이학석사 학위논문

Stochastic Contextual Linear Bandit Models and Applications to Mobile Health Problems

확률 맥락적 선형 밴딧 모형과 모바일 건강 문제에의 응용

2021년 2월

서울대학교 대학원

통계학과

예 성 주

Stochastic Contextual Linear Bandit Models and Applications to Mobile Health Problems

지도교수 장 원 철

이 논문을 이학석사학위논문으로 제출함

2020년 10월

서울대학교 대학원

통계학과

예 성 주

예성주의 이학석사 학위논문을 인준함

2021년 1월

위 원 장	Myunghee Cho Paik	(인)
부위원장	장 원 철	(인)
위 원	임 채 영	(인)

Abstract

Sung Ju Yea
The Department of Statistics
The Graduate School
Seoul National University

Contextual Bandit models utilize information of a Bandit game player to optimize the player's benefit. From the perspective of applications, it is preferred to adopt stochastic linear models in contextual Bandit problems for simple implementations. Hence, stochastic contextual linear Bandit models are widely used in real-world problems such as news article recommendations. Recently, mobile health problems are suggested as another applicable area of contextual Bandit models. However, for real-world applications, we must consider various aspects of the implementation of algorithms that are run on mobile devices. In this thesis, we will briefly review stochastic contextual linear Bandit algorithms, and discuss their applications to mobile health problems. Especially, we will inspect thoroughly the aspects that must be considered when we develop a mobile health application for real-world users.

Keywords: Contextual Multi-Armed Bandit, Reinforcement Learning, Mobile Health Problems
Student Number: 2019-29758

Contents

1 Introduction

Recent rapid developments in machine learning technologies attract interest from both academic researchers and industrial engineers. One of such areas of research and applications is sequential decision making problem over time, concerning a player learning optimal choices at each time point of action. Bandit problems consist of a part of machine learning areas for handling this problem. Especially, stochastic linear Bandit models with contexts have the advantage over other Bandit models in that they are not only easy to implement in an application, but also enjoy good performances in terms of benefit for players.

Almost PC-like mobile devices these days provide opportunities for Bandit models' applications, some examples of which are mobile health problems. In an era when almost every person has at least one mobile device at hand, a great portion of the population in the world can play actions over time to improve their health, with help of mobile devices. However, the real-world condition is not simple as it seems, and it requires several serious considerations before transforming a Bandit model into actual services for users.

In our text, we will briefly review essential concepts of Bandit problems. We will especially emphasize stochastic contextual Bandit problems, which adopt linear models for the sake of applications. There will also follow a discussion about how to implement those Bandit algorithms for mobile health problems in the real-world.

2 Overview of Stochastic Contextual Bandit Problems

2.1 Bandit Problems

As they are well-known, Bandit Problems deal with a situation where a person tries to maximize one's own profit in playing more than two slot machines. Since one's own money is scarce, the person had no choice but to make a strategy of figuring out which machine gives the best result compared to other machines. Here one faces what is called *exploitation versus exploration trade-off*, which means that one needs to try some other machines that had not been played before(=exploration). Newly chosen machines may bring better or worse results than the current machines the person is currently playing with(=exploitation).

This is a brief introduction of what Bandit Problems are. Surely, in the real-world problem, things come in much more sophisticated ways, thus we need to set up problems in formalized ways. Here we introduce some terminologies and concepts which are commonly used in Bandit literatures. For this part, we will consult comprehensive Bandit textbooks([6], [8]). First, from our brief introduction in the previous paragraph, there are four key points that must be considered: *slot machines, time points that choices of machines are made, choices of optimal machines*, and *results from playing a chosen machine that give him back some money.* In the context of decision theory, we can think of machines as possible actions that are to be chosen in each step, which are usually called *arms* in Bandit literatures. Time points are called *rounds* at which those arms are chosen. The behavior of choosing arms can be considered as a strategy or an *algorithm.* The money we get from playing a machine at a given round is called 'reward' from that machine at that round. In this text, we use the following notations for those key concepts and can simply summarize basic multi-armed Bandit Problems:

Algorithm 1: Protocol: Basic Multi-Armed Bandit Problem(adapted from [8])

Given *finite* arms $i = 1, 2, ..., K$ and *finite* rounds $t = 1, 2, ...T$,

for t = *1,2,...,* T **do**

The player chooses an optimal arm $a_t \in \{1, 2, ..., K\}$

Observe a reward $r_{t,a_t} \geq 0$

end

As a typical example, we may think of a news website where daily headlines come up at the very front page of the site(such as Li et al., 2010 [7]). As the business owner of this website, one tries to maximize click rates of those headlines. Hence, the site must choose articles that are expected to attract as many visitors as possible. We can see that this case can be interpreted as another Bandit problem. Arms are news articles that would be exposed in the main page, rounds are each visitors that may click the articles or not, and rewards are actual clicking behaviors by the visitors.

