Master's Thesis
석사 학위논문

# Bandit Parameter Estimation

Sungmin Cha(차 성 민 車 成 敏)

Department of
Information and Communication Engineering

DGIST

2018

Master's Thesis
석사 학위논문

# Bandit Parameter Estimation

Sungmin Cha(차 성 민 車 成 敏)

Department of
Information and Communication Engineering

DGIST

2018

# Bandit Parameter Estimation

Advisor: Professor Suha Kwak
Co-advisor: Professor Taesup Moon

By

Sungmin Cha
Department of Information and Communication Engineering
DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics[1]

11. 24. 2017

Approved by

Professor Suha Kwak (signature)
(Advisor)

Professor Taesup Moon (signature)
(Co-Advisor)

[1] Declaration of Ethical Conduct in Research**:** I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

# Bandit Parameter Estimation

Sungmin Cha

Accepted in partial fulfillment of the requirements for the degree of Master of Science.

11. 24. 2017

| | | |
|---|---|---|
| Head of Committee | (signature) | |
| | Prof. Suha Kwak | |
| Committee Member | (signature) | |
| | Prof. Taesup Moon | |
| Committee Member | (signature) | |
| | Prof. Min-gyu Jo | |

MS/IC
201622025

차 성 민. Sungmin Cha. Bandit Parameter Estimation. Department of Information and Communication Engineering. 2018. 27p. Advisors Prof. Suha Kwak, Co-Advisors Prof. Taesup Moon

## ABSTRACT

Contextual bandit is useful algorithm for the recommendation task in many applications such as NETFLEX, Amazon Echo, etc. Many algorithms are researched and showed a good result in terms of high total reward or low regret. However, when user wants to receive a recommendation in the new task, these algorithms do not use information that learned from before task.

We suggest new topic, Bandit Parameter Estimation, to solve that inefficient problem. In the same setting with Contextual bandit, we consider $\theta^*$ as user's latent profile. And then we propose some algorithms to estimate $\theta^*$ as fast as possible.

We conducted to experiment to verify algorithms that we proposed in two case by using a synthetic dataset. As a result of experiment, we found that our algorithm estimates parameters faster than other algorithms in Contextual bandit.

Keywords: Recommendation system, Bandit algorithm, Contextual bandit, Parameter estimation

# List of contents

# List of figures

# I . INTRODUCTION

## 1.1 Overview


**Figure 1. The recommendation example in NETFLIX**

Recommendation is an important algorithm in many applications these days. For example, various recommendations such as a news recommendation on a portal site, a product recommendation on a shopping site, are provided. Therefore, it is important to give users the right recommendations.

A lot of research has been done in the form of Contextual-Bandit for a good recommendation[1][2][3]. Many algorithms have been studied, and as a result, some algorithms have resulted in fast and good recommendations to the user[1]. However, these algorithms have problem that, since these algorithms are only intended to make good recommendations in a present task, they cannot use the learned user's information from other tasks. Therefore, these algorithms have the disadvantage of learning separately for each other task.

In this thesis, we present Bandit Parameter Estimation so that the information of the user learned in the current task can be used in other tasks. In order to introduce Bandit Parameter Estimation, we will briefly explain Bandit and Contextual-bandit. Next, we will introduce some algorithms used for Bandit and Contextual-bandit such as e-greedy, UCB[4] and linUCB[1]. we will set up problems for Bandit Parameter Estimation and introduce various algorithms for a fast estimation and then verify the proposed algorithm by explaining experimental results on synthetic data. Finally, we will conclude the experimental results and suggests some future work.

## 1.2 Background

Before introducing Bandit Parameter Estimation, we talk about some basic concepts such as Multi-Armed bandit and Contextual bandit to help understand our problem setting.

### 1.2.1 Multi-Armed bandit

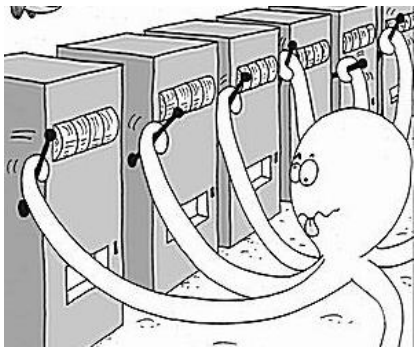

**Figure 2. The example of Multi-armed bandit**

The basic setting of Multi-Armed bandit has a set of $K$ arms, each arm has mean reward $\mu$ for $a \leq \{1, \dots, K\}$ and we can only choice the one arm at $t$. the process of Multi-Armed bandit is as follows:

| | |
|---|---|
| 1. | Init $t$ = 1 |
| 2. | Algorithm chooses an arm $a_t \leq \{1, \dots, K\}$ |
| 3. | World provides stochastic reward $r_t$, with mean $E[r_t] = \mu_{a_t}$, and independent noise |
| 4. | $T = t + 1$, repeat from Step 2 until $t = T$ ($T$ possibly unknown) |

**Figure 3. The setting of Multi-armed bandit[4]**

The purpose of an optimal algorithm to solve Multi-Armed bandit problem is maximizing a total reward so algorithm has to choice $a$ arm that has max reward in every $t$. However, comparing a total reward of algorithm is not appropriate to evaluate the algorithm because the value of a total reward varies with the value of $\mu_{a_t}$, and total reward increases continuously. Therefore, the following new measurement method is used to evaluate the algorithm, Regret :

$$R_t = \sum_{t=1}^{T} [\mu_* - \mu_{a_t}]$$

where $\mu_{a_t}$ denotes the mean reward of $a_t$ is selected at $t$ and $\mu_*$ is the mean reward of the best arm. From the regret, we can compare how fast each algorithm finds the best arm that minimize Regret as soon as possible[4].

