A Bayesian Nonparametric Approach to Multi-Task Learning for Contextual Bandits in Mobile Health

Citation

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37372687

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story
The Harvard community has made this article openly available. Please share how this access benefits you. Submit a story.

Accessibility
A Bayesian Nonparametric Approach to Multi-Task Learning for Contextual Bandits in Mobile Health

Prasidh Hundraj Chhabria

A thesis presented
to the Committee on Degrees
in Applied Mathematics
in partial fulfillment of the requirements
for the degree of

Bachelor of Arts
with Honors

Harvard College
*Cambridge, Massachusetts*

March 2022
Acknowledgments

I owe my deepest gratitude to Professor Susan A. Murphy for advising my foray into the realm of statistical reinforcement learning. While her deep expertise and intuition are an invaluable gift to so many students, I am especially grateful for her kindness in empowering me to drive this work with great autonomy: ideating from scratch, anticipating and tackling challenges. Her work simultaneously advances statistical science and addresses pressing challenges in chronic disease — a rare combination that inspires me to pursue the convergence of my passions.

I am grateful to Kelly Wang Zhang and Eura Shin of the Murphy lab for our brainstorming chats: from offering feedback on this project in its infancy as part of Professor Murphy’s STAT 234 seminar to guiding me through the process of presenting my work at academic conferences.

I am indebted to Professors Joe Blitzstein, Michael Brenner, Finale Doshi-Velez, and Demba Ba for teaching me (and trusting me to teach other students) the foundations of probability theory, statistical inference, and rigorous model-based approaches to ubiquitous inverse problems. Their courses have given me the tools to engage with the contemporary literature and tackle problems in inference and statistical learning.

I owe special thanks to my mentors, Professors Sean Eddy, Ali Asani, K. Viswanath, and Dr. Aastha Kant, for sharing with me their transformative work across the sciences and humanities, and for investing much time and compassion in my growth and wellbeing.

I thank Dean Stephanie Pausell and Kevin Madigan, along with the community of Eliot House at large, for giving me a home away from home and the most gorgeous, historic dining hall in which I have had the privilege of writing this thesis.

I thank the Applied Math advising team for their constant support and for our energizing research lunch events.

To my dear friends who drafted their theses with me: Anagha and Awnit, I am forever grateful for all that I have learned from you. Lauren, Janhavi, Meiling, Shi Le, Shuvom, Roshni, and Zoya – thank you for your unwavering support and for inspiring me with all your work.

Finally, this work is for my mother, who taught me the value and intricacy of human health, my father, who instilled in me an appreciation for the scientific method, and my second mothers (ma’si’s), to whom I owe a lifelong love for learning.
Abstract

Reinforcement learning algorithms have found utility in a number of digital interventions, such as in mobile health (mHealth), wherein $N$ users are followed and sequentially treated over $T$ timesteps with the aim of optimizing a health outcome. In the precision medicine paradigm, the aim of the intervention (algorithm) is to learn a personalized, optimal treatment policy for each of the $N$ users that optimizes the health outcome of interest. Learning such a policy can be prohibitively slow due to sparse and noisy data for each user; in practice, users can disengage from the intervention before a good policy is learned.

To address this problem, we aim to pool information across users to speed learning while preserving individualized treatment. However, pooling data across dissimilar users can lead to disastrous treatment decisions and outcomes. These challenges underscore the need for rigorous, model-based approaches to defining similarity across users before pooling.

We model the $N$-user mHealth setting with a contextual bandit environment and formalize similarity across users using a Dirichlet Process mixture model. We offer a variant of blocked Gibbs sampling to infer clusters among users; we further propose DPMM-Pooling, an integrated intervention algorithm to learn clusters among users and share data within clusters, in order to speed the learning of optimal and individualized treatment policies.

We evaluate DPMM-Pooling in simulated mHealth settings, across a range of parameters such as the number of ground-truth clusters, noise in observed outcomes, and the time of pooling. We find and analyze key bias-variance tradeoffs in pooling pertaining to parameters of the environment. We also find that DPMM-Pooling is relatively robust to likely forms of mild and extreme model misspecification. Finally, we outline the implications of our results for the design of pooling-based mHealth interventions in practice.
Thesis Outline

The main contribution of this manuscript is to propose and evaluate a multi-task learning ("pooling") algorithm based on the Dirichlet Process mixture model formulation. Herein, $N$ contextual bandit tasks are clustered via MCMC inference (blocked Gibbs sampling) such that information may be shared among them to accelerate the reward trajectory across tasks. Our proposed methods are motivated within the mobile health (mHealth) setting, characterized by sparse and noisy data.

Chapter 1 introduces the mHealth setting, including Just-in-Time Adaptive Intervention design; reviews contextual bandits and Thompson Sampling in reinforcement learning; offers the motivating example of HeartSteps, an adaptive intervention; outlines the problem specification.


Chapter 3 formalizes the statistical formulation of our problem setting; presents our proposed algorithm, DPMM-Pooling; evaluates DPMM-Pooling across simulated environments and under model misspecification.

Chapter 4 explains key tradeoffs apparent in our results; outlines the implications of our results for mHealth interventions in practice; identifies avenues for future work.
# Table of Contents

## Chapter 1: Introduction

1.1 Motivation: Mobile Health  
1.1.1 Intervention Design in mHealth: JITAI and MRT  
1.1.2 Characteristic Challenges in mHealth  
1.2 Reinforcement Learning and Contextual Bandits  
1.2.1 Contextual Bandit Setting  
1.2.2 Thompson Sampling  
1.3 Motivating Example: HeartSteps  
1.3.1 Problem Specification

## Chapter 2: Background

2.1 Related Work in Multi-task RL  
2.1.1 Central Questions in Multi-task Learning  
2.1.2 Extant Literature  
2.1.3 Multi-task mHealth  
2.2 The Dirichlet Process  
2.2.1 Alternative Constructions  
2.2.2 Dirichlet Process Mixture Models (DPMM)  
2.3 Inference  
2.3.1 MCMC Methods: Gibbs Sampling  
2.3.2 Variational Methods

## Chapter 3: Models, Methods, and Results

3.1 Statistical Formulation  
3.1.1 Dirichlet Process  
3.1.2 Cluster- and Individual-Specific Parameters  
3.1.3 Reward Model  
3.2 Algorithms  
3.2.1 Baseline: Individual RL  
3.2.2 Proposed Algorithm  
3.2 Experiments  
3.3.1 Generative Environment  
3.3.2 Evaluation Metrics  
3.3.3 Varying Noise ($\sigma^2$)  
3.3.4 Varying Inter-Cluster $\beta$ Distance  
3.3.5 Varying $T_{pool}$  
3.3.6 Varying True Number of Clusters $K^*$  
3.3.7 Investigating Reward Model Misspecification

## Chapter 4: Discussion

4.1 Conclusions and Implications  
4.1.1 Pooling: A Bias-Variance Tradeoff  
4.1.2 Model Misspecification in mHealth  
4.1.3 Ethical Considerations  
4.2 Future Work  
4.2.1 Dynamic Pooling  
4.2.2 Interpretability  
4.2.3 Social Network-Based Priors  
4.2.4 Delayed Effects of Interventions

## Bibliography
Chapter 1

Introduction

The need for model-based multi-task or transfer learning methods pervades a range of disciplines and applications, including robotics and industrial control, but encounters particularly high stakes in the realm of healthcare and precision medicine. Such methods may be used to transfer information across patients, harnessing an enlarged body of data to make quicker, more effective treatment decisions for each individual patient.