Next, we will clarify what it means by 'optimal arm' in the protocol introduced above. Every Bandit problem aims at maximizing what is called *regret*. As we can read from the word's meaning, the regret of a Bandit algorithm is a quantity of how much a player gain over a period of playing the Bandit game compared to the optimal decision making. The exact definition is the following: For $\mu_t^* = \underset{1 \leq i \leq K}{max} E(r_{r,i})$, regret $R(T)$ is defined as

$$R(T) = \sum_{t=1}^{T} (\mu_t^* - E(r_{t,a_t}))$$

Hence, regret $R(T)$ compares the player's actual action with the optimal action at each round t. Every Bandit problem tries to minimize the expectation of this regret,

since an action a_t may be chosen randomly by the algorithm at t. In Bandit literatures, usually regret is expressed asymptotically with big-o notations. Of course, as we already have mentioned earlier, we always face the *exploitation versus exploration* dilemma when it comes to optimizing our regret. Here we introduce a simple Bandit algorithm ϵ-*Greedy*, which explicitly describes this dilemma(see [8] for details):

Algorithm 2: $\epsilon -$ Greedy Algorithm

Given K and T,

for t = *1,2,...,T* **do**

  Observe $X \sim Bernoulli(\epsilon_t)$

  if X = *1* **then**

    exploration: choose an arm with probability of $\frac{1}{K}$

  else

    exploitation: choose an arm with highest average reward by the round $t-1$

  end

end

So far was a short introduction to Basic Bandit Problems. However, they can be developed into more complicated forms depending on how we elaborate the key points: arms, choice of arms, and rewards. Although in this text we assume that arms are finite for practical reasons, we may consider a continuous set for the variable of arms. Decision making at each round can consider external factors such as *contexts*, which may define an optimal state at that round. Finally, we may consider situations in which rewards are gained in randomized ways(*Stochastic*) or we assume that an adversarial nature already had chosen the rewards for each of the arms at all rounds to come(*Adversarial*). All these complicated considerations give birth to various Bandit problems and algorithms. Typical examples of algorithms will be covered in the following sections.

2.2 Stochastic versus Adversarial Bandit Problems

As we have seen above, Bandit problems can be classified into two categories: *stochastic* and *adversarial* problems. This classification is based on different assumptions for each of the problems. A stochastic model assumes that rewards are generated from probability distributions, varying in arms that the player chooses. This assumption usually comes along with the parametric assumptions, such as, those distributions are of subgaussian, etc. Parametric assumptions are for practical reasons(easy implementation) as well as for theoretical simplification.

On the other hand, it can be suggested that these assumptions are too specific and may cause lack of fit, which is an innate problem of model construction. Compared to this model specification issue, adversarial models enjoy much freedom from their rather robust assumption. In an adversarial model, rewards are already selected over time. Hence, no randomized feature comes out of this branch except for the behavior of the player: the player must make his decision of choosing arms randomized for the purpose of safety against the adversarial player. However, this does not mean that choices of arms are totally random. At each round, the player estimates future rewards for each of the arms to make the best choices.

Let us see a few basic examples for each of these two branches. First, we here present *UCB1*(upper confidence bound) algorithm, which is suggested by Auer et al., 2002([2]).

Algorithm 3: UCB1 Algorithm(adapted from [6])

Given $K, T,$ and $\delta > 0$ (hyperparameter),

for t = *1,2,...,T* **do**

Let $UCB_i(t-1) =$

$$\begin{cases} \infty & \text{if } T_i(t-1) = 0 \\ \widehat{\mu_i}(t-1) + \sqrt{\frac{2log(\frac{1}{\delta})}{T_i(t-1)}} & \text{otherwise} \end{cases}$$

where $T_i(t-1)$ is the number of round that the arm i has been played by the round $t-1$, and $\widehat{\mu_i}(t-1) = \frac{1}{T_i(t-1)}(\sum_{t:a_t=i} r_{t,i})$

Choose $a_t = \underset{1 \le i \le K}{argmax} UCB_i(t-1)$

Observe r_{t,a_t}, and update UCB_i for all arms $i = 1, 2, ..., K$

end

As the name of the algorithm implies, at each round UCB1 algorithm chooses an arm which promises the highest reward in an optimistic sense. By *optimistic* here we mean that the algorithm considers the upper confidence bounds of each of arms' rewards computed from the past history of the rewards. Also, what is essential here is that we are computing *confidence interval* for each of the arms' rewards, which implies this model is obviously stochastic. Especially, the above algorithm assumes that the rewards follow subgaussian distributions.