The crucial difficulty to solve Multi-Armed bandit problem is an exploration and exploitation tradeoff. When we are in starting point($t$=0), we don't have any information of each arm. Therefore, we have to choice each arm at least once and check the reward. However, because reward is a stochastic value($\mu_{a_t}$), we cannot totally trust the reward of each arm. we can repeat an enough exploration to get a reliable $\mu_{a_t}$ of each arm. As a result, we can find the best arm but our total reward will be low because the size of $T$ is limited. If we do not explore but exploit only, we also get a low total reward with a high probability Because we do not know that the arm we choice is the best since we did not have an enough exploration[6].

In conclusion, it is most important to control an exploration and exploitation tradeoff to solve Multi-Armed bandit problem. The algorithm that finds the best arm as possible as fast will be the best algorithm. There are some algorithms that show good result so we will introduce algorithms in Related work.

### 1.2.2 K-armed (Linear) Contextual bandit

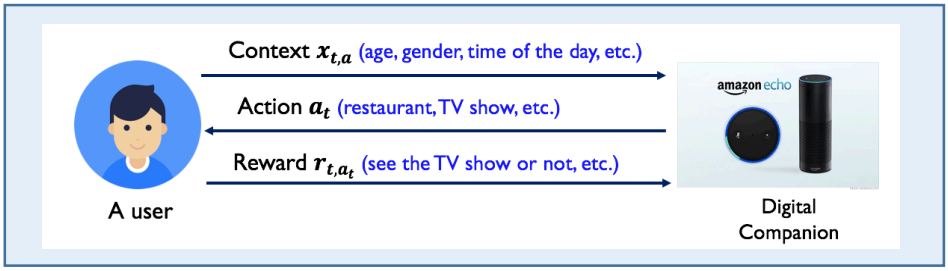


**Figure 4. Contextual Bandit**

The main difference between Bandit and Contextual bandit is that each arm has a context in Contextual bandit. Each context can be regarded as a user, article and product information for recommendation depending on the

task. Therefore, many applications use this setting to give a user a right recommendation and it is actually showing good results in many field. The problem setting of K-armed (Linear) Contextual bandit is as follows:

- **Setting**:
  1. Observe $\boldsymbol{x}_{t,a} \in \mathbb{R}^d$ for $a \in \{1, \dots, K\}$
  2. Pick $a_t$, receive the reward $r_{t,a_t} = x_{t,a_t}^T \theta^* + \epsilon_t$
  3. Incurs regret $r_{t,a_t^*} - r_{t,a_t}$ and update $\theta_t$ (e.g., $\theta_t = (\lambda I + X_t^T X_t)^{-1} X_t^T r_t$) )
- **Goal**: Maximize the reward or minimize the regret
  - ✓ $R(T) = 1/T \sum_{t=1}^{T} (r_{t,a_t^*} - r_{t,a_t})$

**Figure 5. K-armed (Linear) Contextual bandit[4]**

where $a_t^* = argmax_a \theta_t x_{t,a}$ is the best action given $x_{t,a}$, and $r_t$, $r_{t,a_t^*}$ is a best reward at time $t$. Because each arm has a context unlike Multi-armed bandit, we can consider the reward as $x_{t,a_t}^T \theta^* + \epsilon_t$. The purpose of K-armed (Linear) Contextual bandit is also maximizing the reward or minimizing the reward.

K-armed (Linear) Contextual bandit also has an exploration and exploitation tradeoff so many algorithms are suggested. The main approach of these algorithms is to solve tradeoff by using ellipsoid confidence regions. we will look into detail about ellipsoid confidence regions later.

## 1.3 Related work

In this part, we will look into popular algorithms that solve Multi-armed bandit and Contextual Bandit problem. To understand these algorithms is important since these algorithms is used as comparison target of our method.

### 1.3.1 $\varepsilon - greedy$ algorithm

$\varepsilon - greedy$ is very simple but quite a powerful algorithm for Multi-armed bandit. This algorithm just choice the best arm using the reward of each arm identified with $1 - \varepsilon$ probability or randomly selects an arm in all arms. The details of the algorithm are as follows[5]:

$$a_t = \begin{cases} a_t = argmax_{a_t}(\theta_t^T x_{t,a_t}^T), & w.p.\ 1-\varepsilon \\ a_t \sim Unif\{1, \dots, K\}, & w.p. \quad \varepsilon \end{cases}$$

**Figure 6. $\varepsilon - greedy$ algorithm**

This algorithm is not very good theoretically and experimentally, but it is very intuitive because $\varepsilon$ parameter is a term to control an exploration and exploitation tradeoff. If $\varepsilon$ is quite big, algorithm will explore many times and result in the low total reward. The opposite case can easily be considered. Another problem is that the fixed $\varepsilon$ will lead to a meaningless exploration after many trials. The reason is that we already found the best arm as a result of a sufficient exploration.