Intelligent systems designed to transfer data or inferences across patients can offer significant clinical benefits but must be rigorous, principled, and interpretable, as incorrect transfer decisions can have potentially disastrous impacts on treatment outcomes. The onus falls, hence, upon the designer of an intervention to leverage the gains from model-based transfer while, as in any clinical setting, minimizing harm.

Although the methods developed in this work apply to a range of sequential decision-making problems (albeit limited to the linear contextual bandit setting, which we shall soon formalize), we motivate the statistical work in the context of mobile health (mHealth). The mHealth setting serves to exemplify a high-stakes domain wherein room for error is minimal, underscoring the need for careful modeling decisions throughout this work.

This first chapter introduces mHealth, as well as the contextual bandit framework in which our proposed algorithms will act. We explore these frameworks in the single-patient setting before exploring the need for multi-task learning (which we term pooling); finally, we arrive at the central research question for the present work.
1.1 Motivation: Mobile Health

The emergence of precision medicine, a medical approach in which treatment decisions are informed largely by individual patient characteristics, profoundly challenges the traditional statistical approach to clinical decision-making. While the conventional approach uses population averages as estimators for treatment outcomes or efficacy, the precision medicine approach seeks inferences on the individual patient level. [20] Hekler, Klasnja et al. term this the “small data paradigm.” [25]

Within this paradigm, specifically within the domain of chronic disease such as hypertension or diabetes, mobile health (mHealth) is a growing class of methods that seek to sustainably change long-term health behaviors. Broadly, mHealth involves three general tasks: (1) detecting and/or predicting risk-producing behaviors or states in real-time using advanced sensor technology, (2) leveraging machine learning or statistical inference to determine the optimal intervention, and (3) delivering the chosen intervention to promote short- and long-term health behavior change. [34,75] While tasks (1) and (2) present nontrivial challenges in the domain of data science, (2) and (3) involve knowledge of an underlying, dynamic model of health behavior; as a result, mHealth is necessarily an interdisciplinary endeavor spanning computational and behavioral science. [1,41,65]

mHealth interventions have taken the form of text messages (e.g., to aid college students in smoking cessation [9, 57]) or encourage weight loss behaviors in adults [53]) and smartphone apps (e.g., to encourage patients combating alcohol dependence to rethink when in high-risk locations for relapse [23] or nudge employees from their seats whilst sedentary at work [72]).

1.1.1 Intervention Design in mHealth: JITAI and MRT

The majority of mHealth interventions, such as those listed above, employ relatively “static” approaches to both intervention delivery and evaluation.

In terms of delivery, a fixed set of text messages or app nudges are delivered to users at a fixed time of day; the mode of delivery is not adaptively determined according to user characteristics or behavior. This unmet need for personalization foreshadows the utility of reinforcement learning for individualized intervention
delivery in mHealth \cite{18,78}; specifically, we will engage with the *Just-in-Time Adaptive Intervention* (JITAI) design.

In terms of evaluation, mHealth studies generally use the traditional randomized controlled trial (RCT) design, which entails monitoring fixed control and treatment groups over an extended duration until causal conclusions can be drawn. The occasional meta-analysis then synthesizes conclusions from a set of mHealth RCTs. \cite{19,26,43,74} Ideally, a real-time adaptive intervention could benefit from a real-time study design for evaluation, such that causal conclusions can be drawn quickly and applied iteratively to improve patient outcomes. For this purpose, we will review the *micro-randomized trial* (MRT) design.

**JITAI.** Interventions delivered via the JITAI design have three defining characteristics: (1) *individualized*, i.e., use each user or patient’s information to determine when and how to intervene; (2) *adaptive*, i.e., dynamically individualize by accumulating information and modifying intervention decisions over time; and (3) *just-in-time*, i.e., attempt to deliver the selected intervention at the right time. \cite{49,69}

We introduce the key terminology for JITAI, which will largely resemble that of the reinforcement learning framework (see 1.2). Assume the JITAI serves \( N \) users, each labeled \( i \in \{1, 2, \ldots, N\} \).

- The JITAI takes place over a time horizon \( \mathcal{T} \) consisting of equally spaced timesteps \( t = 1, 2, \ldots, T \). A subset of these times is the set of *decision times*, at which interventions are selected and delivered. \cite{14} Hence, the JITAI occurs in an *online* setting, i.e., additional data is observed at each time \( t \), as opposed to all at once (the “offline” setting). For simplicity, let us assume every \( t \) is a decision time.

- Each user \( i \) is said to be in a *state* denoted \( S_{i,t} \) at time \( t \). A \( p \)-dimensional vector, the state includes \( p \) scalar variables describing the user’s physiological or psychological parameters, as well as aspects of the external environment. This state may be measured via smartphone, wearable sensors, or behavioral self-report survey. \cite{24} Some subset of the \( p \) scalar quantities is termed the *set of tailoring variables*; these are used to select interventions for the user. \cite{44,73}
Each user $i$ is described at each $t$ as either available ($l_{i,t} = 1$) or unavailable ($l_{i,t} = 0$) to be engaged. If the user is unavailable, no intervention is delivered. If the user is available, interventions are considered. Possible conditions under which a user may be deemed unavailable include whilst driving or at work. [62]

At each decision point, a set of intervention options $\mathcal{A}$ exists for each available user. For simplicity, we assume $\mathcal{A}$ is fixed across time $t$ and users $i$.

