Optimizing mHealth Interventions with a Bandit (Preprint)

Mashfiqui Rabbi, Ph.D, Department of Statistics, Harvard University
Predrag Klasnja, Ph.D., School of Information, University of Michigan, Ann Arbor
Tanzeem Choudhury, Ph.D., Department of Information Science, Cornell University
Ambuj Tewari, Ph.D., Department of Statistics, University of Michigan, Ann Arbor
Susan Murphy, Ph.D., Department of Statistics and Department of Computer Science, Harvard University

Corresponding author

Mashfiqui Rabbi, Ph.D.
Harvard University
Department of Statistics
1 Oxford St #316,
Cambridge, MA, 02138
United States
Phone no.: 1-603-667-1797
Email: mrabbi@fas.harvard.edu

Abstract

Mobile health (mHealth) interventions can enable new ways to improve health outcomes by intervening in the moment of need or in the right life circumstance. With recent advances in mobile computing and sensing techniques, mHealth interventions are now technologically feasible; current off-the-shelf mobile phones can acquire and process data in real time to deliver relevant interventions in the moment. Learning which intervention to provide in the moment, however, is an optimization problem. This book chapter describes one algorithmic approach, a "bandit algorithm," to optimize mHealth interventions. Bandit algorithms are well-studied and are commonly used in online recommendations (e.g., Google's ad placement, or news recommendations). Below, we walk through simulated and real-world examples to demonstrate how bandit algorithms can be used to personalize and contextualize mHealth interventions. We conclude by discussing challenges in developing bandit-based mobile health interventions.

1. Introduction

Before mHealth, the standard of care was periodic visits to a clinician's office, interspersed with little to no patient support in between visits. At the clinician's office, data is collected to describe the patient's state at that visit time and self-report data about the patient's state prior to the current visit time is collected through an error-prone mechanism of recalling past events. The mHealth model has enabled significant progress in-situ data collection between clinic visits; phone sensors can now capture personal data at a millisecond level, and improvement in user interfaces has reduced the burden of self-report information [18]. mHealth interventions using persuasive design features are promising approaches for improving patients health [19,20].

However providing effective interventions personalized to the patient between patient visits remains challenging.

Two key components of intervening at the right time are personalization and contextualization. Personalization is the process of matching an individual's preferences and lifestyle. e.g., a physical activity intervention can say, "You walked 10 times in the last week near your office. Don't forget to take small walks near your office today." Such personalization can lower barriers to acting on the suggestion [2]. Contextualization takes personalization one step further by delivering interventions at moments of need or at an opportune moment when the intervention is easy to follow [1]. e.g., when a participant reaches the office, a push notification with the earlier walking suggestion can be sent, or, just after a high risk teen reports high stress, a SMS can be sent with ideas to reduce stress.

Contextualization and personalization, are complex problems because different people may prefer different interventions and these preferences can vary by context. Fortunately, similar problems have been solved before. When Google places ads or Netflix suggests movies, they adapt their recommendation based on user preferences and characteristics, utilizing bandit algorithms. Here we describe how bandit algorithms can be repurposed to personalize and contextualize mHealth interventions. We will start with a simple example, where we personalize a daily list of physical activity suggestions to an individual. We will then extend this simple example to account for contextual factors (e.g., weather). We conclude with a real-world example and discuss future challenges in developing personalized/contextualized interventions with bandit algorithms.

2. Background

Bandit algorithms: "Bandit algorithms" are so called because they were first devised for the situation of a gambler playing one-armed bandits (slot machines with a long arm on the side instead of a push button). Each time the gambler picks a slot machine, he/she receives a reward. The bandit problem is to learn how to best sequentially select slot machines so as to maximize total rewards. The fundamental issue of bandit problems is the *exploitation-exploration* tradeoff; here exploitation means re-using highly rewarding slot machines from the past and exploration means trying new or less-used slot machines to gather more information. While exploration may yield less short-term payoff, an exploitation-only approach may miss a highly rewarding slot machine. Researchers have proposed solutions to the bandit's exploit-explore tradeoff across many areas. In particular, once the relevance of bandit algorithms to internet advertising was understood, there was a flurry of work [4]. Nowadays, bandit algorithms are theoretically well understood, and their benefits have been empirically demonstrated [4,5].

An important class of bandit problems is the contextual bandit problem that considers additional contextual information in selecting the slot machine [6]. Contextual bandit problems provide a natural model for developing mobile health interventions. In this model the context is the information about the individual's current circumstances, the slot machines correspond to the different intervention options, and the rewards are near-time, proximal, outcomes [3]. In this setup, optimizing mHealth intervention delivery is the act of learning the intervention option that will result in the best proximal outcome in a given circumstance. This is same as solving the contextual bandit problem.

3. Optimizing intervention with a Bandit algorithm

We will use two examples to explain how bandits can be used to optimize an mHealth intervention for an individual. In section 4, we will discuss another real-world mobile application that builds on the ideas introduced in the first two simple examples.