To solve a problem of $\varepsilon - greedy$, there are various variations. For example, there is a way to avoid a meaningless exploration. That is, while keeping $\varepsilon$ constantly, adaptively update or reduce $\varepsilon$ to a constant rate.

### 1.3.2 UCB

UCB(Upper Confidence Bound) algorithm is used for Multi-armed bandit. This algorithm namely set an upper confidence bound for selecting an arm. There are many variations, such as UCB1, UCB and UCB-Tuned, but we introduce UCB1 algorithm because it is a fundamental and basic algorithm. UCB1 algorithm choice an arm each time $t$ based on :

$$a_t = argmax_{a \in \{1,\dots,K\}}[\hat{\mu}_{a,t} + c\sqrt{\frac{logt}{N_t(a)}}]$$

**Figure 7. UCB algorithm**

where $\hat{\mu}_{a,t}$ is a estimated empirical mean of $a_t$ at time $t$, and $logt$ means the natural logarithm of $t$, $N_t(a)$ denotes counting number that action $a$ has been selected before time $t$, and $c$ controls the degree of an exploration[6].

The main idea of upper confidence bound (UCB) is in the square root term, that means a measure of the variance or uncertainty to estimate value of action $a_t$. Intuitively, whenever action $a_t$ is selected, the uncertainty of $a_t$ is reduced since $N_t(a)$ is increased. On the other hand, when an action other than $a_t$ is selected, $t$ is

increased but $N_t(a)$ is unchanged. The natural logarithm of $t$ means that the increase of value will small over time, but is unbounded.

UCB algorithm choice an arm each time $t$ to take into account both the estimated empirical mean and the uncertainty of $a_t$. As a result, all actions will be selected, however, an action that is selected relatively many times than others but has a low empirical mean will be not selected over time since this action do not have a value for an exploration or exploitation. In contrast, an action that is selected rarely but has a high empirical mean is relatively worthy to be chosen. Finally, after many times, an uncertainty term will be decrease, so an action that has a high empirical mean will be only selected. This algorithm has a great result in terms of theory and actually pretty work well in experiment than other algorithms. However, this algorithm has also disadvantages, that UCB have to explore all arms to compute an empirical mean of all arms since it is essential to compute UCB.

### 1.3.3 linUCB

In the K-armed (Linear) Contextual Bandit, linUCB is useful and powerful algorithm. To introduce this algorithm, we begin with similar approach used in UCB. The goal of linUCB is to set confidence regions $C_t$ that has $\theta$ with sufficient confidence. We can suppose ellipsoid confidence regions as below:

$$C_t = \{v \,|\|v - \widehat{w}_t\|_{\Sigma_t^{-1}} \leq c_t\}, \quad (1)$$

where $\Sigma_t^{-1}$ is a symmetric positive definite matrix, and:

$$\|v - \widehat{w}_t\|_{\Sigma_t^{-1}} = \sqrt{(v - \widehat{w}_t)^T \Sigma_t^{-1} (v - \widehat{w}_t)}. \quad (2)$$

For example, (2) is the standard 2-norm, and $C_t$ would be a ball of radius $C_t$ centered at $\widehat{w}_t$ when $\Sigma_t^{-1}$ is the identity matrix. Also, we can estimate $\widehat{w}_t$ by using ridge regression :

$$\widehat{w}_t = argmin_v \lambda \|v\|^2 + \sum_{t'} (\hat{r}_{t'} - v^T x_{t',a})^2, \quad (3)$$

we can consider (2) as a Gaussian confidence regions determined the parameter $c_t$, if one modeled the posterior distribution of $w$ as Gaussian with mean $\widehat{w}_t$ and covariance $\Sigma_t^{-1}$, then (2) is a high confidence region that contains $\widehat{w}_t$[4][8].

By using this ellipsoid confidence regions, UCB1-sytle algorithm choose the context as below:

$$a_t = argmax_{a\in\{1,\dots,K\}} max_{v\in C_t} v^T x_{t,a}, \quad (4)$$

this is equivalent to:

$$a_t = argmax_{a\in\{1,,K\}} \widehat{w}^T x_{t,a} + c^t \sqrt{x_{t,a-t}^{T\ -1} x_{t,a}}$$

**Figure 8. linUCB algorithm**

# Ⅱ. METERIALS

## 2.1 Problem setting for Bandit Parameter Estimation

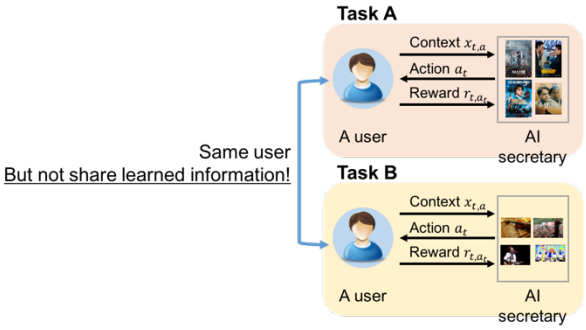


**Figure 9. The recommendation task for same user in different two tasks**

The most algorithm for contextual bandit is focused on maximizing total reward or minimizing a regret as soon as possible. As a result, some algorithm that we checked in introduction show good result in the both case. However, there is a problem that, if the algorithm makes a good recommendation to a user in the task, every time a task is changed for the same user, it must be newly learned for a good recommendation. In other words, the characteristics of the user learned in a specific task cannot be reflected in other tasks. To solve this problem, we suggest new settings for Bandit Parameter Estimation and Initial Iteration that learns user profile to rapidly adapt to a new task.