A decision rule (or policy) $\Pi$ is used to select an intervention $A_{i,t}$ from $\mathcal{A}$. [54]

For each $i$ at each $t$, the short-term goal of the intervention is measured via the proximal outcome; in the context of smoking cessation, for example, this could be a daily or hourly self-report of stress and/or smoking. [4, 22] We denote the proximal outcome $R_{i,t}$ due to its connection to a key quantity in reinforcement learning, as we shall see in 1.2.

The end goal (or distal outcome) of an mHealth intervention is generally a sustained change in health behavior, wellbeing, or “self-efficacy.” [3] Abstract indicators of this type are difficult to quantify [45]; more generally, it is often impossible to obtain observations of such a distal outcome within the horizon $T$. Hence, we will disregard the distal outcome to adequately simplify the setting.

**MRT.** The MRT is an experimental design framework designed to inform the JITAI intervention design and provide data to estimate individual-level treatment effects for interventions over time. Within this framework, the JITAI decision rule $\Pi$ involves sequential randomization. At each decision point $t$, user $i$’s state $S_{i,t}$ is captured; accordingly, an intervention $A_{i,t}$ is chosen by some explicit randomization; finally, the proximal outcome $R_{i,t}$ is observed. [31, 46, 47]

Sequential randomization within the MRT allows for estimation of treatment effects for an intervention. Consider, as we will throughout the present work, a binary set of intervention options $\mathcal{A} = \{0, 1\}$ where 1 represents treatment and 0 denotes no treatment. At each decision time $t$, the decision rule $\Pi$ prescribes a
randomization to $A_{i,t} = 1$ or $A_{i,t} = 0$ for each user $i$. [13]

Further, as in the causal inference literature [58], we define the potential outcomes for user $i$ as random variables $R_{i,t}(a)$, representing the underlying outcome for user $i$ under some choice of intervention $a$, where $a = 0$ or $1$. Then, we are interested in estimating the treatment effect for user $i$,

$$\beta_i = \mathbb{E}[R_{i,t}(1) - R_{i,t}(0)]$$

which is equivalent to the difference in potential outcomes under treatment and no treatment for user $i$. Note that at a single $t$, we may only observe $R_{i,t}(A_{i,t})$, that is, the outcome under the chosen intervention option; we may never simultaneously observe both potential outcomes for a user $i$ at a single time $t$.

The treatment effect $\beta_i$ is then estimated based on some working statistical model for the proximal outcome. One such example is

$$R_{i,t} \mid A_{i,t} = R_{i,t}(A_{i,t})$$

$$R_{i,t}(a) \sim \mathcal{N}(f(S_{i,t}, a), \sigma^2)$$

where $f$ is some function of state and chosen intervention, and $\sigma^2$ is some noise parameter. In 1.3, we will explore a similar problem of inference for our specific setting.

Over a sufficiently long time horizon $T$, the MRT design has the advantage of randomizing each user to control and/or treatment several (perhaps hundreds or thousands of) times, allowing for precise inference. [40] However, we rely on some key assumptions [10, 16] in order to draw causal conclusions from an MRT:

- **Consistency**: the observed outcomes $R_{i,t}$ are equivalent to the potential outcomes under the actual treatment or intervention. In other words, $R_{i,t} = R_{i,t}(0)$ at decision times $t$ when $A_{i,t} = 0$ and $R_{i,t} = R_{i,t}(1)$ at times $t$ when $A_{i,t} = 1$.

- **Sequential ignorability**: for all $i$ and each $t \leq T$, the potential outcomes

$$\{R_{i,t}(0), R_{i,t}(1), R_{i,t-1}(0), R_{i,t-1}(1), ..., R_{i,T}(0), R_{i,T}(1)\}$$
are independent of $A_{i,t}$. This is guaranteed by design, as the randomization probabilities $P(A_{i,t} = 1)$ are always explicitly known and based purely on data observed before time $t$.

- **Stationarity**: the distribution of the potential outcomes remains fixed over time; i.e., $f(S_{i,t}, a)$ and $\sigma^2$ do not vary with time. Then, as per Eq. 1.1, $\beta_i$ remains fixed over time as well.

- **No delayed effects**: there is no effect of action $A_{i,t}$ on potential or observed outcomes after time $t$. Since the potential outcomes at time $t$ depend only on a function of the current state and action $f(S_{i,t}, A_{i,t})$, we index potential outcomes only with a single action, i.e., $R_{i,t}(1)$ and $R_{i,t}(0)$.

The final assumption may be dropped, but such an acknowledgment of nonstationarity complicates both intervention design and causal inference. [7,37] Hence, we choose to assume a stationary setting for the remainder of this work.

### 1.1.2 Characteristic Challenges in mHealth

1. **Disengagement and burden.** Behavioral interventions for chronic conditions, while ideally instrumental in improving long-term health behaviors, are likely to induce cognitive burden, such as by overwhelming users or interfering with their daily activities. [15,33,38] Hence, a successful JITAI should prioritize learning an optimal treatment policy for users as quickly as possible, before potential disengagement.

2. **Sparse and noisy data.** mHealth users are often unavailable to engage; consequently data on each individual user is likely sparsely collected. [48,64] State and proximal outcome data collected by wearable sensors is also generally noisy (high variance) [17,30,55], hence limiting data quality and slowing inference of treatment effects.

3. **Nonstationarity.** Parameters we assume to remain constant over time (such as $f(S_{i,t}, a)$ or $\sigma^2$) are likely to be dynamic in a real-life setting. [6,42]

4. **Misspecification.** In addition to nonstationarity, our models are likely to be otherwise misspecified. For example, the working model for the proximal outcome is unlikely to fully match the data generating process in reality.
5. Post-study causal inference. While a JITAI aims to optimize a specified outcome for each of its users over the time horizon $T$, it is also important to preserve the ability to conduct causal inference after $T$ has elapsed. These objectives are not always complementary. In this work, we attempt to address the first two challenges. In order to formalize this attempt, we proceed to set up the reinforcement learning framework.

### 1.2 Reinforcement Learning and Contextual Bandits

In the reinforcement learning (RL) framework, an agent uses a decision rule to adapt to its state in the environment by choosing optimal actions. A sequence of optimal actions is one that maximizes a cumulative sum of rewards received as a result. In the general case, the agent’s objective over a time horizon $t = 1, \ldots, T$ can be defined as

$$\max_{\Pi} \sum_{t=1}^{T} R_t$$  \hspace{2cm} (1.2)

where $R_t$ represents the reward obtained at time $t$. Towards this objective, the agent’s task is to learn an optimal policy (or decision rule), which is formally a mapping $\Pi : S \mapsto A$ from the state space (or set of all possible states) $S$ to the action space $A$ so as to maximize cumulative sum of rewards. Hence, the maximization in (1.2) is over the space of all available policies.