In our first example, the bandit algorithm will be used to select an optimal set of five physical activity suggestions, for an individual, from a set of ten suggestions. A set of five suggestions is optimal if the set leads to the highest level of daily activity for that individual. The second example extends the first by finding a set of five suggestions for each of several contexts. Contextualizing suggestions can be helpful because the same suggestion may be more actionable in certain contexts (e.g., good weather or day of the week).

---

1. Walk 30 minutes
2. Add intervals: walk 5 minutes, walk very fast for 5 minutes, repeat 3 times
3. Take the stairs instead of the elevator whenever possible
4. Go for a walk with a friend or your dog
5. Swim a lap, rest 1 minute, repeat 10 times
6. Attend a fitness class at your gym

---

7. Try some of the strength training and bodyweight exercises illustrated by the fitness app on your phone
8. Do yoga
9. Park at the far end of the parking lot to walk farther
10. Do yardwork for at least 10 minutes

Table 1: List of 10 suggestions

3.1 Personalizing suggestions for an individual

Consider a scenario in which Jane's health plan gives her a physical activity tracker and a smartphone app. Jane’s health plan has found that the ten activity suggestions from Table 1 often work for many less-active people to increase their activity. Note that the order of suggestions in Table 1 does not imply any specific ranking. It is unlikely, however, that every individual will be able to follow or prefer to follow all the 10 suggestions equally and there will be inter-personal variability in which suggestions are followed and to what degree. Thus, we set the goal of learning the five suggestions with the highest chance of maximizing Jane’s activity. We use the bandit algorithm, which is running as part of Jane’s smartphone app, to achieve this goal. Each morning, the app issues a set of 5 suggestions. The app then monitors Jane’s activities throughout the day and uses that information to choose 5 suggestions for the following day.

Formally, we will refer to each set of five activity suggestions as an *intervention option* or *action.* This intervention option or action is the particular choice of the five suggestions. On the morning of day t, the app suggests to Jane the action A_t , where $A_t = [S_{t1}, S_{t2}, S_{t3}, \dots, S_{t10}]^T$ is a 10×1 vector of binary variables. S_{ti} has a value of 1 if the i-th suggestion from Table 1 is shown to Jane on day t, and 0 otherwise. Thus A_t will have 5 entries equal to 1 and 5 entries equal to 0. Further, let Y_t denote the number of active minutes for Jane on day t, which might be called the *proximal outcome* or *reward* of action A_t.

Consider the following linear regression model for the mean of the daily active minutes Y_t on day t in terms of the suggestions:

$$\begin{aligned} \boldsymbol{E}[\boldsymbol{Y_t}|\boldsymbol{A_t}] &= \sum_{i=1}^{10} \boldsymbol{\beta_i S_{ti}} \\ &= \boldsymbol{\beta^T A_t} \end{aligned} \tag{1}$$

where the second equality is written more compactly by using vector notation, $\beta = [\beta_1, \beta_2, \dots, \beta_{10}]^T$. Here $\beta_1, \beta_2, \beta_3, \dots, \beta_{10}$ respectively represent suggestion $1, 2,3, \dots ,10$s contribution to Jane's number of active minutes. Therefore, Equation 1 has the following simple interpretation: Y_t, the number of daily active minutes, is the sum of the effects of the 5 activity suggestions provided on day t (i.e., suggestions for which $S_{ti} = 1$).

Formally, our goal is to discover the best action $A_t = a^*$ that is, the set of 5 suggestions that makes Jane most active (that results in the highest mean daily active minutes). We can formally write this goal as: given β, determine the action a^* for which

$$\boldsymbol{\beta^T a^*} \geq \boldsymbol{\beta^T a} \tag{2}$$

where a is a combination of 5 suggestions from Table 1. β is, however, unknown. We can estimate Jane's a^* by running experiments in the following way: at the start of a day t, the app selects action A_t (in other words, it delivers to Jane a combination of 5 suggestions from Table 1). The tracker then counts the number of minutes Jane is active on the day (note that this number is the proximal outcome Y_t). If the 5 suggestions are useful, then Jane will be more active that day and Y_t will be high compared to other days with a different set of 5 suggestions. Now, the question is: how to select the 5 suggestions each day? One simple approach is to select 5 suggestions out of 10 with equal probability. But such a uniform selection strategy will select more useful and less useful suggestions equally. A more sophisticated approach is to use the information already available from the past experiments to select future suggestions that will both yield additional information about a^* and give as few less useful suggestions as possible. Note that here we face the same exploit-explore tradeoff faced by the classic bandit setting's gambler – i.e., how to balance exploiting suggestions that seemed useful in the past with exploring less frequently issued suggestions.