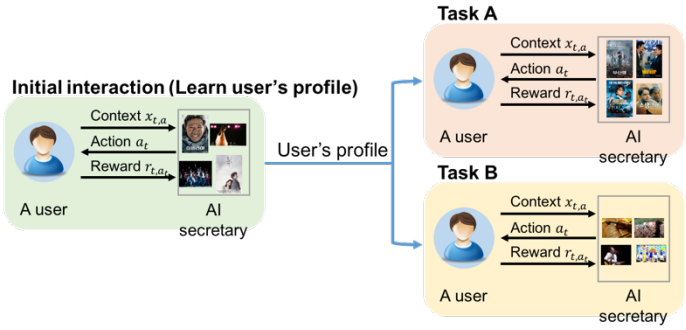


**Figure 10. New setting for learning user profile**

Our new setting is the same as Figure 10. It is similar to a situation that a user joins new web site or buy a new

artificial intelligence based secretary. In the initial iteration, we can ask a few questions about the user such as a hobby, interest and specialty. If we can learn the characteristics of the user through this process, we can quickly make good recommendations in each task. However, the point is that, since the process of the signup should not be long, it is important to learn user profile quickly.

In K-armed (Linear) contextual bandit setting, we suggest new problem called Bandit Parameter Estimation. For this problem, we begin with setting that used in K-armed (Linear) contextual bandit. However, we change the goal as below:

- **Setting**:
  1. Observe $\boldsymbol{x}_{t,a} \in \mathbb{R}^d$ for $a \in \{1, \dots, K\}$
  2. Pick $a_t$, receive the reward $r_{t,a_t} = x_{t,a_t}^T \boldsymbol{\theta}^* + \epsilon_t$
  3. Incurs regret $r_{t,a_t^*} - r_{t,a_t}$ and update $\theta_t$ (e.g., $\theta_t = (\lambda I + X_t^T X_t)^{-1} X_t^T r_t)$ )
- **Goal**: Minimize $R(T) = 1/T \sum_{i=1}^{T} L_t$ as fast as possible
  - ✓ $L_t := ||\theta_t - \boldsymbol{\theta}^*||_2$

**Figure 11. the problem setting for BPE**

We interpret $\theta^* \in \mathbb{R}^d$ as a latent user profile reflected in the initial iteration. Since $r_{t,a_t}$ is determined by $x_{t,a_t}^T \theta^*$, we can suppose that $\boldsymbol{\theta}^*$ can be said to contain the user's current interests or behavior in the current task. Therefore, if we learn $\boldsymbol{\theta}^*$ quickly in the current task, we can use $\boldsymbol{\theta}^*$ as additional information for the same user in other tasks. In this study, we aim to concentrate on finding which algorithm can quickly estimates $\boldsymbol{\theta}^*$.

## 2.2 The uncertainty ellipsoid of $\boldsymbol{\theta}^*$

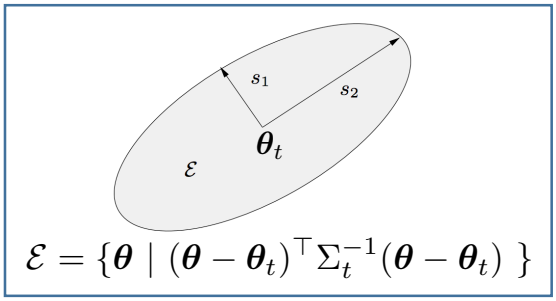


**Figure 12. The uncertainty ellipsoid of $\boldsymbol{\theta}^*$ [7]**

From the K-armed (Linear) contextual bandit and linUCB, we could induce ellipsoid confidence regions. Figure 12 shows that the ellipsoid centered at $\theta_t$ and the uncertainty ellipsoid of $\theta^*$ defined by $\Sigma_t^{-1}$. $\Sigma_t^{-1}(\Sigma_t = \lambda I + X_t^T X_t)$ is a symmetric positive definite matrix so $s_i$ in the Figure 10 is unit eigenvector of $\boldsymbol{\Sigma_t^{-1}}$. we can see that the short axis of ellipsoid (eigenvector $s_1$ corresponding $\lambda_{\max}(\Sigma_t^{-1})$)) is stretched most by sensing and the long axis of ellipsoid (eigenvector $s_2$ corresponding $\lambda_{\min}(\Sigma_t^{-1})$) is stretched least by sensing[7].
Our new goal is that estimating $\boldsymbol{\theta}^*$ as fast as possible. To do this, we have to find algorithm that shrink the uncertainty ellipsoid of $\boldsymbol{\theta}^*$ in all directions rapidly.

