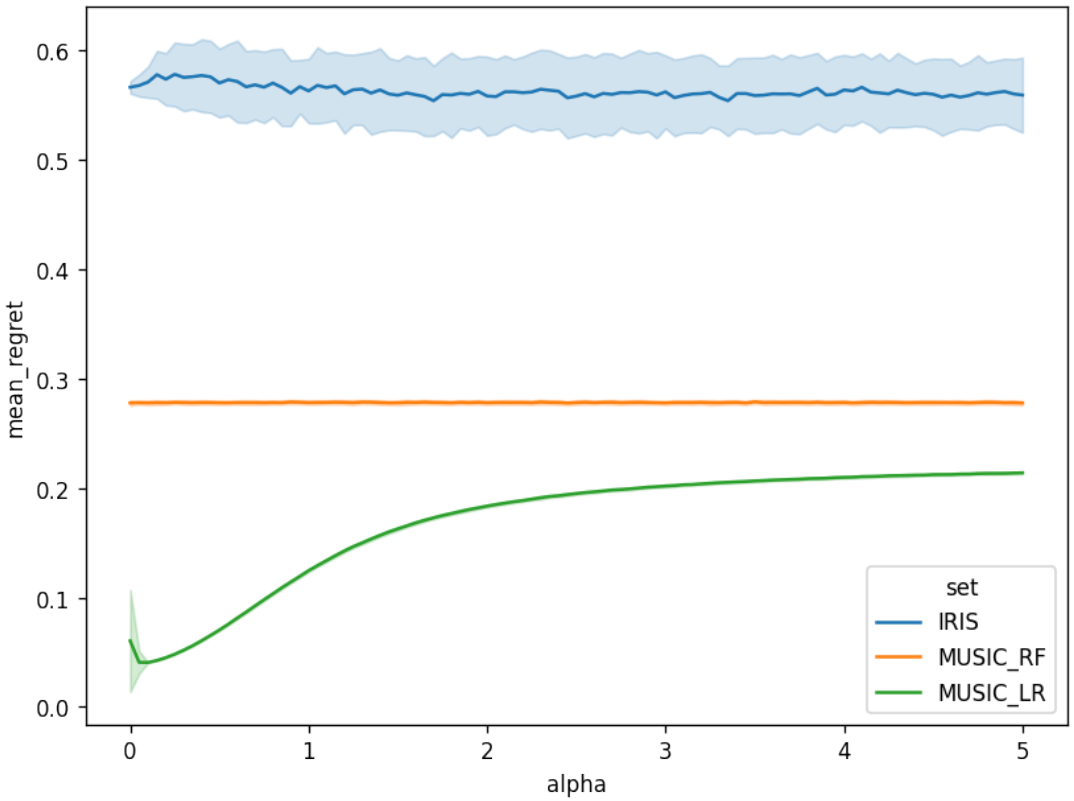
Experiment on hyper-parameter II

- Softmax Algorithm



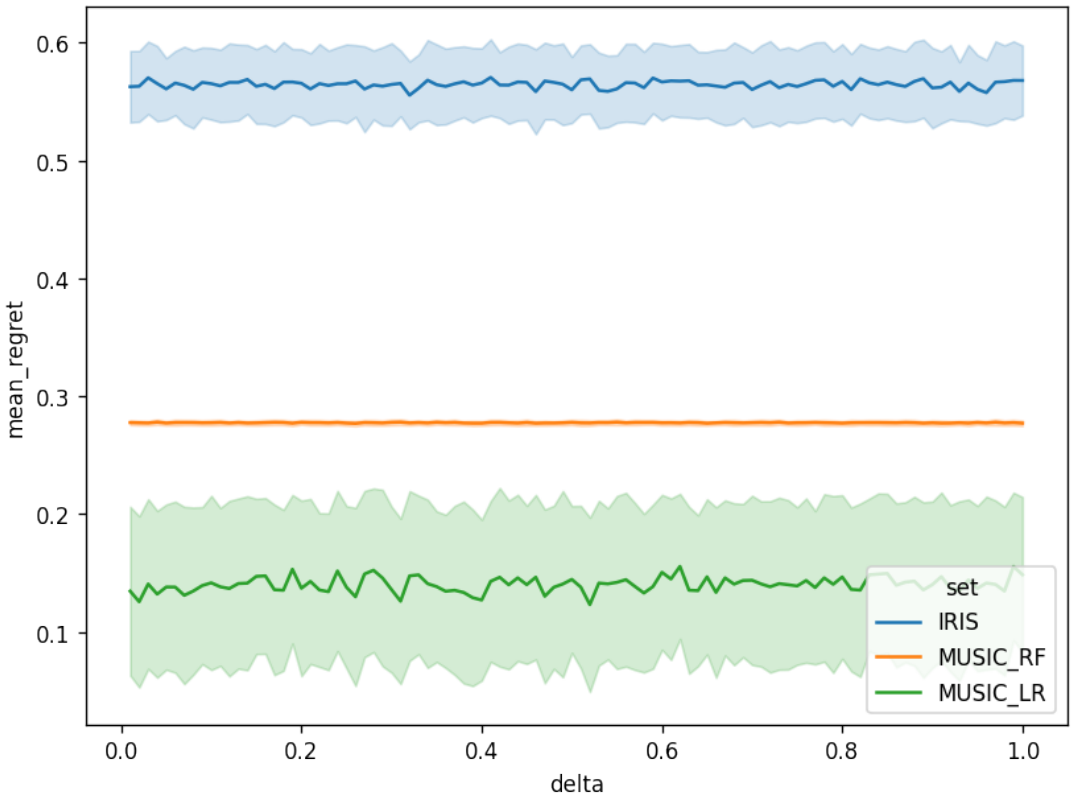
Experiment on hyper-parameter III

- UCB1 Algorithm