The reader will recognize parallels in notation and structure to the JITAI-MRT framework in mHealth. The JITAI algorithm, as the agent, iteratively observes user $i$’s state $S_{i,t}$, selects an intervention (action) $A_{i,t}$ according to some policy $\Pi$, and observes the proximal outcome (reward) $R_{i,t}$. Additionally, the JITAI-MRT design requires that the policy $\Pi$ involve some explicit randomization at each $t$. Given that the JITAI is sequential, the policy $\Pi$ for a user $i$ is not static; in the extreme case, it is updated at every time $t$, so we employ a subscript $\Pi_{i,t}$. 

[5, 21]
1.2.1 Contextual Bandit Setting

One simple yet powerful specification of the reinforcement learning framework is the **contextual bandit** setting. Woodroofe proposed the contextual bandit setting in 1979, motivated by the problem of treating patients each with certain defining characteristics (or **contexts**). [76] Since then, contextual bands have been widely applied to model clinical decision-making problems. [68]

We will construct the contextual bandit by considering a single user \( i \) in the JITAI. At each \( t \), the agent observes a state \( S_{i,t} \) and must select an action or intervention \( A_{i,t} \) from a fixed set of actions \( \mathcal{A} \) using the current policy \( \Pi_{i,t} \). In the contextual bandit setting (Figure 1.1), the agent believes:

- that the reward \( R_{i,t} \) is some function of the state (or context) \( S_{i,t} \), action \( A_{i,t} \), and some defined parameters. Above, we termed this \( f(S_{i,t}, A_{i,t}) \). For the rest of this work, we will assume a linear functional form for \( f \), as is common in the literature across applications spanning health, advertisements, and recommendation engines. [8, 28, 35, 36, 77]
- that the action \( A_{i,t} \) is independent of the following state \( S_{i,t+1} \).

For the sake of brevity, we will henceforth refer to the contextual bandit JITAI as the **mHealth setting**, although one can easily construct several other viable RL formulations for mHealth.

As in Eq. 1.1, we encountered the idea of a scalar treatment effect \( \beta_i \) for user \( i \), in the case of a binary action space \( \mathcal{A} = \{0, 1\} \). In the contextual bandit setting, with a linear reward function, we can say

\[
R_{i,t} = g(S_{i,t})^\top \omega_i + A_{i,t} \cdot f(S_{i,t})^\top \beta_i + \epsilon_{i,t}
\]

(1.3)

where \( \epsilon_{i,t} \) is some zero-mean scalar random variable such as \( \epsilon_{i,t} \sim \mathcal{N}(0, \sigma^2) \); \( \beta_i \) is now a vector of treatment effect coefficients; \( \omega_i \) is a vector of coefficients corresponding to a “baseline” reward (without treatment); \( f, g \) represent sub-vectors constructed with summaries of features in the \( p \)-dimensional vector \( S_{i,t} \).

Note that \( f \) is not to be confused with \( f(S_{i,t}, A_{i,t}) \), the mean function for the potential outcome at time \( t \) in state \( S_{i,t} \) and under action \( A_{i,t} \). In the above case of
Eq. 1.3, this mean of the reward is \( f(S_{i,t}, A_{i,t}) = g(S_{i,t})^\top \omega_i + A_{i,t} \cdot f(S_{i,t})^\top \beta_i \).

Alternatively, in the language of potential outcomes, by the consistency assumption:

\[
\begin{align*}
E(R_{i,t}(0)) &= g(S_{i,t})^\top \omega_i \\
E(R_{i,t}(1)) &= g(S_{i,t})^\top \omega_i + f(S_{i,t})^\top \beta_i \\
E(R_{i,t}(1) - R_{i,t}(0)) &= f(S_{i,t})^\top \beta_i
\end{align*}
\]

We are now familiar with the contextual bandit JITAI as the “mHealth setting”. However, the question of how to optimize a policy \( \Pi \), such that the agent selects actions to maximize a sum of rewards, remains unanswered. We introduce the Thompson sampling algorithm, which provides the foundation for the algorithms to be proposed in the present work.
1.2.2 Thompson Sampling

Selecting actions or interventions to maximize rewards is a nontrivial problem in reinforcement learning, particularly because it is subject to the explore-exploit tradeoff. In our binary action space \( A = \{0, 1\} \), in order to select reward-maximizing actions, the agent must learn which action (1 = treatment or 0 = no treatment) is optimal. Seeking to maximize the objective in Eq. 1.2, the agent must constantly choose between exploiting its current knowledge (i.e., choosing the action it currently perceives as optimal) and exploring other actions in order to gain information and identify the optimal action with greater certainty. [52, 67]

Thompson sampling [60, 71] provides an elegant solution to said tradeoff. [11, 29] Eq. 1.6 captures that treatment is optimal (i.e., expected to yield greater reward than no treatment) when \( f(S_{i,t})^\top \beta_i > 0 \). The agent does not know the ground truth \( \beta_i \), but may update its beliefs on the underlying \( \beta_i \) as more realizations of state-action-reward tuples \((S_{i,t}, A_{i,t}, R_{i,t})\) are observed. Under Thompson sampling, at time \( t \), the agent would simply sample \( \beta_i \) from its current distribution. If, for the sampled \( \beta_i \), it is found that \( f(S_{i,t})^\top \beta_i > 0 \), the agent selects action \( A_{i,t} = 1 \); else, it selects \( A_{i,t} = 0 \). This has the effect of sampling, at each \( t \), each action with the current (perceived) probability that the action is optimal. This complies perfectly with the JITAI-MRT design, in which the policy (decision rule for actions) must involve some randomization at each decision time.

Thompson sampling is, hence, an inherently Bayesian procedure. [50, 51] In our setting, we set a prior distribution on \( \beta_i \), such as

\[
\beta_i \sim \mathcal{N} (\mu_{i,1}, \Sigma_{i,1})
\]  

(1.7)

As the agent observes additional state-action-reward tuples, it may update the distribution of \( \beta_i \) (see Chapter 3 for a detailed discussion of priors and posterior updates in this setting). This estimation of \( \beta \) is, then, a case of Bayesian linear regression.

At each time \( t \), action \( A_{i,t} = 1 \) is sampled with probability \( \mathbb{P}(f(S_{i,t})^\top \beta_i > 0) \), according to the current distribution of \( \beta_i \) at time \( t \).

The general algorithm for Thompson sampling in our contextual bandit JITAI
setting (with a linear reward function \[2\] and a Gaussian prior) is outlined below.