### 2.2.1 $\boldsymbol{Max(minEig.val)}$

The minimum eigenvalue of the uncertainty ellipsoid is the log axis of ellipsoid. As mentioned earlier, algorithm have to shrink the uncertainty ellipsoid in all directions. Selecting an arm(context) that maximize a minimum eigenvalue is lead to shrink ellipsoid in a complete circle. $Max(minEig.val)$ algorithm is as follow:

**Input :** $x_t,\ t,\ X_t$ **Output:** $x_{t,a}$

**Algorithm:**

```
arr = []
for i in range(size(x_t))
  if t = 0:
    X_t = x_t[i]
    Σ = λI + X_t^T X_t
    min_EV = getMinEigval(Σ)
    arr.append(min_EV)
  else:
    tempX_t = X_t
    tempX_t = stack(tempX_t, x_t[i])
    Σ = λI + X_t^T X_t
    min_EV = getMinEigval(Σ)
    arr.append(min_EV)
action = argmax(arr)
x_{t,a} = x_t[action]
```

**Figure 13.** $\boldsymbol{Max(minEig.val)}$ **algorithm**

Where $x_t$ denotes whole arms at time $t$, and $X_t$ denotes arms that are selected by algorithm up to time $t$. $t$ means current time. The algorithm quite simple and other subsequent algorithms will go through a similar process. First, algorithm get $x_t$ and calculate $min\_EV$ from $x_t[i]$ and $X_t$. Next, save $min\_EV$ into array. Finally, algorithm choice an arm that maximize minimum eigenvalue of $\Sigma$.

### 2.2.2 $Max(Tr(\Sigma_t))$

The trace of $\Sigma$ is same as the arithmetic mean of $\lambda_i(\Sigma_t)$. Intuitively, we can think that maximizing the trace of $\Sigma$ can lead to shrink ellipsoid. $Max(Tr(\Sigma_t))$ algorithm is as follow:

**Input :** $x_t,\ t,\ X_t$ **Output:** $x_{t,a}$

**Algorithm:**

```
arr = []
for i in range(size(x_t))
  if t = 0:
    X_t = x_t[i]
    Σ = λI + X_t^T X_t
    trace = getTrace(Σ)
    arr.append(trace)
  else:
    tempX_t = X_t
    tempX_t = stack(tempX_t, x_t[i])
    Σ = λI + X_t^T X_t
    trace = getTrace(Σ)
    arr.append(trace)
action = argmax(arr)
x_{t,a} = x_t[action]
```

**Figure 14.** $Max(Tr(\Sigma_t))$ **algorithm**

The difference with $Max(minEig.val)$ is just selecting the arm that maximize trace of $\Sigma$. However, in the face of a computational cost, this algorithm is more efficient than $Max(minEig.val)$ because the cost of getting an eigenvalue is more expensive than getting the trace of matrix.

### 2.2.3 $Min(Tr(\Sigma_t^{-1}))$

Similar with $Max(Tr(\Sigma_t))$, we could think that the harmonic mean of $\lambda_i(\Sigma_t)$. Because the arithmetic mean is always greater than or equal to the harmonic mean, maximizing the harmonic mean of $\lambda_i(\Sigma_t)$ can be effective.

**Input :** $x_t,\ t,\ X_t$ **Output:** $x_{t,a}$

**Algorithm:**

```
arr = []
for i in range(size(x_t))
  if t = 0:
    X_t = x_t[i]
    Σ = λI + X_t^T X_t
    trace = getTrace(Σ^-1)
    arr.append(trace)
  else:
    tempX_t = X_t
    tempX_t = stack(tempX_t, x_t[i])
    Σ = λI + X_t^T X_t
    trace = getTrace(Σ^-1)
    arr.append(trace)
action = argmin(arr)
```

| | |
|---|---|
| 7. $arr.append(\boldsymbol{trace})$ | 15. $x_{t,a} = x_t[\boldsymbol{action}]$ |

**Figure 15.** $\boldsymbol{Min(Tr(\Sigma_t^{-1}))}$ **algorithm**

$Min\left(Tr(\Sigma_t^{-1})\right)$ algorithm is just adding an inverse function into part of getTrace(Σ). As a result, computational cost is expensive than $Max(Tr(\Sigma_t))$ slightly.

### 2.2.4 $\boldsymbol{Max(Det(\Sigma_t))}$

Because $\Sigma_t$ is a symmetric positive definite matrix, the determinant of $\Sigma_t$ is the geometric mean of $\lambda_i(\Sigma_t)$. The geometric mean is less than or equal to the arithmetic mean, and is greater than or equal to the harmonic mean. Therefore, $Max(Det(\Sigma_t))$ will also lead to shrinkage the uncertainty ellipsoid in the all directions.

| **Input :** $x_t,\ t,\ X_t$ **Output:** $x_{t,a}$ | |
|---|---|
| **Algorithm:** | 16. *else*: |
| 8. $arr = []$ | 17. $tempX_t = X_t$ |
| 9. $for\ i\ in\ range(size(x_t))$ | 18. $tempX_t = stack(tempX_t, x_t[i])$ |
| 10. $if\ t = 0$: | 19. $\Sigma = \lambda I + X_t^T X_t$ |
| 11. $X_t = x_t[i]$ | 20. $\boldsymbol{det = getDet(\Sigma)}$ |
| 12. $\Sigma = \lambda I + X_t^T X_t$ | 21. $arr.append(\boldsymbol{det})$ |
| **13.** $\boldsymbol{det = getDet(\Sigma)}$ | 22. $\boldsymbol{action = argmax(arr)}$ |
| 14. $arr.append(\boldsymbol{det})$ | 23. $x_{t,a} = x_t[\boldsymbol{action}]$ |

**Figure 16.** $\boldsymbol{Max(Det(\Sigma_t))}$ **algorithm**

The time complexity of calculating determinant in matrix is known as $O(n^3)$. Therefore, $Max(Det(\Sigma_t))$ is a undesirable algorithm in terms of the time complexity than other algorithms we mentioned.