Next, let us move onto one of adversarial models, *Exp3* algorithm, introduced by Auer et al., 1995([3]). We can see that the algorithm estimates even the rewards of the arms that the player did not select, for the future selection of the arms. Also, we can say that this algorithm is more simple in terms of model assumptions compared to the UCB1 above.

Algorithm 4: Exp3 Algorithm(This is an adapted version by [6])

Given $K, T,$ and $\eta > 0$ (hyperparameter),

Initialize $\widehat{S}_{0,i} = 0$ for all arm i

for t = *1,2,...,T* **do**

Let $P_{t,i} = \frac{exp(\eta \widehat{S}_{t-1,i})}{\sum_{j=1}^{K} exp(\eta \widehat{S}_{t-1,j})}$

Pick $a_t \sim P_t = (P_{t,1}, ..., P_{t,K})^t$

Observe r_{t,a_t}

Update $\widehat{S}_{t,i}$ from $\widehat{S}_{t-1,i}$ and r_{t,a_t} such that

$$\widehat{S}_{t,i} = \widehat{S}_{t-1,i} + 1 - \frac{I(a_t = i)(1 - r_{r,i})}{P_{t,i}}$$

where $1 - \frac{I(a_t=i)(1-r_{r,i})}{P_{t,i}}$ is an unbiased estimator of $r_{t,i}$

end

2.3 Contextual Bandit Problems

By far, we have discussed possible conditions of the Bandit problems. Given K arms, our algorithm chooses an optimal arm(either stochastic or adversarial), and the player observes rewards r_{t,a_t}. These rewards are used to update the parameters for the future decision making. However, in the real-world, we need to consider some external information which is relevant to our decision making. This information is usually called *context* in Bandit literatures.

For example, let us recall the news article recommendation problem introduced in the section 2.1. Clearly, what articles a reader choose are closely related to one's own age, gender, website log-on time, some special events occurring at that time(such as natural disasters or political events), log-on location, local weather, and so on. Hence

in a very hot summer, a news article would very likely to attract attention from the readers if it deals with the global warming issue. In contrast, this topic would not be much discussed in a mild and temperate whether.

Now our Bandit algorithm must consider these contexts. The same arm would may give back different rewards in different contexts. Let $\mathcal{C}$ be the set of context vectors c_t at round t, which are available over the time period of our bandit problem. Thus, rather than directly picking an optimal arm i at the round t, we need to estimate an optimal function from $\mathcal{C}$ to the set of arms $\{1, 2, ..., K\}$, which also may be depend on t and the history in the past(hence it may be dependent on $c_1, ..., c_{t-1}, r_{1,a_1}, ..., r_{t,a_{t-1}}$). One thing to note is that context c_t is considered as a deterministic input, by which we mean that contexts are not variables affected by actions a_τ or rewards $r_{t,a_\tau}(\tau = 1, 2, ..., t-1)$.

Therefore, either stochastic or adversarial, we extend the basic Bandit problem protocol in the section 2.1 to contextual ones as the following:

Algorithm 5: Protocol: Basic Contextual Bandit Problem(modified from [8])

Given $K, T,$ and $, \mathcal{C}$

for *t = 1,2,...,T* **do**

Observe the context $c_t \in \mathcal{C}$

The player chooses an optimal arm a_t, considering c_t and the past history

$c_j, r_{j,a_j}(j = 1, 2, ..., t-1)$

Observe the reward $r_{t,a_t} \geq 0$

end

3 Algorithms: Linear UCB and Linear Thompson Sampling

3.1 From Application's Perspective: Linear Models

In the previous chapter, we briefly reviewed the basic concepts of Bandit problems, among which we emphasized what stochastic, adversarial, and contextual mean and why they are important. In this chapter, we will be in a more practical point of view, and narrow down our interests in Bandit Problems to a rather specific branch: Stochastic Linear Bandit Problems with Contexts. First of all, we will discuss why we prefer this stochastic linear models to other kinds of Bandit models. After then, we will introduce two classical examples: Linear UCB(LinUCB) and Linear Thompson Sampling model(Linear TS).

Why Stochastic? We know that a stochastic model assumes that a reward from an arm at a certain round is obtained from a probability distribution. If an engineer tries to choose a Bandit model, he would rather choose this stochastic model than an adversarial model, because 1)it is easy for implementation. Since a stochastic model is usually deployed as a parametric model, it can enjoy plenty of statistical or machine learning related techniques. And 2)if it is a correct model, it shows a better performance than adversarial models. This fact can be easily proved, using Jenson's inequality and the convexity of the max function(see [6]).