**Algorithm 1.1** Thompson Sampling for Contextual Bandit JITAI

**input**
- • prior parameters \(\{\mu_1, \Sigma_{i,1}\}\)
- • length of time horizon \(T\)

fix prior distribution \(\beta_t \sim \mathcal{N}(\mu_0, \Sigma_0)\)

for \(t = 1, \ldots, T\) do

observe \(S_{t,1}\)

sample \(\beta_t \sim \mathcal{N}(\mu_t, \Sigma_t)\)

if \(f(S_{t,1})^\top \beta_t > 0\) then

select \(A_{t,1} = 1\)

else

select \(A_{t,1} = 0\)

observe \(R_{t,1}\)

update \(\mu_{t,i,t+1}, \Sigma_{t,i,t+1}\) using appropriate Bayesian update or conjugacy

---

**1.3 Motivating Example: HeartSteps**

For concreteness, we motivate the contributions of this work with the example of HeartSteps, a JITAI to encourage physical activity among stage 1 hypertension patients. \[32\]

In this setting, the state \(S_{i,t}\) is a vector of scalar features, including location, current temperature, and the user’s level of app engagement. The action space \(A\) is binary, where \(A_{i,t} = 1\) represents treatment via a push notification suggesting physical activity and \(A_{i,t} = 0\) represents no intervention. Push notifications for treatment are designed carefully based on the domain behavioral science. Finally, the user’s 30-minute step count following time \(t\) constitutes the reward \(R_{i,t}\).

In HeartSteps V2, the second iteration of the micro-randomized trial, Thompson sampling was used as the action selection algorithm. \[39\]

Revisiting 1.1.2, recall that the challenges of disengagement and noisy/sparse data necessitate that, in the mHealth setting, we design an agent that can learn the optimal treatment policy for each user as quickly as possible.
1.3.1 Problem Specification

If the data collected on each individual user is sparse and noisy, a natural solution is to leverage data across users to speed the agent’s learning of optimal treatment policies. [12, 56]

One proposal to leverage other users’ data would be to learn one common treatment policy across all users, by pooling every user’s state-action-reward observations together. We will call this full pooling. Although a full pooling approach would still consider each individual user’s current context in order to make individual treatment decisions, users are likely to differ across parameters such as treatment effects $\beta_i$. Therefore, the full pooling proposal, in sharing data across dissimilar users, defeats the purpose of precision medicine, which is to accurately personalize. The only logical way to leverage other users’ data is, then, to share data exclusively across those users who are (in some interpretable way) “similar” in their response to treatment.

This abstract notion of “similarity” must be formalized rigorously, as using user $j$’s data to inform treatments for user $i$ could be detrimental if $i$ and $j$ are not, in fact, similar in their responses to treatment. Hence, we must model a structure of similarity across users, infer which users are indeed “similar” according to this model, and share data across these “similar” users.

In our model, a user $i$’s response to treatment is most fully captured by the treatment effect coefficients $\beta_i$. Hence, we can define users $i$ and $j$ as similar if $\beta_i = \beta_j$. More generally, we assume that users exist in clusters, each of which shares a common $\beta$ parameter.

The present work will propose a solution of the following general form: we run a JITAI over a time horizon $t = 1, 2, \ldots, T$. Until some pre-determined timestep $t = T_{\text{pool}}$, we run an individualized JITAI, collecting data and delivering interventions to each user independently. At time $t = T_{\text{pool}}$, we infer the underlying clusters amongst users. Thereafter, for $t = T_{\text{pool}} + 1, \ldots, T$, we pool all observed data across users in the same inferred cluster; then, to select actions $A_{i,t}$ for each user $i$, we use the information from all the data in its cluster, as opposed to user $i$’s own data alone.
This approach is illustrated via graphical models in Fig. 1.2 and Fig. 1.3. Fig. 1.2 resembles a JITAI in the contextual bandit setting with no pooling, whilst Fig. 1.3 identifies the additional dependencies arising from the pooling of data post $T_{pool}$ (in blue arrows).

What follows is a doubly challenging problem. On one hand, since learning an optimal, personalized treatment policy in the mHealth setting is prohibitively slow, we would benefit from sharing data across similar users. On the other hand, in order to infer said similarity, we only have access to sparse and noisy data on each user, for a short interval $t = 1, \ldots, T_{pool}$. This frames the central research question of our work: how, in a sparse and noisy data setting such as the contextual bandit JITAI, can we pool data in a principled manner such that we accelerate learning treatment policies without compromising on their optimality?
Figure 1.2: Graphical model of the $N$-user contextual bandit setting without pooling
Figure 1.3: Graphical model of the $N$-user contextual bandit setting with pooling
Chapter 2

Background

The need for pooling in mHealth arises from a *multi-task problem*: since each user sits in a contextual bandit environment, learning the optimal treatment policy for a particular user is a contextual bandit task; with $N$ users, we have $N$ such tasks. Our aim is to infer and leverage shared structure across these several tasks to improve performance on each.

We now review relevant work in multi-task reinforcement learning, situating our contribution within the existing landscape of approaches. We then review the theory of the Dirichlet process, which will inform our model of the multi-task mHealth setting, and examine inference techniques for Dirichlet process mixture models in the literature.

2.1 Related Work in Multi-task RL

The intuition motivating multi-task learning is captured by Sebastian Thrun in his 1995 manuscript “Is Learning the $n$-th Thing Any Easier Than Learning the First?” [113] Thrun posits that transferring knowledge across $N$ tasks that share some underlying structure is likely to enable each individual task to learn efficiently from less training data. This heuristic highlights the utility of multi-task methods in mHealth, wherein data on each task (user) is noisy and sparse. Similar motivations are seen in applications spanning drug discovery [28,88], computer vision [128,26], and recommendation engines. [125]
2.1.1 Central Questions in Multi-task Learning

Zhang and Yang [135] define multi-task learning (MTL) as an effort involving \( N \) tasks in which the learning of each task \( i \) is assisted by knowledge contained in some or all of the other tasks \( j \in \{1, \ldots, N\}, j \neq i \). They characterize the three central questions of MTL: when to share, what to share, and how to share?

**When to share.** The question of “when to share,” in Zhang and Yang’s offline context, implies a principled criterion for identifying situations in which it is beneficial to share information across tasks, as opposed to relying on single-task learning. [132] This necessitates a model for similarity or shared structure across tasks; many MTL works have modeled tasks as realizations of some underlying “task distribution” which may be formalized in a variety of ways. [117]

However, in the context of a JITAI, which is an online, sequential setting, the question of “when to share” also finds additional meaning. Concretely, at what time \( t \) in the course of a contextual bandit JITAI would it be most beneficial to infer the underlying groups among users (tasks) and share information within these inferred clusters? In the online setting, one envision a case in which attempting to infer similarity among users too early may lead to very inaccurate inference due to lack of sufficient data. However, waiting too long to infer clusters and pool data across users can only defeat the purpose of pooling, which is to accelerate learning across users or tasks.