An effective approach to delivering less useful suggestions as little as possible is "optimism in the face of uncertainty" epitomized by the Upper Confidence Bound (UCB) technique [7,8]. Bandit algorithms based on the UCB have been well studied and possess guarantees of minimizing the number of less useful suggestions. The key intuition behind the UCB idea is the following: First, for each choice of action a_t, a confidence interval is constructed for the linear combination $\beta^T a_t$. Recall this linear combination represents $E[Y_t|A_t = a_t]$, the expected proximal outcome after receiving action, a_t. Then the UCB bandit algorithm selects the action with the highest upper confidence limit. Note that the upper confidence limit for $\beta^T a_t$ can be high for either of two reasons: (1) either $\beta^T a_t$ is large and thus a_t is a good action to make Jane active, or (2) the confidence interval is very wide with a high upper limit, indicating that there is much uncertainty about the value of $\beta^T a_t$. Using the upper confidence limit represents UCB's optimism; UCB is optimistic that actions with high upper confidence limits will be the best actions, even though a larger upper confidence limit can mean more uncertainty. However, if an action with high upper confidence is indeed not the optimal action, then selecting the action will reduce the uncertainty about the effect of this action. This will help UCB realize that the action is indeed not useful.

How does UCB choose an action using the upper confidence interval? By following these two steps. The first step involves using Equation 1 to estimate β assuming homogeneous error variance. We might use ridge regression to estimate β because ridge regression regularizes to avoid overfitting, especially when Jane has just begun to use the app and we have less data [8,9]. In this case the estimator of β, denoted by $\widehat{\beta_t}$, after t days of using the bandit algorithm is:

$$\widehat{\boldsymbol{\beta}_t} = \widehat{\boldsymbol{\Sigma}}_t^{-1}\left(\sum_{u=1}^{t} \boldsymbol{A}_u \boldsymbol{Y}_u\right) \tag{3}$$

where $\hat{\Sigma}_t^{-1} = \sum_{u=1}^{t}(A_u A_u^T) + I_{10}$ and I_{10} is an 10×10 identity matrix. Equation 3 is the standard solution for ridge regression. The second step is to construct an upper confidence limit for $\beta^T a$ for each possible action a; the upper confidence limit on day t for action a is given by $\hat{\beta}_t^T a + \alpha\sqrt{a^T \hat{\Sigma}_t^{-1} a}$, where α is an appropriate critical value. Note, since we assumed homogeneous error variance, $\hat{\Sigma}_t^{-1}$ is proportional to the covariance for $\widehat{\beta_t}$, and $a^T \hat{\Sigma}_t^{-1} a$ is the covariance of $\beta^T a$. Thus, $\sqrt{a^T \hat{\Sigma}_t^{-1} a}$ represents standard deviation of $\beta^T a$ and the upper confidence limit of $\beta^T a$ has an interpretable form, which is simply the current estimate, $\hat{\beta}_t^T a$, plus its standard deviation multiplied up to a constant factor α. Then, to choose the UCB action for day $t + 1$, we calculate the a_{t+1} for which

$$\hat{\boldsymbol{\beta}}_t^T \boldsymbol{a}_{t+1} + \boldsymbol{\alpha}\sqrt{\boldsymbol{a}_{t+1}^T \hat{\boldsymbol{\Sigma}}_t^{-1} \boldsymbol{a}_{t+1}} \;\geq\; \hat{\boldsymbol{\beta}}_t^T \boldsymbol{a} + \boldsymbol{\alpha}\sqrt{\boldsymbol{a}^T \hat{\boldsymbol{\Sigma}}_t^{-1} \boldsymbol{a}} \tag{4}$$

for all actions a. i.e., a_{t+1} is selected to maximize the upper confidence limit on the mean of Y_{t+1}. This approach possesses strong guarantees to minimize the number of less useful suggestions [8,17].

Here we summarize how the UCB bandit algorithm works on Jane's smartphone. First there is an "exploration phase" to allow the UCB algorithm to form preliminary estimates of β. This phase lasts for a number of days, say t_0 days, during which each morning the UCB bandit algorithm randomly selects an action, that is, uniformly selects five activity suggestions from the 10, and delivers these suggestions to Jane in the application. Then at the end of day t_0, the UCB bandit uses an incremental calculation to form $\hat{\beta}_{t_0}$ and $\hat{\Sigma}_{t_0}$ based on the selected action, Jane's activity minutes, Y_{t_0}, for that day and the prior day's $\hat{\beta}_{t_0-1}$ and $\hat{\Sigma}_{t_0-1}$. Next the UCB algorithm calculates the upper confidence limit for each action and selects the action a_{t_0+1} with the highest upper confidence limit. On the next morning, Jane is provided the five suggestions as specified by a_{t_0+1}. The UCB algorithm repeats the process by estimating new $\hat{\beta}_{t_0+1}$ $\hat{\Sigma}_{t_0+1}$ and an updated set of 5 suggestions are chosen for the next day and so on.