Why Linear? Similarly, a practitioner would definitely prefer linear models. Most of linear models only involve the matrix algebra, and most of the matrix algebra algorithms are well supported by high-performing modern computer programs(for example, we have BLAS or LAPACK in C language which are also adopted by C - family languages such

as C++'s *Armadillo* package or Swift's *Accelerate* framework). Moreover, linear models are simple in optimization, so that there is no need to implement complicated techniques like Stochastic Gradient Descent in Deep Neural Network algorithms.

For these reasons, many of practical researches on Bandit problems make discussions about stochastic linear models([5], [7]). However, what is exactly linear in a stochastic model? Since we assume that a stochastic model adopts a parametric approach, a reward $r_{t,i}$ follows a certain parametric distribution $\mathcal{D}_a$. We suppose this distribution has its mean as *linear*, so that $E(r_{t,i}) = \theta_i^T b(c_t, i)$, where $b(c_t, i)$ is called a *feature vector*(which means it captures *true feature* from the arm i in the context c_t), and θ_i is the weight vector for our model, which keeps updated as the round t goes forward.

3.2 Linear UCB

As a next step, we introduce the Linear UCB(LinUCB) algorithm, first introduced by Li et al., 2010([7]). Like UCB1, LinUCB calculates upper confidence bound, but here linearity assumption involves so that the form of UCB contains matrix operations. Practically, it was proposed as a recommendation system algorithm, which verifies our argument that stochastic linear models are suitable for applications. Although there are several developments afterwards, here we focus on the very basic algorithm that Li et al., 2010([7]) introduced. Here, a reward $r_{t,i}$ is a bounded positive real-number, and d is the dimension of the feature vector $b_{c_t,i}$ and the weight vector θ_i.

Algorithm The LinUCB algorithm is as follows:

Algorithm 6: Linear UCB(Li et al., 2010([7]))

Given $K, T,$ and $\alpha > 0$,

Initialize $A_i = I_d$, $f_i = 0_{d\times 1}$ for all $i = 1, 2, ..., K$

for t = *1,2,...,T* **do**

  Observe the context $c_t \in \mathcal{C}$

  for i = *1,2,...,K* **do**

    Create the feature vector $b(c_t, i) \in \mathbb{R}^d$

    Update $\theta_i \leftarrow A_i^{-1} f_i$

    Compute $UCB_{t,i} = \theta_i^T b(c_t, i) + \alpha\sqrt{b(c_t, i)^T A_i^{-1} b(c_t, i)}$

  end

  The algorithm chooses an optimal arm $a_t = \underset{1\le i\le K}{argmax} UCB_{t,i}$

  Observe $r_{t,a_t} \geq 0$ and update:

  $A_{a_t} \leftarrow A_{a_t} + b(c_t, a_t)b(c_t, a_t)^T$

  $f_{a_t} \leftarrow f_{a_t} + r_{t,a_t} b(c_t, a_t)$

end

Essential Idea First, as the name shows, LinUCB shares the same idea with UCB1 in the section 2.2. That is, for each round t, given context c_t, with updated θ_i of each arm i from the previous history(actions, rewards, and contexts by the round $t-1$), LinUCB chooses the arm a_t which has the greatest $UCB_{t,i}$. However, the way the algorithm calculates upper confidence bounds is where the linearity assumption is required. A detailed proof can be found in such as Chu et al., 2011([12]). But the proofs do not tell what motivation is underlying. Here we will see a simple case where rewards follow gaussian distribution, so that we could get an idea of how to obtain such UCBs. Suppose for fixed arm i, we have $T_i(t)$ number of data that is obtained by the round t with d features(that is, by the round t the arm i has been chosen $T_i(t)$ times). Let $D_i(t)$ be

the $T_i(t) \times d$ matrix, each of the row of which is the feature vector $(1, b(c_\tau, i)^T)^T (\tau = 1, 2, ..., T_i(t))$ augmented with 1 in the first element position(this is what is called *design matrix* in Linear Regression problems). Also, let $y_i(t) = (r_{1,i}, r_{2,i}, ..., r_{T_i(t),i})^T$. In this setting, a simple way to choose the best arm at the next round $t + 1$ is to predict the reward. If we assume that the expected reward from the arm i is *linear* in the feature vector $b(c_t, i)$, one of reasonable solutions is the Linear Regression, and we get the following estimator of θ_i at the round t:

$$\widehat{\theta}_i = (D_i(t)^T D_i(t))^{-1} D_i(t)^T y_i(t)$$

Hence, if we here assume that the rewards $r_{\tau,i}$ are *i.i.d. gaussian* with the mean $\theta_i^T b(c_\tau, i)$ and the standard deviation $\sigma > 0$, we obtain:

$$\widehat{E(r_{\tau,i})} = \widehat{\theta}_i^T b(c_\tau, i) \sim N(\theta_i^T b(c_\tau, i), \sigma^2 b(c_\tau, i)^T (D_i(t)^T D_i(t))^{-1} b(c_\tau, i))$$