**How to share.** The second question of “how to share” points to a choice of four general approaches, as presented by Zhang and Yang, to modeling similarity across tasks in MTL. [103] The present work relies on the “task clustering approach,” in which tasks (i.e., mHealth users) are assumed to exist in underlying groups or clusters. The two-step MTL problem that follows is to (1) infer the underlying clusters and (2) share knowledge effectively within these clusters. For conciseness, we forgo a detailed discussion of the other three classes of MTL methods reviewed by Zhang and Yang.

**What to share.** Finally, the question of “what to share” involves the precise information that will be shared across tasks. [136] In our mHealth setting, we will propose sharing all state-action-reward tuples \((S_{t,t}, A_{t,t}, R_{t,t})\) across users that are considered similar. These shared tuples will then feed into the Thompson
sampling-based policy for each individual user (see Chapter 3 for details).

In the literature, the effectiveness of such information transfer across tasks can be evaluated by comparing to simple baselines, such as the cases in which either no data is shared or all data is shared across all tasks indiscriminately. We will employ similar baselines in Chapter 3.

2.1.2 Extant Literature

We review a brief taxonomy of methods towards multi-task learning in RL.

**Distillation and Representation.** Much work has focused on learning a common policy or functional representation for the reward function across all $N$ tasks in a multi-task problem. [47, 99] Teh et al’s “Distral” [106] learns a common (distilled) policy for all $N$ users; then, this distilled policy is used as a regularizer in the objective function that is maximized to learn the optimal policy in each individual task $i$. (This distilled policy across tasks can be viewed as an informative policy prior within a Bayesian paradigm.) Alternatively, advances in representation learning seek to learn a common low-dimensional functional form for (part of) the reward function that is assumed to be shared across tasks.

**Neighborhood Computation.** A popular alternative approach to prescribing a model of similarity across tasks is computing a “neighborhood” of related tasks for each task $i$ and sharing data within each neighborhood. [25, 62] While such methods are flexible, drawbacks include (1) the absence of a model to make conclusions interpretable and (2) the need to prescribe some arbitrary hyperparameter thresholding a maximum distance between two tasks $i$ and $j$ such that $(i, j)$ are deemed neighbors. Neighborhood computation must not be confused with our model-based task clustering approach, which infers an underlying structure across tasks as defined by some statistical model.

**Graph and Network Inference.** The relationships between tasks are often described by undirected weighted graphs or network structures. The exact network structure across tasks is learned as data is observed. Edges connecting tasks on a graph structure may be informed by a variety of observed features; popular works in this domain include Gang of Bandits (GOB) [15] and Horde of Bandits. [116]
**Meta-Learning.** Meta-learning broadly refers to the idea of modeling and exploiting shared features across tasks, often in the form of a “task distribution,” and applying this structure to efficiently solve new, previously unseen tasks. \[119, 41\] Several variations of model-agnostic meta-learning (MAML) \[33, 31, 137\] have established algorithms to sample models for a task \(i\) from some distribution over models. Meta learning is often described as “learning to learn.” Our work deviates from meta-learning in that we are primarily interested in directly inferring clusters of similar tasks (or users).

**Online MTL.** Online multi-task learning addresses the specific case in which new tasks are added one after the other. A number of solutions have been proposed in this area, many of which are Bayesian (e.g., set priors based on previous tasks). \[32, 73\] In mHealth, online MTL may be useful in the case of users joining a JITAI or micro-randomized clinical trial one after another (due to rolling trial recruitment or late adoption).

**Bayesian Nonparametrics.** A number of more recent model-based approaches to MTL have used Bayesian nonparametric methods, as does the present work. We detail the theoretical framework for Bayesian nonparametrics in 2.2 as our contribution lies within this domain. Past work has included learning a shared covariance function across tasks \[11, 102\] or using a Dirichlet Process prior \[127\] for simple offline prediction and classification tasks (in traditional supervised learning).

**2.1.3 Multi-task mHealth**

The utility of multi-task methods in the challenging mHealth setting has been widely recognized. Much work in this domain has focused on the offline setting, such as using hierarchical Bayesian formulations to infer shared structure across simple prediction tasks (such as predicting a user’s stress level tomorrow). \[104, 49\]

In the online JITAI domain, there remains an urgent need for principled methods to transfer information across \(N\) users (each representing a distinct task). Tomkins et al develop “Intelligent Pooling,” which uses a mixed effects model to describe how users deviate from a single population average parameter or set of parameters. \[114\] However, they acknowledge that such an approach may overlook clusters
of similar users or other, more complex forms of structure and heterogeneity in the data. Finally, Shin et al [97] consider a Gaussian process regression approach to learning the optimal treatment policy in a JITAI-MRT involving $N$ users; they propose a method to share kernel information across users to better inform the kernel selection process for each user $i$.

### 2.2 The Dirichlet Process

Among probabilistic models, the frequentist approach defines model parameters as unknown yet fixed quantities, while Bayesian approaches treat parameters as random variables. In the Bayesian paradigm, parameters are assigned prior distributions, which are updated to form posterior distributions after data is observed.

Probabilistic models can further be classified as parametric or nonparametric. While parametric models utilize a fixed and finite number of parameters, nonparametric approaches assume an infinite-dimensional parameter space. This confers the advantage of adaptive model complexity: while parametric models may easily over- or under-fit data by assuming an incorrect level of complexity (i.e., number of parameters), nonparametric models, appropriately regularized, can adapt or grow the parameter space as more data is seen, since no assumptions are made about the number of parameters beforehand.

The Dirichlet process (DP), initially proposed by Ferguson in 1973 [30], laid the foundation for Bayesian nonparametrics. The DP is a stochastic process from which draws are discrete probability distributions.

**Formal Definition.** Let $P_0$ be a distribution over sample space $\Theta$. Further let $(B_1, \ldots, B_k)$ represent a finite measurable partition of $\Theta$. For a *random distribution* or random probability measure $G$, we say $G$ is itself distributed according to a Dirichlet process, denoted $G \sim \text{DP}(\alpha, P_0)$, if

$$
(G(B_1), \ldots, G(B_k)) \sim \text{Dirichlet}(\alpha P_0(B_1), \ldots, \alpha P_0(B_k))
$$

(2.1)

for any finite measurable partition $(B_1, \ldots, B_k)$ of $\Theta$.