3.1.1 A simulation example

In this section, we use a simulated example to demonstrate how a UCB bandit algorithm can personalize suggestions for Jane. We assume the following simple model of how Jane responds to the suggestions: When Jane sees a suggestion, she follows it with probability p or does not follow it with probability $1-p$. If Jane follows the suggestion, she spends D minutes following it on a particular day. We assume D is random and normally distributed, because Jane may not spend the same amount of time each time she follows the same suggestion. In Table 2, we created an artificial example scenario with p and D values for different suggestions. The D values are written as mean $\pm$ standard deviation. We also show the expected number of activity minutes that Jane spends following a suggestion when she sees it. This expected number is $p \times E[D] + (1-p) \times 0 = pE[D]$. These expected minutes are also β values in equation 1. Note that β values are unknown in real world setting. We use known β values in a simulated example to show how the UCB algorithm finds the suggestions with higher β values.

Suggestions	$\boldsymbol{p}$	**Duration, *D* (in minutes)**	**Expected duration** $pE[D]$ **(in minutes)**
1. Walk 30 minutes	1	15 ± 4	15.0
2. Add intervals: walk 5 minutes, walk very fast 5 minutes, repeat 3 times	$\frac{1}{90}$	21 ± 5	0.4
3. Take the stairs instead of the elevator whenever possible	$\frac{5}{7}$	7.5 ± 2	5.2
4. Go with a friend or your dog for a walk	$\frac{6}{7}$	22 ± 10	18.9
5. Swim a lap, rest for 1 minute, repeat 10 times	0	–	–
6.Attend a fitness class at your gym	$\frac{1}{14}$	31 ± 5	2.2
7. Try some of the strength training and bodyweight exercises illustrated by the fitness app on your phone	0	–	–

8. Yoga	$\frac{4}{7}$	18 ± 3	10.3
9. Park at the far end of the parking lot to walk further	$\frac{4}{7}$	11 ± 2	6.3
10. Do yardwork for at least 10 minutes	$\frac{3}{14}$	24 ± 5	5.1

Table 2: A simulated scenario for Jane where p represents the probability of following a suggestion when Jane sees it, and if the suggestion is followed, "Duration" represents the number of daily minutes spent following the suggestion. Finally, p and "Duration" are used to compute the expected value $pE[D]$, which also represents β values for the suggestion.

With the above setup, we run the simulation in two stages. In the first stage, suggestions are included with equal probability in the five suggestions on each of the first fourteen days. This initial "exploration phase" helps to form an initial estimate of β. In the second stage, we run the UCB bandit algorithm: on each day, we compute $\widehat{\beta_t}$, according to equation 3, and choose an action using Equation 4. We run these simulation for 56 days, or 8 weeks. We run 200 instances of the simulation to account for randomness in the problem. One source of this randomness comes from the exploration phase, where the app generates non-identical sequences of random suggestions based on when Jane starts using the app. We deal with this randomness by resetting the randomization seed after each simulation run. Another source of randomness comes from the within-person variability of how Jane responds to the suggestions. We create a second stream of random numbers to simulate how Jane responds to the suggestions. The seed of this second stream remains unchanged after each simulation run; we do not reset this seed because, doing so will add the randomness of resetting the seeds to the within-person variability.

Table 3 shows the results, where we report the mean of the β estimates. At the top, we list the actual β values. We then list in each row how many times a suggestion is issued by UCB over a two week period. We use boldface for the top five suggestions (1st, 3rd, 4th, 8th, 9th in Table 1). The simulation shows that after the two-week exploration phase, UCB chooses the top (boldfaced) suggestions more times than the less useful ones. Since a suggestion can be picked only once a day, the top suggestions 1, 2, and 8 from Table 3 are picked nearly every day after the exploration phase (11-14 days between week 3-4, 5-6, and 7-8). However, suggestions 3, 9, and 10 all have similar β values. As a result, UCB is often uncertain among them and chooses the 10th suggestion sometimes wrongly, since it is not in the top five suggestions.

Suggestions	**1**	2	**3**	**4**	5	6	7	**8**	**9**	10
β	**15.0**	0.4	**5.2**	**18.9**	0.0	2.2	0.0	**10.3**	**6.3**	5.1
$\widehat{N}$ (week 1-2)	**7.1**	7.2	**7.0**	**7.0**	6.8	6.9	6.9	**6.8**	**7.1**	7.1
$\widehat{N}$ (week 3-4)	**12.4**	3.9	**6.3**	**13.4**	2.8	4.5	2.5	**9.6**	**7.8**	6.5
$\widehat{N}$ (week 5-6)	**12.8**	3.5	**6.3**	**13.7**	2.6	4.3	1.7	**10.1**	**8.1**	6.7
$\widehat{N}$ (week 7-8)	**13.1**	3.4	**6.4**	**13.8**	2.4	4.3	1.6	**10.1**	**7.8**	6.8

Table 3: Number of times suggestions are picked by the app within each of the two-week intervals. $\widehat{N}$ denotes the number of days the app selects a suggestion in the time frame mentioned within parenthesis. Note, the number of times a suggestion can be selected during a two-week period is at most 14 (i.e., $\widehat{N} \leq 14$).