Here we can see how the linear assumption and the gaussian assumption merge, to obtain the upper confidence bound of the reward $r_{t,i}$. If we simply follow this argumentation, then we will have our UCB as $\widehat{\theta}_i^T b(c_\tau, i) + \alpha \sqrt{b(c_\tau, i)^T (D_i(t)^T D_i(t))^{-1} b(c_\tau, i)}$, which has a similar form with what we see in the algorithm. If we follow the argumentation in Li et al., 2010([7]) that the algorithm adopts Ridge Regression, then we could not obtain such a result above, due to the fact that Ridge Regression provides an unbiased mean estimator. Therefore, a proof such as Chu et al., 2011([12]) starts without gaussian assumption but uses other techniques such as Azuma's inequality.

Exploitation versus Exploration The difference between UCB1 and LinUCB is not only the ideas of linearity. The algorithm chooses an arm if and only if the UCB of that arm is the optimal. Since UCB of an arm is subdivided into the mean part and the confidence interval part, exploitation and exploration is not explicitly separated as

UCB1, and may work together to bring the player the minimum regret(surely, it could be also that two concepts are in trade-off relation).

Summary The LinUCB algorithm mainly adopts the ideas of linearity of the reward distribution, which guarantees easy implementation in application. This algorithm also shows a good performance, with the regret bound of $\tilde{O}(\sqrt{Td})$([12]).

3.3 Linear Thompson Sampling

Compared to LinUCB, Linear Thompson Sampling(Linear TS) distinguishes itself with its Bayesian setting, accumulating the information at each round in the weight vector θ_i for each arm i. Each of weight vectors has its prior as gaussian distribution. Moreover, this algorithm lucidly expose the gaussian assumption of rewards, so that by conjugacy, computation of the posterior part is simply made. The name of the algorithm comes from Thompson, 1933([10]), which suggested the basic idea of Bandit problems with a Bayesian way of solution.

Algorithm In our text, we follow the setting of Agrawal and Goyal, 2013([1]), with some modification consulting Greenwald et al., 2017([5]), allocating the weight vector for each of the arms. We use the same notations for this Linear TS as in LinUCB.

Algorithm 7: Linear Thompson Sampling(Agrawal and Goyal, 2013([1]))

Given $K, T,$ and $\alpha > 0$,

Initialize $A_i = I_d, f_i = 0_{d\times 1}, \mu(\theta_i) = 0_{d\times 1}$ for all $i = 1, 2, ..., K$

for t = *1,2,...,T* **do**

  Observe the context $c_t \in \mathcal{C}$

  for i = *1,2,...,K* **do**

    Create the feature vector $b(c_t, i) \in \mathbb{R}^d$

    Generate $\theta_i \sim N(\mu(\theta_i), \alpha^2 A_i^{-1})$

  end

  The algorithm chooses an optimal arm $a_t = \underset{1\leq i\leq K}{argmax}(\theta_i^T b(c_t, i))$

  Observe $r_{t,a_t} \geq 0$ and update:

  $A_{a_t} \leftarrow A_{a_t} + b(c_t, a_t)b(c_t, a_t)^T$

  $f_{a_t} \leftarrow f_{a_t} + r_{t,a_t} b(c_t, a_t)$

  $\mu(\theta_{a_t}) = A_{a_t}^{-1} f$

end

Essential Idea The basic ideas of Linear TS are linearity, normality assumptions and the Bayesian inference. The normality assumption of rewards affects the way the variables are updated due to the conjugacy of gaussian distribution in the Bayesian inference. In LinUCB, the Linear Regression idea underlies in the update of the variables, which can be shown by a few matrix algebra concerning how to compute the matrix of the form $(I + ab^T)^{-1}(a, b$ are vectors)(As a matter of fact, Li et al., 2010([7]) refers to the Bayesian way of interpreting the result of the paper, which shows a close relation between LinUCB and Linear TS).

As the round goes on, the weight vector θ_i is updated by the Bayesian inference, storing the information from the history by the previous round. In this way, the algo-

rithm can show which variables of the context vectors are more influential on the reward of an arm. After observing a realized random value of the weighted vector of each of the arms, the player chooses an optimal arm that shows the largest expected mean value of the reward of that arm(conditioned by the weighted vector's realized value). The reward obtained is used for updating the weighted vector of each of the arms.