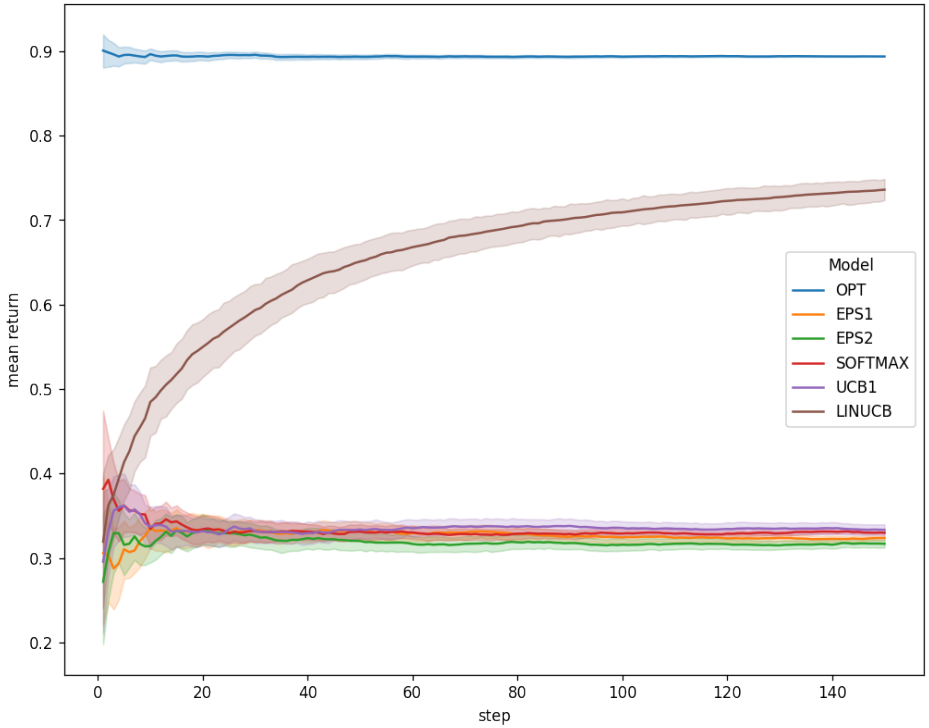
Experiment on hyper-parameter IV

- LinUCB Algorithm
  - Compared to non-contextual algorithms, LinUCB algorithm show indifferent performances with varying hyper-parameter.