3.2. Optimizing interventions for different contexts

In the earlier section, we discussed an example of personalizing suggestions with the UCB algorithm. Our goal was to demonstrate the inner workings of a bandit algorithm in a simple setting. Here we discuss extending the prior example to a more realistic setting where we tailor suggestion based on users' context. Indeed, context can determine whether, and the degree to which, certain suggestions are actionable. For example, Jane may only be able to act on the yardwork suggestion on the weekend, or she may appreciate and act on the reminder to take her dog for a walk when the weather is good. By adapting suggestions to different contexts, we hope to enhance her activity level. Fortunately, we can contextualize suggestions by re-purposing the bandit technique described already. We briefly describe one way to do so below.

	Context
1	Bad weather, weekend
2	Bad weather, weekday
3	Good weather, weekend
4	Good weather, weekday

Table 4: Different types of contexts

For clarity, we will first consider a very simple context involving only the weather and day of the week. For these two contexts, there are two states (i) weekend or weekday, (ii) good or bad weather, where we consider the whole day as bad weather if only part is. Thus, each day belongs to one of four different context combinations (see Table 4). Note this simple characterization of only 4 contexts is to convey the idea of contextualization rather than actually to realistically handle a large number of contexts.

For these four context combinations, the task of contextualizing suggestions boils down to optimizing the suggestions for each of the four. An intuitive approach is to use 4 different bandit algorithms, one for each context combination. Depending on the context on day t, the corresponding bandit would be activated for optimizing suggestions for that context. Recall that an action is a set of five activity suggestions from the 10 in Table 1. Each of the four different bandit algorithms uses a model such as Equation 1 but with different βs due to the different contexts. We represent this difference by sub-scripting β as β_k for the k-th ($k = 1,2,3,4$) context. So, the goal is to learn the optimal action a_k^* that maximizes the average number of minutes active for Jane in context k. That is, for $k = 1,2,3,4$ the goal is to learn the action a_k^* which satisfies

$$\beta_k^T a_k^* \geq \beta_k^T a_k$$

Again, one UCB bandit algorithm can be run per context to learn the optimal five suggestions for that context.

Note that using a separate bandit algorithm for each context is not a feasible approach in a real-world setting; there are too many possible contexts. It would take the bandit algorithm many days to obtain good estimates of the β_k parameters. However, we can use a few tricks to handle large number of contexts. First, we may know a priori that some suggestions are equally actionable across different contexts and some suggestions are not at all actionable in certain contexts. If the suggestions are equally actionable across contexts, we can use the same

β_k parameter values for these contexts. And if a suggestion is not actionable in a given context we can set its parameter in β_k to zero. Second, we can pool information across people. For example, some suggestions, such as yardwork, are more actionable on weekends for most people. Thus, we don't need to find β_k for each user individually. Pooling information, however, requires a Bayesian approach where for a new user, initially β_k is pooled from prior users and once some data from the user is available, β_k is then adapted to make more user-specific changes. Bayesian approaches to bandit algorithms are beyond the scope of this chapter; but the techniques are along the same lines as UCB [10].