Exploitation versus Exploration We may view this algorithm as a sophisticated version of ϵ-Greedy algorithm, in a sense that it requires randomization before making the decision of whether to explore or exploit. However, contrary to ϵ-Greedy where randomization's result directly decides exploration without considering the past, Linear TS's exploration is largely affected by the history, the information of which is stored in the weight vectors.

Summary Linear TS is a intuitive model in a sense that it accumulates information from the past to make future decisions, by its Bayesian idea. Linear TS enjoys its regret bound of $\tilde{O}(\frac{d^2}{\epsilon}\sqrt{T^{1+\epsilon}})(0 < \epsilon < 1)$([1]).

4 Applications: Mobile Health Problems

Now is the time to move onto the real-world. By now, we have briefly looked through the basic concepts of Bandit problems, and especially focused on the Stochastic Linear Bandit models with Contexts for applications. One of such applicable area of our contextual bandit models is *mobile health*. In the following sections, we will discuss overall mobile health problems and the specific ways of how to apply the Bandit models to them, considering many of discourses on this topic(such as [5], [9], [11]).

4.1 Mobile Health Problems

Above of all, we need to clarify what exactly mobile health problems are. There could be several versions of these, but here in our text, we will define them as *developments of suitable algorithms which are run on a person's own mobile device such as a mobile phone, such that the algorithms help to foster the person's health-related activities like walking, sleeping, taking medicine, etc.*

In an age where the pace of development of technology is faster than ever, the importance of mobile health problem gets bigger. There are at least four reasons for this. First, the ubiquity of mobile devices means mobile health problem can affect such a many individuals nowadays. Second, today's high performing devices compared to the past can run even heavy software programs such as online video games, which require high level of CPU and GPU. If we can use such computing resources for our leisure, there is no reason that we hesitate to use them for our own health. Third, from morning to night, many of mobile device users are engaged in their own devices deeply. There are a myriad of services such as financial support, schedule management, social entertainment provided through the mobile devices. Hence, it could be our opportunity to deploy this heavy engagement for the user's health improvement. Finally, as Internet is commonly used, the private information issue needs to be coped with very seriously. If we use each of the individual's health data within one's own devices only, then the safety of the information could be well guaranteed than the case in which we run a central server as our database.

Of course, already there are some movements for this mobile health problems, and the manufacturing companies like Apple provide iOS developers with their own software

development kit such as Healthkit[1], for easy implementation of health-related services which may require users' personal health data.

4.2 Stochastic Linear Bandit Model with Contexts for Mobile Health Problems

There could be many of machine learning algorithms for application to mobile health problems. In this section, we will discuss how stochastic linear Bandit models with contexts distinguish themselves from the others. There are at least three points to be noticed. First of all, as we already have seen in the section 2.1 with an example([7]), contextual Bandit models are quite suitable for recommendation system problems. Taking into account users' contexts, contextual Bandit models choose optimal actions over a period of time. As Tewari and Murphy, 2017([9]) points out, in terms of mobile health problems, algorithms recommend the best actions for users at each time point of activity engagement. Secondly, every Bandit model is basically a sequential decision making process model over a certain period of time. Tracking users' behaviors over time, it tries to maximize the overall benefits of users throughout the whole period, balancing between exploration and exploitation of the available actions. Finally, linear models are desirable than other machine learning models, due to the limit of the current mobile devices' CPU and GPU performances. Hence it would be better to adopt parametric and simple models for mobile applications. The algorithms such as Linear TS or LinUCB that we have covered in the previous chapter can be examples for such applicable algorithms.

Now, we develop a possible protocol of stochastic linear Bandit model with contexts for mobile health problems, from the basic contextual Bandit protocol shown in the section 2.3.

[1]https://developer.apple.com/documentation/healthkit

Algorithm 8: Protocol: Stochastic Linear Bandit Model with Contexts for Mobile Health Problems

Given K, T,

Initialize the parameters of the model, which is linear in these parameters

for *t = 1,2,...,T* **do**

Observe the context data c_t, which could be the user's personal, location, time data, etc.

Based on the past history $\{(c_1, a_1, r_{1,a_1}), ..., (c_{t-1}, a_{t-1}, r_{t-1,a_{t-1}})\}$, and the current context data c_t, the algorithm finds an optimal arm a_t, which is recommended to the user

The user gives back the reward r_{t,a_t} to the algorithm by executing the action a_t

The algorithm updates the history and the parameter of the model.

end

It is important to note that the algorithm is run *locally* on the device, not on some external server for computation. This is because in many cases it is not allowed to export users' personal data outside their own devices. For example, Apple confines the use of personal data within one's local device or personal cloud.[2].