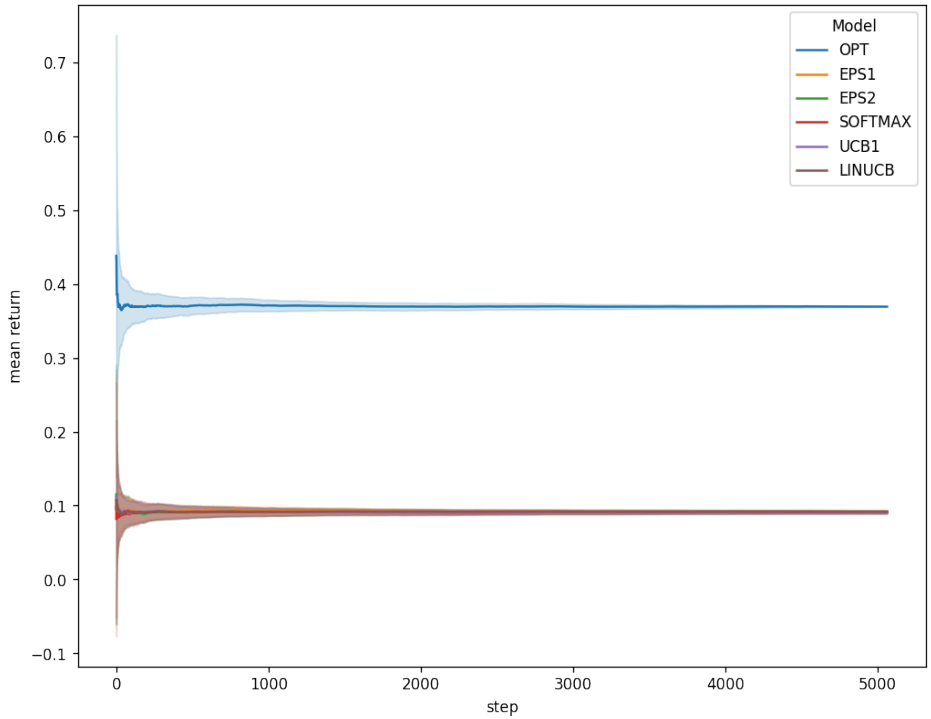
Experiment on performance I

- Performance on **IRIS** : As t increases, mean reward of LinUCB reaches 0.72, whereas, non-contextual algorithms are as low as 0.3.



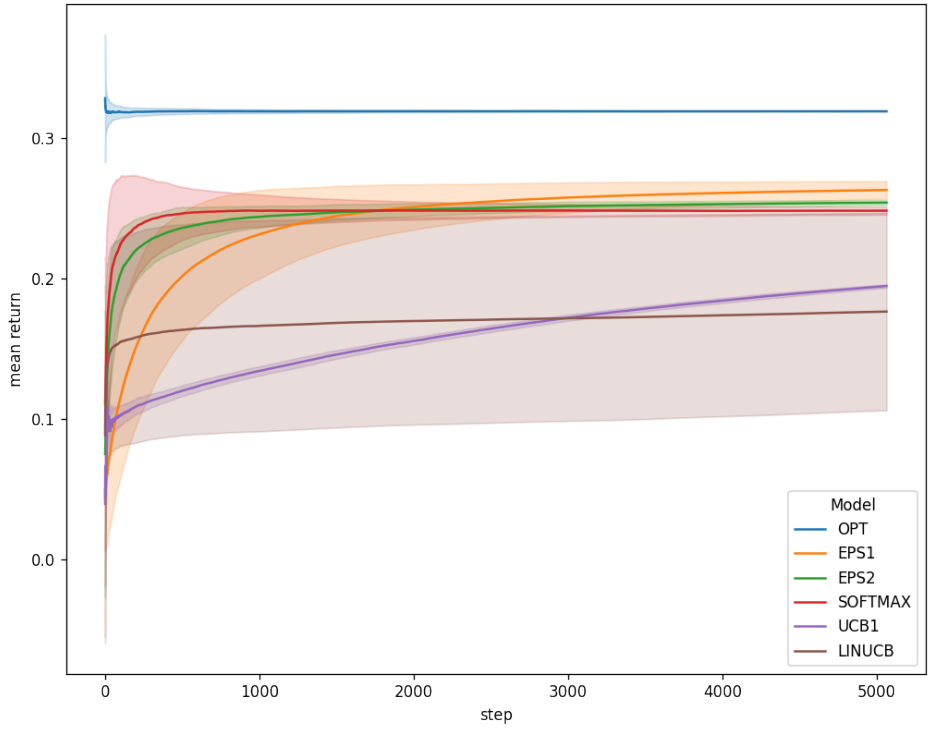
Experiment on performance II

- Performance on **MUSIC RF** : As t increases, mean reward of LinUCB reaches 0.72, whereas, non-contextual algorithms are as low as 0.3.