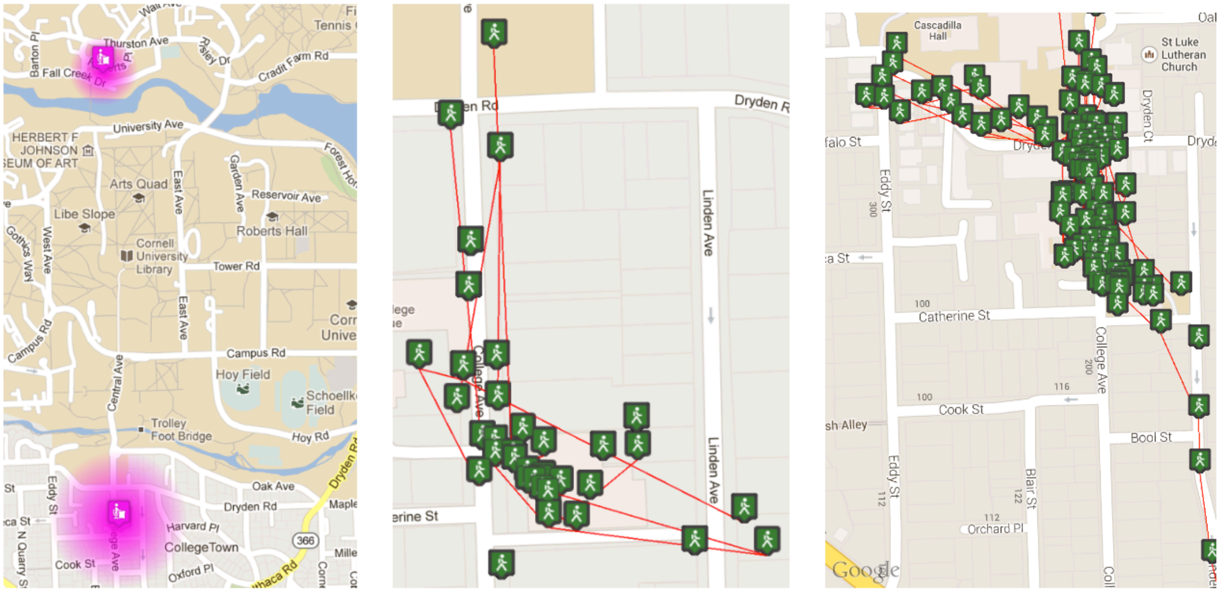
4. A real-world example

Earlier, we gave two simple examples of how the UCB bandit algorithm can personalize and contextualize mobile health interventions. Real-world examples, however, are more complicated, with many potential suggestions and many contexts. Below we discuss an mHealth app called MyBehavior that has been deployed multiple times in real world studies [11,12]. MyBehavior utilizes phone sensor data to design unique suggestions for an individual and subsequently uses a bandit algorithm to find the activity suggestions that maximize chances of daily calorie burns. Like the example in Section 3, MyBehavior issues the suggestions once each morning. The number of suggestions, however, is higher than in Table 1 because the suggestions in MyBehavior closely match an individual's routine behaviors, and routine behaviors are dynamic. In the following, we briefly discuss how MyBehavior uses the bandit algorithm. More information on this can be found in [13].

4.1 MyBehavior: Optimizing individualized suggestions to promote more physical activity

The following discussion of MyBehavior first covers how unique suggestions are created for each individual. We then briefly discuss how a bandit algorithm is used to find optimal activity suggestions that have the highest chance of maximizing an individual's daily calorie burn.

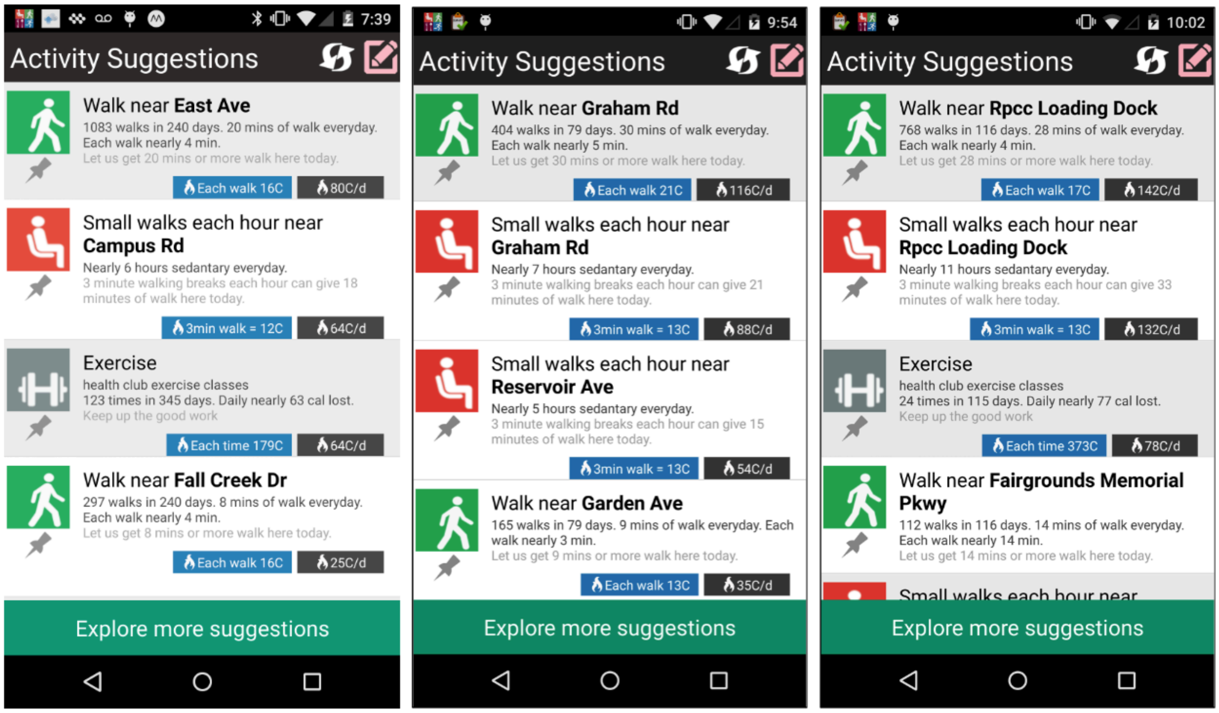
The MyBehavior app tracks an individual's physical activity and location every minute. The detected physical activities include walking, running, driving, and being stationary. The app then analyzes the location-tagged activity data to find patterns that are representative of the user's behaviors. Figure 1 shows several examples of behaviors found by MyBehavior. Figure 1a and Figure 1b respectively contain places where a user stayed stationary and a location where the user frequently walked. Figure 1c shows similar walking behaviors from another user. MyBehavior uses these behavioral patterns to generate suggestions that are unique to each individual. For example, one intervention may suggest an activity goal at specific locations that the user regularly goes to. Such tailoring makes feedback more compelling, since a user's familiarity with the location enhances adherence [1].