# Ⅲ. METHOD

## 3.1 Generating synthetic data

Before applying our algorithm to a real dataset, we generate synthetic data and verify our algorithm. A target parameter($\theta^*$) for a estimation, an initial parameter of each algorithm($\theta_t$) and $k$ contexts($x_{t,a}$) are generated based on Gaussian distribution($\mu = 0, \sigma = 1$). The size of $k$ and $d$ depends on experimental cases. For the experiment, we set two values, episode($ep$) and time($t$). Each $ep$ means a new experimental setting and $t$ means selecting one arm in an $ep$. Therefore, the target parameter($\theta^*$) and the initial parameter($\theta_t$) are generated when new $ep$ starts but $k$ arms($x_{t,a}$) are renewed at each $t$.

## 3.2 The experiment process

In order to compare the performance of each algorithm according to the change of $k$(the number of arms in each $t$) and $d$(the size of context), we proceeded in two separate experiments. They are 'Case1 : Various $k$, fixed $d$' and 'Case 2 : Various $d$, fixed $k$'. When we fixed a value($k$ or $d$), we set a value to $25$. In the case of a various value, the experiment was conducted within the range of $[10, 25, 50, 75]$. On the other hand, the other parameters are set as follows in common. The number of max episode($ep$) and time($T$) in each experimentation are respectively set to $20$ and $2500$. Also, $\alpha$ for linUCB is $0.01$ and $\lambda$ for ridge regression is $0.1$. As a result, we carried out experiments on 8 cases(but two are duplicated).