4.3 Problem Settings and Implementations in the Real-World

We have seen a possible protocol of how to run our Bandit algorithm in users' local devices. However, actually implementing algorithm for a development project requires much more details for consideration. Suppose we are building a mobile application program for health and fitness. How do we implement a stochastic linear Bandit algorithm with contexts in order to provide a user a better choice for one's health improvement?

[2] `https://developer.apple.com/documentation/healthkit/protecting_user_privacy`

The followings are such considerations required when we face this kind of real-world problems.

Contents Our project must start from this. We need to define what our goal of creating this application is, how we can provide benefits to our users, what exact services we can provide to them, and so on. These considerations are the core part of the project for its blueprint. The goal should be narrowed down into specific health problems to be dealt with. For instance, rather than a mere health improvement, it could be like desirable fluid intake, medicine intake, avoiding sedentary behavior, or mental health care. The actual action made by a user for this goal may be walking, running, or other types of exercises. A typical one is walking, and there are active researches on this specific action(like the HeartSteps data([4])). To measure how much a user walk, we could simply measure the total count of one's steps in a day, or only consider the maximum stepcounts per hour in a day.

Data As we already have seen in the protocol, context data could be a user's personal data such as one's BMI, age, sex, race, or heartrate. Also, current weather, time, location could affect the user's behavior([4]). According to what the contents we have decided at the beginning, required context data could be changed. However, since these types of data are private, we need to obtain a consent from the user prior to running the algorithm. If a type of data is expected not to obtain user's consent easily, we may have to consider omitting that type. On top of that, there are such types that could not be measured physically by a typical mobile device, like blood pressure or body fat percentage. We should be aware of what types of data are *practically* available for our algorithm, which would be measured by devices of our *targeted* user. Suppose our service is for seniors, who are not expected to have devices of top-notch technology like

watch-like mobile devices. Then it would be nonsense to utilize a user's heartrate data, which would not be captured by one's typical mobile devices.

The points made in the above paragraph can be applied to rewards of algorithm as well. If we have set rewards for the algorithm like swimming hour, it would be fairly difficult for an ordinary mobile device user to get this type of reward, unless one directly input one's own swimming hour by oneself. But this way of gathering one's rewards would be unstable, since it could be that one could not precisely remember his exact exercise hour or might forget to input the hour. Even though the contents of our service themselves are important, to make it realizable those practical issues must be taken into account.

Implementation of Algorithm When it comes to the algorithm part, things get more complicated to be considered. This part is a step which specify the contents developed at the beginning into details, in order to create real-world services. There are at least four points to be discussed: hyperparameters, arms, feature vectors, and rounds. First, setting hyperparameters such as α in LinUCB or Linear TS depends on how much regret bound with desirable probability we want from the algorithm, as the theoretical proofs of the algorithm show(such as [1], [12]). However, if necessary, it could be arbitrary chosen, after observing some experiments with real data. Next, setting arms could be another big consideration. In our text, we assume them to be finite. However, if the form of the arms is close to a continuous variable, we may need to discretize. For example, in the case of recommending walking, we may set up some finite intervals of stepcounts and recommend those intervals. On the other hand, we may designate a few *special* arms, as Greenwald et al., 2017([5]) points out. In the stepcount case, we may suggest an action where the user do nothing. For the matter of feature vectors

$b(c_t, i)$, we may simply set them all equal to c_t, regardless of arms. However, we may take another approach to set them differently for each of the arms. Specific domain knowledge could help this problem. Finally, designing rounds is also a matter of concern. Since a mobile device does not have much big data storage capability, we cannot have T that big enough. Of course, the rounds that are too far from the current round would be relatively irrelevant, so they could be deleted(We do not have to take this into consideration if we use LinUCB or Linear TS, since we keep updating the parameters, which present the past history). Setting the rounds in terms of recommendation time points is also important. This is what we call the matter of *JITAI*(Just-In-Time Adaptive Interventions)([9]). We may set the recommendation time points as fixed, such as the algorithm runs at 7:00am everyday. Conversely, we could set them dynamically, which means we push alarms for the user when one's own action is most desirable for oneself. This is the ideal one, but then we may include these recommendation time points as another context variables, which could burden the computation of the algorithm.

4.4 Further Considerations

If we have thoroughly checked the points discussed in the previous section, now we are ready to develop a practical service. However, for the sake of the quality of our service, we may need to think of the following issues additionally.

How to Set Initial Values? First, it would be better for the performance of the algorithm to set its parameters' initial values from the start. In the case of Linear TS or LinUCB, we initialized the parameters as simple as zero vectors or identity matrices. However, for the matter of convergence, proper initial values might bring in better results. But since we are utilizing a person's private information, we are not allowed to accumulate several individuals' data in a central server. Therefore, we could purchase

sample personal data for the initial value setting, or may consult the domain experts' advice.