(a) (b) (c)

Figure 1: Visualization of a user's movements over a week (a) Heatmap showing the locations where the user is stationary everyday (b) Location traces of frequent walks for the user (c) Location traces of frequent walks for another user

Specifically, MyBehavior creates three kinds of uniquely individualized suggestions: (i) for stationary behaviors, MyBehavior pinpoints the locations where the user tends to be stationary and suggests taking small walking breaks every hour in these locations. (ii) for walking behaviors, MyBehavior locates the different places the user usually walks and suggests continuing to walk in those locations (iii) for other behaviors, e.g., participation in yoga class or gym exercises, MyBehavior simply reminds the user to keep up the good work. Figure 2 shows several screen shots of the MyBehavior app, where Figures 2a-c are suggestions for three separate users. Since MyBehavior suggestions are tailored to the user, the first suggestion at the top of each screen shot is to walk, but the locations are different. Also, the first and third users receive a gym weight training exercise suggestion that the second user does not.



(a) (b) (c)

Figure 2: MyBehavior app screenshots for three different users[1]

Now, how does MyBehavior decide which suggestions to give? MyBehavior uses a bandit algorithm like that in Section 3's first example, where suggestions are issued once a day. But MyBehavior can offer many more suggestions than Table 1 contains, depending on the variety of locations in which a user might be sedentary or active, etc. Fortunately, the bandit algorithm can still efficiently adapt to these high numbers of tailored suggestions. Rabbi et al [13] details how this optimization works, but the key intuitions are the following: (i) Most human behaviors are highly repetitive and routine and occur in the same locations. Routine behaviors and locations will be detected early and thus included soon in the individual's list of suggestions. (ii) The suggestions relating to routine behaviors and locations are more likely to be followed than

[1] Figure 1 and 2 have been reproduced from Rabbi et al. [12] with appropriate permission from the authors.

suggestions of non-routine behaviors in non-routine locations. Thus, the bandit will learn about the effects of these suggestions more quickly and these suggestions will likely remain effective if the user's routine does not change.

5. Discussion

In the last two sections, we discussed several examples of how bandit algorithms can optimize mobile health interventions. The bandit algorithm balances experimenting with different activity suggestions and selecting activity suggestions that currently appear most useful. This balancing act ensures that the algorithm acquires necessary information while maintaining an engaging user experience by providing as few less-useful suggestions as possible. While we showed that bandit algorithms can be useful to personalize and contextualize suggestions, there are additional complexities in real-world mHeath intervention settings that generate new challenges for bandit algorithms to address:

Ignoring delayed effects: In bandit algorithms, the optimal action is the action that maximizes the immediate reward (proximal outcome). In other words, bandit algorithms ignore the potential impact of the action on future context and future proximal outcomes. Some actions, however, can have long-term negative effects even if the short-term effect is positive. e.g., delivering an office walking suggestion may increase a user's current activity level, but the user might become bored after repeating the office walk several days, thus future suggestions may be less effective. In these cases, other algorithms that explicitly allow past actions to impact future outcomes [14] might be used. Precisely, the outcome of these algorithms are $Y_t + V(X_{t+1})$, where $V(X_{t+1})$ is the prediction of the impact of the actions on future proximal outcomes given the context X_{t+1} at the time $t+1$ (a bandit algorithm acts as if $V(X_{t+1})$=0). These algorithms tend to learn more slowly than bandit algorithms, since we need additional data to form the prediction $V(X_{t+1})$. We conjecture that the noisier the data is, the harder it will be to form high quality predictions of $V(X_{t+1})$ and thus as a result, bandit algorithms may still be preferable.

Non-stationarity: Most bandit algorithms assume "stationary" settings; i.e., the responsivity of a user in a given context to an action does not change with time. This assumption can be violated in real-word settings; in MyBehavior, for example, we observed that many suggestions become ineffective when people switched job and moved from one location to another. Such changes over time are often referred to as "non-stationarity." Other types of non-stationarity can be caused by life events such as a significant other's illness or aging. Bandit algorithms are typically slow to adapt to non-stationarity. Speeding up this process is a critical direction for future bandit research.