# Ⅳ. Experimental Result

## 4.1 The experiment case 1 : Various k, fixed d

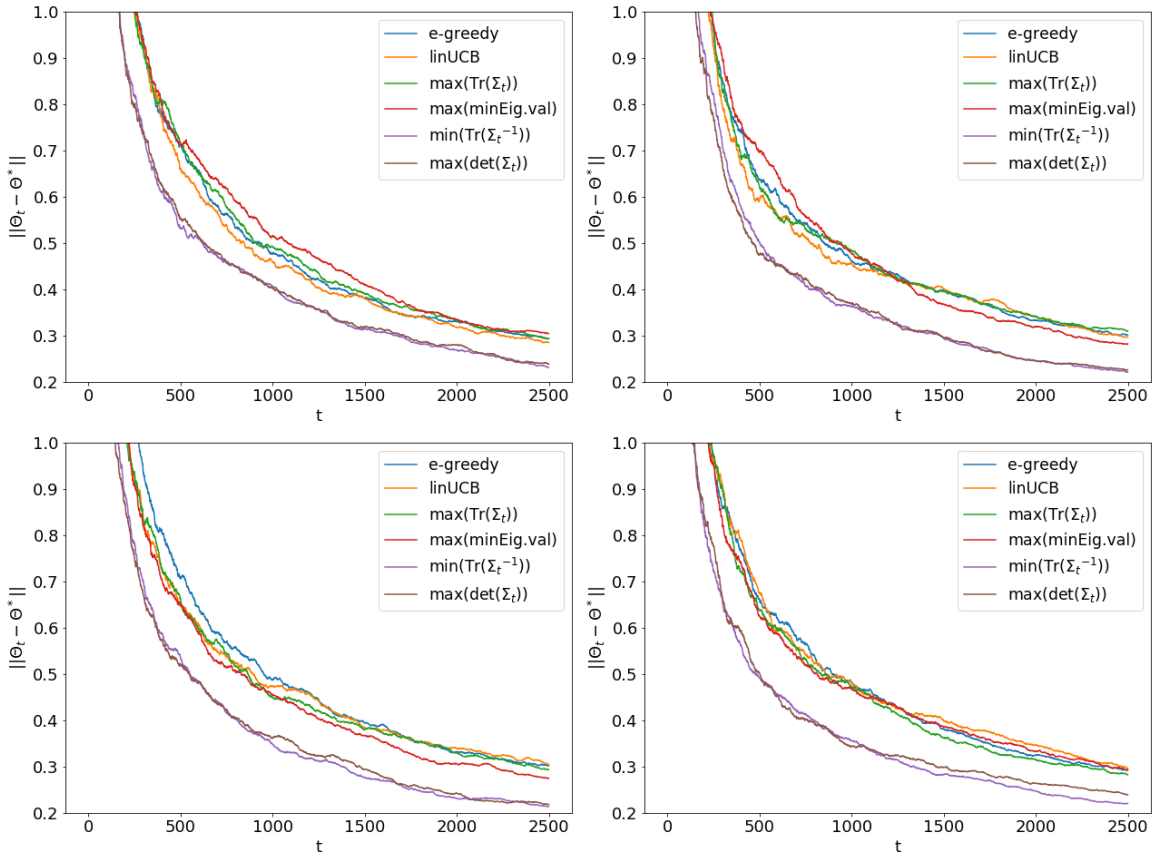


**Figure 17. Experimental results of case 1**

We fixed the value of d and experimented with changing the value of k in the range of $[10, 25, 50, 75]$. Figure 17 is a graph of the result when the experiment was carried out by gradually increasing d value from the upper left corner. The x-axis represents $t$ and the y-axis represents the norm between $\theta^*$ and $\theta_t$ of each algorithm($\|\theta_t - \theta^*\|$). While the other algorithms show similar results, we can see that $Min(Tr(\Sigma_t^{-1}))$ and $Max(Det(\Sigma_t))$ show the best result similarly from Figure 17 but the time complexity of $Max(Det(\Sigma_t))$ is quite expensive than $Min(Tr(\sum_t{}^{-1}))$. Therefore, we can think $Min(Tr(\sum_t{}^{-1}))$ is better than other algorithms to estimate $\theta^*$. In the case of linUCB(yellow line), it is confirmed that as $k$ increases, the result becomes worse

than other algorithms. However, $Min(Tr(\Sigma_t^{-1}))$ shows better results than other algorithms in the same case. For example, in the case of $d = 25$, $k = 25$(upper left graph), $\mathrm{Min}(\mathrm{Tr}(\Sigma_t^{-1}))$ achieves $\|\theta_t - \theta^*\| = 0.4$ at $t = 1000$, while the other algorithm obtain near $t = 1500$. However, in the case of $d = 25, k = 75$(bottom right graph), $t$ must reach approximately $2000$ in order for other algorithms to achieve the value achieved by $Min(Tr(\Sigma_t^{-1}))$ at $t = 1000$. As a result, $Min(Tr(\Sigma_t^{-1}))$ is more stable than other algorithms for estimating $\theta^*$, and it can be confirmed that $Min(Tr(\Sigma_t^{-1}))$ shows better results as k increases.

## 4.1 The experiment case 2 : Various d, fixed k

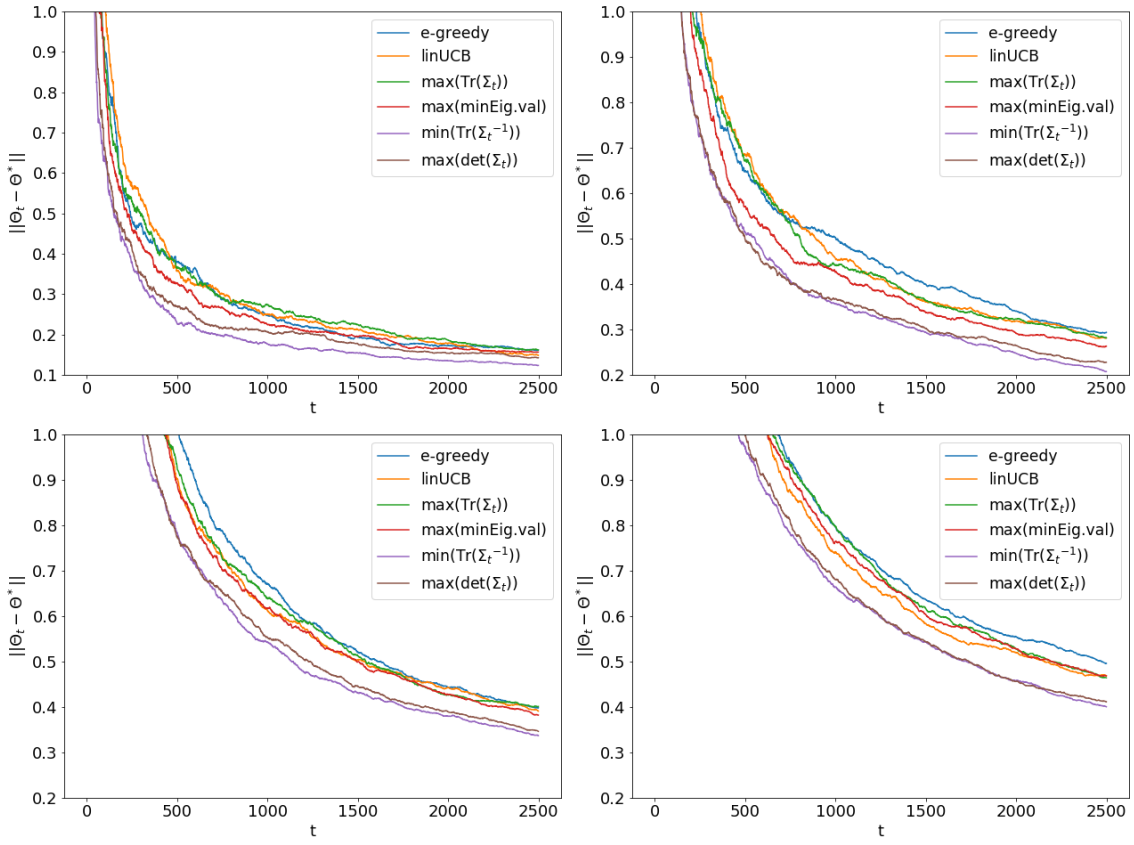


**Figure 18. Experimental results of case 2**

Figure 18 is another case in our experiment. We fixed $\mathrm{k} = 25$ and changed k in the range of $[10, 25, 50, 75]$. As above graph, we can check that $Min(Tr(\Sigma_t^{-1}))$ shows better result than other algorithms again. Especially, we can confirm that $Min(Tr(\Sigma_t^{-1}))$ gets the same $\|\theta_t - \theta^*\|$ faster by about 67% than other algorithms from

