Contents

1 Questions

What is the price of personalization? We work with HeartSteps v1 data in order to choose an online learning algorithm for future HeartSteps trials. This project aims to answer the question on whether it is possible to tailor just-in-time intervention messages to individual users in a way that does not jeopardize the results of other users.

We choose to answer the following questions (see Table 2.1 for notation):

1. How robust is the bandit algorithm to misspecification of linear model for the true generative model given context? How does this vary with signal to noise ratio (Equation 2.2)?

2. How fast does the bandit algorithm result in action selection probabilities equal to 0.2 or 0.8? How does this vary with SNR? How does this vary with the diagonal variance terms in Σ_Θ, the prior on Θ?

3. If there is no treatment effect (i.e. the generative model in the simulation for the reward does not depend on action $\mathcal{A}$, or that the true value of interaction terms of Θ are 0), then what doe the selection probabilities do? Does the choice of the prior mean, μ_Θ affect this? For example, for the interaction terms, if μ_Θ is not close to 0 but in reality true $\Theta = 0$.

4. How sensitive is the bandit algorithm to really good initialization of the Θ coefficients in the linear model? that is, when the prior on Θ has a mean μ_Θ that is close or equal to the true values of Θ in the true reward generative model. How sensitive is the bandit algorithm to really poor initialization of the coefficients in the linear model?

5. If the true underlying model for the conditional mean of the reward given context varies with time *t* in a way that leads to differing optimal actions from some commonly occurring contexts, then does the bandit algorithm adjust to this?

2 Methods

Throughout this project, we work with the HeartSteps v1 Data, hereupon abbreviated HSv1. Features have been created from the measurements through domain science, through which we assume that the Bandit algorithms use linear models for the purposes of the project.

The overall methodology is the following. We first designate a variant of the Bandit algorithm, as well as a 'true' generative model we use to form simulated rewards from the context and given action. Next, we split HSv1 user data into K-fold cross validation sets. For each split, perform the following:

1. Use the training and testing data to generate training simulated users and testing simulated users.
2. Use training simulated users to tune parameters of the given Bandit algorithm variation.
3. Run simulated users from test data, observe quality metrics.

Each part is described in more detail below.

2.1 Reward Generative Model

We set different generative models, where we assume:

$$\mathcal{R} = \begin{bmatrix} f_1(\mathcal{S}), & \mathcal{A} \odot f_2(\mathcal{S}) \end{bmatrix}^T \Theta + \varepsilon \tag{2.1}$$

and that f_1, f_2 are linear, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ for σ^2, which is a tuning parameter. Recall that the Signal-to-Noise-Ratio (SNR) is computed as

$$\frac{\mathrm{Var}\left(\begin{bmatrix} f_1(\mathcal{S}), & \mathcal{A} \odot f_2(\mathcal{S}) \end{bmatrix} \Theta\right)}{\sigma^2}. \tag{2.2}$$

Table 2.1: Notations

Term	Name	Description
$\mathcal{S}$	Context	Set of 7 features
p_1	Baseline features dimension	7 features with 1 bias term
p_2	Interaction features dimension	3 features
$\mathcal{A}$	Actions	Binary – active message sent, no active message sent
$\mathcal{R}$	Reward	Step count in 30 minutes following decision point
$f_1 : \mathcal{S} \to \mathbb{R}^{p_1}$	Baseline feature mapping	Maps context to baseline features
$f_2 : \mathcal{S} \to \mathbb{R}^{p_2}$	Interaction feature mapping	Maps context to interaction features, which are multiplied by $\mathcal{A}$
Θ	Linear model coefficients	From regression $\mathcal{R} \sim [f_1(\mathcal{S}) + f_2(\mathcal{A} \cdot \mathcal{S})]\Theta$
ε	Linear model residuals	Residuals from HSv1 data after regression
N	Number of users	$N = 48$ in HSv1; N varies for simulations
T	Number of days in study	$T = 41$ in HSv1; $T = 90$ in HSv2
D	Number of decision points per day	Set to $D = 5$
K	Number of cross-validations per test	Currently $K = 5$
B	Number of batches per simulation	Ideally $B = 1000$, worst case $B = 100$

We describe these models more in Section 3.1.