Dealing with less data: In real world applications, where the number of contexts and actions are many, bandit algorithms will need a lot of burdensome experimentation to find the optimal action for a given context. One way around this is to use a "warm start." A warm start set of decision rules that link the context to the action can be constructed using data from micro-randomized trials [15] involving similar individuals. Recently Lei et al. [16] developed a bandit algorithm that can employ a warm start. However, we still need to test whether, and in which settings, warm starts will sufficiently speed up learning.

Adverse effects: Since mHealth interventions are generally behavioral, the risk of personal harm is often minimal. Nonetheless, there could be potential iatrogenic effect because phones cannot capture every piece of contextual information and bandit algorithms ignore the long-term effects of interventions. Since bandit algorithms don't take interventions' long-term effects into account, the algorithm may notify or otherwise deliver interventions too much and thus cause annoyance and reduce app engagement. Future work needs to investigate how to account for such long-term adverse effects. Furthermore, current phone sensors cannot automatically capture critical contextual information such as a user's health risks, preferences, barriers, emotional states, etc. Incomplete information may cause the algorithm to provide less appealing (e.g., not suggesting an activity that a user likes but didn't do often in the past) and inappropriate suggestions (e.g., asking someone who is injured to walk). Providing human control over the suggestion generation process can mitigate these problems; e.g., a user can delete inappropriate suggestions and prioritize the suggestions that are more appealing [12].

References:

1. Fogg, B. J. (2009). A behavior model for persuasive design. In Proceedings of the 4th international Conference on Persuasive Technology, ACM, 40.

2. Hochbaum, G., Rosenstock, I., & Kegels, S. (1952). Health belief model. United States Public Health Service.

3. Nahum-Shani, I., Smith, S. N., Spring, B. J., Collins, L. M., Witkiewitz, K., Tewari, A., & Murphy, S. A. (2017). Just-in-time adaptive interventions (JITAIs) in mobile health: key components and design principles for ongoing health behavior support. Annals of Behavioral Medicine, 52(6), 446-462.

4. Bubeck, S., & Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Foundations and Trends® in Machine Learning, 5(1), 1-122.

5. Chapelle, O., Joachims, T., Radlinski, F., & Yue, Y. (2012). Large-scale validation and analysis of interleaved search evaluation. ACM Transactions on Information Systems (TOIS), 30(1), 6.

6. Woodroofe, M. (1979). A one-armed bandit problem with a concomitant variable. Journal of the American Statistical Association, 74(368), 799-806.

7. Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2-3), 235-256.

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

9. Bishop, C. M. (2007). Pattern recognition and machine learning. Springer.

10. Chapelle, O., & Li, L. (2011). An empirical evaluation of thompson sampling. In Advances in neural information processing systems, 2249-2257.

11. Rabbi, M., Aung, M. S., Gay, G., Reid, M. C., & Choudhury, T. (2018). Feasibility and Acceptability of Mobile Phone–Based Auto-Personalized Physical Activity Recommendations for Chronic Pain Self-Management: Pilot Study on Adults. Journal of medical Internet research, 20(10), e10147.

12. Rabbi, M., Aung, M. H., Zhang, M., & Choudhury, T. (2015). MyBehavior: automatic personalized health feedback from user behaviors and preferences using smartphones. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 707-718.

13. Rabbi, M., Aung, M. H., & Choudhury, T. (2017). Towards health recommendation systems: an approach for providing automated personalized health feedback from mobile data. In Mobile Health. Springer, Cham., 519-542.

14. Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT press

15. Klasnja, P., Hekler, E. B., Shiffman, S., Boruvka, A., Almirall, D., Tewari, A., & Murphy, S. A. (2015). Microrandomized trials: An experimental design for developing just-in-time adaptive interventions. Health Psychology, 34(S), 1220.

16. Lei, H., Tewari, A., & Murphy, S. (2014). An actor-critic contextual bandit algorithm for personalized interventions using mobile devices. Advances in Neural Information Processing Systems, 27.

17. Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research, 3(Nov), 397-422.

18. Kubiak, T. & Smyth, J.M. (2019). Connecting domains – ecological momentary assessment in a mobile sensing framework. In H. Baumeister & C. Montag (eds.), Mobile Sensing and Digital Phenotyping: New Developments in Psychoinformatics (p. x-x). Springer: Berlin.

19. Baumeister, H., Kraft, R., Baumel, A., Pryss, R., & Messner, E.-M. (2019). Persuasive e-health design for behavior change. In H. Baumeister & C. Montag (eds.), Mobile Sensing and Digital Phenotyping: New Developments in Psychoinformatics (p. x-x). Springer: Berlin.

20. Messner, E.-M., Probst, T., O´Rourke, T., Baumeister, H. & Stoyanov, S. (2019). mHealth applications: potentials, limitations, current quality and future directions. In H. Baumeister & C. Montag (eds.), Mobile Sensing and Digital Phenotyping: New Developments in Psychoinformatics (p. x-x). Springer: Berlin.