$P_0$ is called the *base distribution* of the DP; $\alpha > 0$ is the *concentration parameter*.
**Intuition.** One intuitive view of the DP is as a distribution over a space of distributions, which are each discrete with probability 1 (i.e., a set of discrete atoms, each assigned some probability). These atoms in turn can be viewed as draws from the base distribution $P_0$.

Any distribution $G$ sampled from the DP is a *discrete* distribution defined on atoms that are drawn from $P_0$ – which itself may be continuous. Therefore, draws from $G$ can, with non-zero probability, be identical to previous draws. [1] In fact, distributions $G$ sampled from the DP exhibit a “rich gets richer” (or, as in the network science literature, “preferential attachment”) property; the more an atom is repeated in a draw from $G$, the greater its probability of being drawn again.

Intuitively, this positions the DP well for clustering applications, wherein each atom is viewed as a “cluster”, with potential for infinitely many clusters, each with a countable mass.

**Conjugacy.** The DP offers great Bayesian utility as a prior distribution. Consider $N$ draws $\{y_i\}_{i=1}^N$ from an unknown distribution $G$. Placing a DP prior on $G$ gives the precise form of Eq. 2.1. [105] Then, by the conjugacy properties of the finite-case Dirichlet distribution, the posterior follows:

$$
(G(B_1), \ldots, G(B_k)) \mid y_1, \ldots, y_N \sim \text{Dirichlet}
\left(
\left(\alpha P_0(B_1) + \sum_{i=1}^{N} 1_{y_i \in B_1}\right), \ldots, \left(\alpha P_0(B_k) + \sum_{i=1}^{N} 1_{y_i \in B_k}\right)
\right)
$$

(2.2)

for any partition $(B_1, \ldots, B_k)$, where $1_{y_i \in B_k}$ denotes the indicator that draw $y_i$ fell into bin $k$. The posterior in Eq. 2.2 is once again a Dirichlet process, often denoted

$$
G \mid y_1, \ldots, y_N \sim \text{DP}
\left(
\alpha + N, \frac{\alpha}{\alpha + N} \cdot P_0 + \frac{N}{\alpha + N} \cdot \sum_{i=1}^{N} \delta_{y_i}
\right)
$$

(2.3)

where the posterior base distribution is a weighted mean of the prior base distribution $P_0$ and a collection of point masses at the observed $\{y_i\}_{i=1}^N$.

The role of $\alpha$ as a concentration parameter is critical in dictating the influence of the prior. As in the case of the discrete Dirichlet distribution, as $\alpha \rightarrow 0$, the prior
becomes uninformative and the posterior is dominated by point masses at the observed data points. When $\alpha \to \infty$, the prior dominates, and draws of discrete distributions $G$ from the DP will concentrate around the base distribution $P_0$.

2.2.1 Alternative Constructions

In using finite partitions $(B_1, \ldots, B_k)$, the above constructions fail to intuitively capture the nature of a draw from the DP. We will explore two additional constructions or representations of the DP data generating process, both of which we will later employ in modeling the multi-task mHealth setting. Note that all constructions of the DP presented in this chapter are equivalent, but each serves a distinct role in highlighting specific properties of the stochastic process.

**Chinese Restaurant Process.** Proposed in 1983, the Chinese Restaurant Process (CRP) considers the analogy of a restaurant with an infinite number of tables, $i \in \{1, 2, \ldots\}$, each with infinite capacity. [8] Customers arrive one by one; the first customer is seated at the first table. Each subsequent customer either sits at
a new table, with probability proportional to parameter \( \alpha \), or at an existing table, with a preference for more heavily occupied tables.

We use \( z_i \) to represent the table at which the \( i \)th customer is seated; for instance, \( z_1 = 1 \). Let \( j \) denote the number of non-empty tables (i.e., seating at least one customer), when the \( i \)th customer arrives. Then:

\[
P(z_i = k \mid z_{-i}, \alpha) = \begin{cases} \frac{n_k}{i-1+\alpha} & \text{for } k \leq j \\ \frac{\alpha}{i-1+\alpha} & \text{for } k = j + 1 \end{cases}
\]  

(2.4)

where \( n_k \) is the number of customers currently seated at table \( j \), \( z_{-i} \) denotes the table assignments for all customers currently seated, and \( j + 1 \) is a new table.

Eq. 2.4 formalizes \( \text{CRP}(\alpha, N) \), the Chinese Restaurant Process with \( N \) total customers and concentration parameter \( \alpha \).

To fully induce a draw from the DP, we combine the CRP with one additional step. Each table \( k \) draws a single \( \theta^{(k)} \), independent of other tables, from the base distribution \( P_0 \). The complete mechanism of a draw from the DP is then expressed in terms of the CRP as follows:

\[
\begin{align*}
\{ z_i \}_{i=1}^N & \sim \text{CRP}(\alpha, N) \\
\theta^{(k)} & \sim P_0 \quad \text{i.i.d. } \forall k \in \{ z_1, \ldots, z_N \}
\end{align*}
\]

(2.5)

The collection of \( \{ \theta^{(k)} : k = z_i \}_{i=1}^N \) is a draw of \( N \) observations from \( \text{DP}(\alpha, P_0) \). In other words, each customer reports the \( \theta^{(k)} \) on their table, yielding \( N \) observations from the Dirichlet process.

This construction highlights two key features of the DP. First, it separates the draws of clusters or groups \( z_i \) from the draws of atoms \( \theta^{(k)} \) that each cluster adopts. Next, the CRP demonstrates the adaptive nature of this nonparametric model: the number of tables is free to grow with the number of customers. The CRP hence provides intuition for the role of \( \alpha \) in Fig. 2.1; for a fixed \( N \), a higher \( \alpha \) is likely to yield more tables and, hence, more unique draws of \( \theta^{(k)} \) from the base distribution \( P_0 \). Conversely, a small \( \alpha \) likely constrains the number of tables, hence sampling a small number of points repeatedly from \( P_0 \) (that is, since every customer at each of the few tables reports the same \( \theta^{(k)} \)).
**Stick-Breaking Construction.** Another equivalent representation uses a stick-breaking analogy [38] to induce $G \sim \text{DP}(\alpha, P_0)$ with the following generative process:

$$
\begin{align*}
V_k & \sim \text{Beta}(1, \alpha) \\
\pi_k &= V_k \prod_{i<k}(1 - V_i) \\
\theta^{(k)} & \sim P_0 \\
G(x) &= \sum_{k=1}^{\infty} \pi_k \delta_{\theta^{(k)}}(x)
\end{align*}
$$

(2.6)

Note that $\delta_{\theta^{(k)}}(x)$ is a degenerate distribution with all mass at $\theta^{(k)}$.