2.2 Cross-Validation

We use $K = 5$ fold cross-validation to separate training and test batches. We do not train the bandit algorithm's parameters on the test batches, as to simulate application of HSv1 data for HSv2.

To do this, we randomly order the $N = 48$ users, then group the randomly ordered users into K groups; the K-th fold cross-validation is performed holding the K-th group out as the test sample, and the remaining groups in as the training sample.

For each user, we have a series across all days T and decision points t per day of Reward, Action, and Context. We will refer to them as $(\mathcal{R}, \mathcal{A}, \mathcal{S})_{(N,T,t)}$, which are implicitly indexed in order by the user, day, and decision point. Thus, there are $N \times T \times t$ data points.

2.3 Residual Formation

Within each test batch and training batch, we can conduct ordinary least squares linear regression to residualize additional effects for each user from the contextual and interaction effects. Specifically, we use the model in Equation 2.3:

$$\mathcal{R} \sim \begin{bmatrix} f_1(\mathcal{S}), & \mathcal{A} \odot f_2(\mathcal{S}) \end{bmatrix}^T \Theta + \varepsilon \tag{2.3}$$

where $\odot$ denotes element-wise multiplication; i.e., for each user $n \leq N$, day $t \leq T$, and decision point $d \leq D$, we multiply the p_2-dimensional interaction component by the action of that day. Recall that f_1, f_2 are the baseline and interaction functions that are parameters of the generative model.

We obtain $\hat{\Theta}$ as well as a series of ε for each data point.

2.4 Simulated User Generation

For both the training and test original users' series of $(\mathcal{R}, \mathcal{A}, \mathcal{S})$, we can create N_{sim} users with duration T_{sim} data, keeping t decision points per day.

To create each of the N_{sim} simulated users, we randomly sample an existing user from HSv1, concatenate all available days of the sampled existing user to the simulated user, and sample until the simulated user has N_{sim} days of data points.

This is reproduced in Algorithm 1, where we feed in either a training or testing batch of users $(\mathcal{S}, \varepsilon)^{batch}_{N,T,D}$.

We note finally that there are no missing data points in each of the N_{sim} users, and that for testing, we set availability to True always.

Algorithm 1: Simulated User Generation Pseudocode

```
Data: $(\mathcal{S}, \varepsilon)^{batch}_{N,T,D}$
Result: $(\mathcal{S}, \varepsilon)^{sim}_{N,T,D}$
1 for $1 \leq n \leq N_{new}$ do
2     while $(\mathcal{S}, \varepsilon)^{batch}_{n,:,:}$ does not have $N_{sim}$ data points do
3         Sample a single user $n' \in [1, N_{batch}]$ without replacement;
4         Append $(S, \varepsilon)^{batch}_{n',:,:}$ to $(S, \varepsilon)^{sim}_{n,:,:}$;
5     end
6 end
```

2.5 Training Bandit Algorithm Tuning Parameters

We will try to set $N_{sim} = 1000$ for the number of training users, as to average across the data. If there is computational difficulty, we will use $N_{sim} = 100$ for the training.

For each variant of the Bandit Algorithm, we will tune parameters differently. These will be addressed in Section 3.2.

Ultimately, the models will be tuned according to minimizing the mean regret across all users and across the entire T_{sim} day simulation.

Should we minimize based on a different metric? A weighted average of standard deviation across users of mean regret per user too?

2.6 Simulated User Testing and Quality Metrics

Once the initialization parameters have been trained for the batch, we run B batches of the testing simulated users, and analyze the quality metrics described below:

1. Examine time series of $\pi_t(1|S_t)$, the probability of taking action 1 for all t, looking through several users.
2. Examine $\pi_t(1|S_t)$ vs $opt_t(S_t)$ for all t, looking through several users.