$k = 25, d = 10$ case(right top graph). linUCB shows better result than other algorithm when $d$ increase unlike experiment in case 1. However, the results of linUCB is still insufficient than $Min(Tr(\Sigma_t^{-1}))$. In conclusion, we can confirm that $Min(Tr(\Sigma_t^{-1}))$ shows the best performance even in experiments that change $d$.

# Ⅴ. Discussion

## 5.1 Conclusion

We suggested Bandit Parameter Estimation motivated by current bandit algorithms problem that cannot use information of getting from user's choice in current task when algorithm start to recommendation to user in another task. In the same K-armed (Linear) contextual bandit setting, we changed the goal that estimates $\boldsymbol{\theta}^*$ as fast as possible because $\theta^*$ can be considered as user's latent profile. We also introduced some algorithms that is to shrink the uncertainty ellipsoid rapidly in all directions and verify our algorithms by simulating in two different experimental cases. From the experimental result, we are able to confirm that $Min(Tr(\Sigma_t^{-1}))$ is most rapidly converged at the lowest value of $\|\theta_t - \theta^*\|$.

## 5.2 Future work

The motivation of our research is learning user's latent profile by estimating $\boldsymbol{\theta}^*$. We found some algorithms that estimate $\boldsymbol{\theta}^*$ rapidly from this research but don't experiment that it really help to adapt another task. Therefore, we think that there are two big topics for future work. First, we will have to verify our algorithm in real dataset. We check that our algorithm works well in case of using Gaussian based synthetic dataset. However, real dataset such as Yahoo Today's Module is different with synthetic dataset so this work is essential. Second, we will check that user's latent profile is meaningful. When we estimate $\boldsymbol{\theta}^*$ in task, we will use that as additional information of user's in another recommendation task and then adapt popular bandit algorithms to show that user's latent profile help to converge max total reward or min regret more fast than just use context.

# References


[1] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

[2] Li, Lihong, et al. "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms." *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011.

[3] Li, Lihong, et al. "An unbiased offline evaluation of contextual bandit algorithms with generalized linear models." *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*. 2012.

[4] Yue, Yisong. "A Note on Linear Bandits and Confidence Sets." (2016).

[5] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." *Machine learning* 47.2-3 (2002): 235-256.

[6] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. Vol. 1. No. 1. Cambridge: MIT press, 1998.

[7] S. Boyd and S. Lall. "EE263: Introduction to Linear Dynamical Systems, Lec25 Ellipsoid" Stanford University, 2017

[8] Y. Yue "CS159: Advanced Topic in Machine Learning, Lec4 Linear Bandits & Applications" California Institute of Technology, 2016

# 요 약 문

## Bandit Parameter Estimation

최근 많은 애플리케이션에서 사용자 맞춤형 추천을 제공하고 있다. 이때 주로 사용되는 알고리즘은 Contextual Bandit 이라는 형태로 이미 많은 연구가 진행되어 좋은 결과를 보여주고 있다. 하지만 이 알고리즘들은 특정 유저에 대해서 하나의 Task 에서는 빠르게 사용자에게 맞는 추천을 제공하고 있으나 만약 새로운 Task 에 대해 추천을 제공해야 할 때, 이전 Task 에서 학습한 정보를 이용하지 못하고 Task 별로 독립적으로 다시 학습해야 하므로 효율적이지 않다. 이러한 점에서 동기를 얻어 Contextual Bandit 과 같은 환경에서 최근 사용자의 프로필을 학습하기 위한 Bandit Parameter Estimation 이라는 형태의 새로운 문제를 제시하였다. 빠른 학습을 위하여 The uncertainty ellipsoid 을 수축하기 위한 몇 가지 알고리즘을 제시하였고 실험을 위해 만든 데이터 셋에서 제시한 알고리즘이 기존의 Contextual bandit 알고리즘보다 빠르게 Parameter Estimation 을 수행하는 것을 확인했다. 또한 향후 연구 주제로 본 논문을 통해 확인된 알고리즘을 실제 데이터에 적용하여 알고리즘을 검증하는 것 그리고 학습된 사용자의 프로필을 추가적으로 이용하여 Contextual Bandit 에 사용되는 알고리즘을 사용했을 때 프로필을 사용하지 않았을 때 보다 더 빠르게 좋은 추천을 제공하는지 확인하는 연구가 필요하다는 것을 제시하였다.

핵심어: Recommendation, Bandit, Contextual Bandit, Parameter Estimation