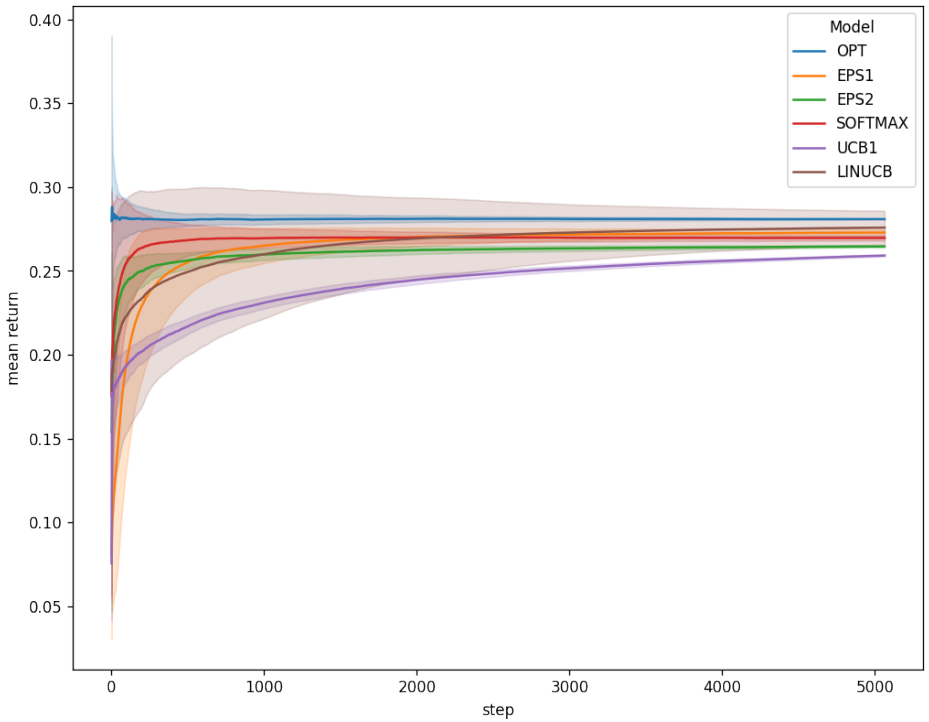
Experiment on performance III

- Performance on **MUSIC LR** : Epsilon-Greedy based models outperform UCB based models by twice as low regrets.



Experiment on performance IV

- Performance on **MUSIC LR 2**



Conclusion

- Non-contextual Algorithms are comparably sensitive to hyper-parameters. In particularly, when difference in reward is comparatively huge, performance of Non-contextual algorithms are prone to hyper-parameters(**MUSIC LR** case).
- When difference in reward is comparatively huge, simpler approach may be preferable(**MUSIC LR** case).
- If linearity assumption between context and reward holds (**IRIS, MUSIC LR 2** case), LinUCB algorithm is likely to out-perform non-contextual algorithms.
- Flexible modelling of reward and context may offer better performance.

Questions you should ask yourself when deploying bandit algorithms I

- How sure are you that you won't subtly corrupt your deployment code?
- How many different tests are you planning to run simultaneously?
- How long do you plan to run your tests?
- How many users are you willing to expose to non-preferred versions of your site?

Questions you should ask yourself when deploying bandit algorithms II

- How well-chosen is your metric of success?
- What additional information about context do you have when choosing arms?
- How much traffic can your algorithm handle before it starts to slow your site down?
- How much will you have to distort the setup we've introduced when you admit that visitors to real websites are concurrent and aren't arriving sequentially as in our simulations?

A Taxonomy of Bandit Algorithms I

- Curiosity: Does the algorithm keep track of how much it knows about each arm? Does the algorithm try to gain knowledge explicitly, rather than incidentally? In other words, is the algorithm curious?
- Increased Exploitation over Time: Does the algorithm explicitly try to explore less over time? In other words, does the algorithm use annealing?
- Strategic Exploration: What factors determine the algorithm's decision at each time point? Does it maximize reward, knowledge, or a combination of the two?

A Taxonomy of Bandit Algorithms II

- Number of Tunable Parameters: How many parameters does the algorithm have? Since you have to tune these parameters, it's generally better to use algorithms that have fewer parameters.
- Initialization Strategy: What assumptions does the algorithm make about the value of arms it has not yet explored?
- Context-Aware: Is the algorithm able to use background context about the value of the arms?

Learning life lessons from bandit algorithms

- Trade-offs, trade-offs, trade-offs
- God does play dice
- Defaults matter alot
- Take a chance
- Everybody's gotta grow up some time
- Leave your mistakes behind
- Don't be cocky
- Context matters

References I

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. *arXiv preprint arXiv:1205.2606*, 2012.

J. White. *Bandit algorithms for website optimization*. " O'Reilly Media, Inc.", 2012.