Missing Data Problems One of the more serious issues is the missing data problem. Sometimes a device could malfunction, so that a part of context data or reward data may be missing. On the other hand, a user may change one's mind and decide not to provide one's own health data to our service. If such events happen, we may stop the algorithm running at a round until the missing data problem is resolved. However, we may design more flexible and robust algorithms which also utilize such missing data for better performance. This could be another important research topic for researchers and engineers.

5 Conclusion

Throughout three chapters, we reviewed stochastic contextual Bandit models' concepts, basic examples of linear models such as LinUCB and Linear TS, and discussed several points required in their practical applications to mobile health problems. We anticipate further developments in hardware parts of mobile devices in the near future, so that much more various and important types of data could be obtained from users. For example, a sophisticated 3D computer vision technology could reconstruct one's own body pointcloud, so that a user can conveniently manage one's own bodyshape visually. Following such technological trends, we may propose more complicated contextual Bandit algorithms rather than mere stochastic linear models introduced in this text, and develop more various contents as services.

References

[1] Agrawal, S. and Goyal, N. "Thompson Sampling For Contextual Bandits With Linear Payoffs". In: *30th International Conference on Machine Learning (ICML)* 28 (2013), pp. 127–135.

[2] Auer, P., Cesa-Bianchi, N., and Fischer, P. "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning 47* (2002), pp. 235–256.

[3] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. "Gambling in a rigged casino: The adversarial multi-armed bandit problem". In: *Proceedings of IEEE 36th Annual Foundations of Computer Science* (1995), pp. 322–331. DOI: 10.1109/SFCS.1995.492488.

[4] Dempsey, W., Liao, P., Klasnja, P., Nahum-Shani, I., Murphy, S. A. "Randomised trials for the Fitbit generation". In: *Significance(Oxford, England) 12(6)* (2015), pp. 20–23.

[5] Greenewald, K., Tewari, A., Murphy, S., and Klasnja, P. "Action Centered Contextual Bandits". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5977–5985.

[6] Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2020. DOI: 10.1017/9781108571401.

[7] Li, L., Chu, W., Langford, J. and Schapire, R. E. "A Contextual-Bandit Approach to Personalized News Article Recommendation". In: *Proceedings of the 19th International Conference on World Wide Web* (2010), pp. 661–670. DOI: 10.1145/1772690.1772758.

[8] Slivkins, A. "Introduction to Multi-Armed Bandits". In: *arXiv e-prints* (2019). arXiv: 1904.07272 [cs.LG].

[9] Tewari, A. and Murphy, S. "From Ads to Interventions: Contextual Bandits in Mobile Health". In: *Mobile Health: Sensors, Analytic Methods, and Applications* (2017), pp. 495–517. DOI: 10.1007/978-3-319-51394-2_25.

[10] Thompson, W. R. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". In: *Biometrika, 25(3/4)* (1933), pp. 285–294.

[11] Tomkins, S., Liao, P., Klasnja, P. and Murphy, S. "IntelligentPooling: Practical Thompson Sampling for mHealth". In: *arXiv e-prints* (2020). arXiv: 2008.01571 [cs.LG].

[12] Wei Chu, W. and Lihong Li, L., Reyzin, L., and Schapire, R. "Contextual Bandits with Linear Payoff Functions". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* 15 (2011), pp. 208–214.

국문초록

맥락적 밴딧 모형은 밴딧 게임에서 참여자의 정보를 이용하여 효용을 최적화하려 한다. 응용의 관점에서 본다면, 확률적 선형 모형을 맥락적 밴딧 문제에 적용하는 것이 선호된다. 때문에 확률 맥락적 선형 밴딧 모형은 뉴스 기사 추천 알고리즘과 같은 실생활의 문제에서 널리 활용되고 있다. 최근, 모바일 건강 문제가 맥락적 밴딧 모형의 또다른 활용 분야로 제시되고 있다. 하지만 현실 세계에서의 응용을 위해서는, 모바일 기기에서 작동하는 알고리즘의 구현에 필요한 것들을 여러 관점에서 고려하는 것이 필수적이다. 본 논문에서는 확률 맥락적 선형 밴딧 알고리즘을 간단히 복습하고, 이후에 모바일 건강 문제에의 적용에 대하여 논의한다. 특히 모바일 건강 어플리케이션을 개발할 때 고려해야할 측면들에 대하여 상세히 고찰한다.

주요어: 맥락적 다중선택 밴딧 모형, 강화학습, 모바일 건강 문제
학 번: 2019-29758