   We define $opt_t(S_t)$ as the optimal probability in context S_t:

$$opt_t(S_t) = 0.8\ 1\{\text{optimal action is 1 in } S_t\} + 0.2\ 1\{\text{optimal action is 0 in } S_t\}. \quad (2.4)$$

3. Examine $|\pi_t(1|S_t) - opt_t(S_t)|$ for all users, averaging over all N users for each time point t.
4. Examine $|\pi_t(1|S_t) - opt_t(S_t)|$ for all users, plotting histogram of each user's mean
5. Examine cumulative regret over t, plotting average over all users as well as for several individual users.

   Cumulative regret is the expected reward of the bandit minus reward of the optimal policy.
6. Examine the number of actions taken at each time t, plotting histogram across all t for each of several simulated users, as well as the average taken across all N users plotted at each time t.

3

Models

Several different variants of the Multi-armed Contextual Bandit algorithm were investigated to test feasibility in the HeartSteps mobile application.

3.1 Reward Generative Models

3.1.1 Basic Linear Generative Model

The most basic generative model for rewards is based on the equation:

where we set $p_1 = 8, p_2 = 3, f_1 : \mathcal{S} \to \mathbb{B}^{p_1}$ to be the identity of the 7 features plus a bias feature, and $f_2 : \mathcal{S} \to \mathbb{R}^{p_2}$ to be the identity on the first 3 features.

3.1.2 Additional Models

Can test time-varying reward functions, or perhaps some non-identity/non-linear functions?

3.2 Bandit Algorithm Variants

Can also use this section to describe motivation for each part of the Bandit algorithm.

We currently are using Peng's Algorithm 2 (Kristjan's Bandit Algorithm for HS2 (Action-Center Version), which contains a Gaussian Process, a Feedback Controller based on recent dosage, probability clipping, with action centering on the Gaussian Process update.

We plan on including/excluding each of the 4 modifications above (for $2^4 = 16$ slightly different variants), checking whether we need to include them or not.

There are the following parameters to optimize:

- Gaussian Prior parameters $(\gamma, \mu_\Theta, \Sigma_\Theta)$ (Peng uses μ_β, Σ_β instead)

- Feedback controller parameters (λ_c, N_c, T_c), where N_c is the desired dosage (number of 1 actions) over the past T_c decision times, and λ_c is the coefficient on how powerful the controller is
- σ^2, an estimate of the reward noise variance
- Probability clipping $(\pi_{\min}, \pi_{\max})$. We set these to $(0.2, 0.8)$ from domain science, that we require some amount of randomization.
- Baseline Features $f_1 : \mathcal{S} \to \mathbb{R}^{p_1}, f_2 : \mathcal{S} \to \mathbb{R}^{p_2}$. We set these to identity functions, where f_1 gives a bias term plus the original 7 context features, and f_2 gives the first 3 features for interaction terms.

We aim to tune the parameters in the above order.

1. Tune γ through parameter sweep on γ.
2. Tune Σ_Θ setting it to $v\mathbb{I}$, where $v \in \mathbb{R}$ is a scalar and $\mathbb{I}$ is the identity matrix; parameter sweep on v.
3. Set $\Sigma_\Theta = \Theta$ from the training data regression.
4. Tune λ_c through parameter sweep on λ_c.
5. Tune T_c through parameter sweep on T_c. Will stick to some discrete values such as $(5, 10, 50, 70)$ to not overfit.
6. Tune N_c setting it to mT_c, where T_c was the optimal value from the previous tuning step, and m is a scalar, likely in the range $(0.25, 0.75)$.
7. Set σ^2 to $\hat{\sigma^2}$, the empirical residual (noise) variance.
8. Set $\pi_{\min} = 0.2$, $\pi_{\max} = 0.8$.
9. Set f_1, f_2 to identity mappings.

Quick Note: on test data (i.e. random $\mathcal{S}$ and $\Theta = 100 \cdot \text{range}(11)$, and small Gaussian noise), the implemented Bandit Algorithm works very well to learn the true Θ when no action-centering occurs; otherwise, 3 coefficients in Θ are thrown off by subtracting π_t from A_t on line 26 in the algorithm, but the remainder are perfectly fine.