We begin with a stick of length 1, representing a total probability to be allocated. We break the first fragment with length $V_1 \sim \text{Beta}(1, \alpha)$, which becomes the first probability $\pi_1$. We draw the first “atom” $\theta^{(1)}$ from the base distribution $P_0$; to it we assign probability $\pi_1$.

Then, we break off a fraction $V_2 \sim \text{Beta}(1, \alpha)$ of the remaining stick fragment. We assign the length of this fragment, $V_2(1 - V_1)$, to $\pi_2$, draw the second “atom” $\theta^{(2)}$ from the base distribution $P_0$, and assign this atom probability $\pi_2$. We repeat ad infinitum. The discrete distribution $G$ we have sampled, consisting of “atoms” and their respective probabilities, is equivalently a draw from $\text{DP}(\alpha, P_0)$.

### 2.2.2 Dirichlet Process Mixture Models (DPMM)

We have now seen realizations of, or draws from, the Dirichlet process as discrete distributions of atoms originating from some base distribution $P_0$. In the case of a Dirichlet process mixture model (DPMM), these atoms in turn enter the distribution of another random quantity. [107] A simple general case is:

$$
\begin{align*}
\{z_i\}_{i=1}^{N} & \sim \text{CRP}(\alpha, N) \quad \forall i \in \{1, \ldots, N\} \\
\theta^{(k)} & \sim P_0 \quad \text{i.i.d.} \quad \forall k \in \{z_1, \ldots, z_N\} \\
y_i | z_i = k, \theta^{(k)} & \sim f(y_i | \theta^{(k)}) \quad \forall i \in \{1, \ldots, N\}
\end{align*}
$$

(2.7)

where $f$ is some valid specified probability mass or density function.

The graphical model 2.2 illustrates the DPMM in Eq. 2.7. Each observed $y_i$
Figure 2.2: A generic Dirichlet Process mixture model with $N$ observations

originates, via $f$, from the parameter $\theta^{(k)}$ corresponding to its true latent mixture component $z_i = k$.

Critically, the number of mixture components (i.e., unique $z_i$) is unspecified in the model; mixture components are instead drawn from the CRP. Hence, the DPMM is an infinite mixture model, unlike more traditional finite mixtures in which $K$ components must be specified a priori. [29, 23]

As mixture components can be conceptualized as “clusters” among observations, the DPMM has gained much popularity in the model-based clustering literature, particularly in settings where pre-specifying the number of clusters is difficult and/or subjective. [63] Applications span document modeling and clustering [131, 118, 133], manufacturing process control [67], epidemiological time series clustering [93], and early efforts in offline multi-task learning for classification.

### 2.3 Inference

While the DPMM can flexibly model a variety of probabilistic phenomena involving clusters or mixtures, classical inference methods on the DPMM (such as a full closed-form posterior) are generally intractable. While efforts have been made to adapt classical algorithms such as expectation-maximization (EM) to the DPMM [53], these invariably involve approximations and seek point estimates rather than full posterior distributions. Hence, the DPMM literature generally draws on Markov Chain Monte Carlo (MCMC) and variational inference.
2.3.1 MCMC Methods: Gibbs Sampling

A rich set of algorithms originating from the Metropolis-Hastings algorithm, MCMC methods sample from a target distribution of interest by constructing a Markov Chain whose stationary distribution is, by design, the target distribution. [19]

Among these methods, a *Gibbs sampler* draws from the joint distribution of $V$ random variables $q(X_1, ..., X_V)$ by iteratively sampling from each of the single-variable conditional distributions $q(X_v | X_{-v})$ for all $v \in \{1, ..., V\}$, where $X_{-v}$ denotes the set of all other variables $(X_1, ..., X_{v-1}, X_{v+1}, ..., X_V)$ excluding $X_v$. [37]

**Algorithm 2.1** General Gibbs Sampler

```plaintext
input
• number of samples $M$
• burn-in number $M_b$
• set of variables $(X_1, ..., X_V)$

for $m = 1, ..., M$ do
  for $v = 1, ..., V$ do
    sample $X_v$ from conditional $q(X_v | X_{-v})$
  discard burn-in samples $m = 1, ..., M_b$
```

Gibbs sampling methods in the literature for the DPMM employ the stick-breaking construction from 2.6. [79, 129] Although we model the DPMM using the Chinese Restaurant Process construction in 2.7, we may use this alternative construction for inference since the two are equivalent.

MCMC methods produce samples from target distribution by converging to said distribution as the equilibrium distribution of the Markov chain. This convergence usually occurs after some number of initial samples, hence we define a "burn-in" number $M_b$ of initial samples which we later discard.

Upon observing $\{y_i\}_{i=1}^N$, we are interested in learning:

- the number $K$ of mixture components (clusters) among the $N$ observations
- cluster assignments $\{z_i\}_{i=1}^N$
• cluster assignment probabilities $\pi_k \ \forall k$
• cluster parameters $\theta^{(k)} \ \forall k$

To simplify notation, let $z$, $\theta$, and $\pi$ denote the set of cluster assignments $\{z_i\}_{i=1}^N$, set of cluster parameters $\{\theta^{(k)}\}$, and stick-breaking parameters $\{\pi_k\}$ respectively.

In our case, the target distribution from which we wish to sample is the joint posterior $q(z, \theta, \pi | y)$.

The DPMM is an infinite mixture model; however, sampling from a posterior over an infinite number of mixture components (or a dynamic number of components) is computationally difficult. Therefore, we use a form of the blocked Gibbs sampler, in which we assume an upper bound $K_{\text{max}}$ on the number of cluster components.

[48] We use the blocked Gibbs sampler to iteratively sample from the conditionals of $z$, $\theta$, and $\pi$.

Additionally, the hyperparameter $\alpha$ is often difficult to tune and has significant influence on the number of clusters inferred, as in 2.2 [69]. Therefore, we seek to learn $\alpha$ from the data by setting a Gamma hyperprior $\alpha \sim \text{Gamma}(\gamma_1, \gamma_2)$, which we update via conjugacy. Algorithm 2.2 illustrates the full blocked Gibbs sampler for a DPMM.

In a practical setting, one may be interested, as we will soon be, in consolidating this posterior sample of size $M$ into a single set of cluster assignments $\{z_i\}_{i=1}^N$ across $N$ observations. The question of reducing the full posterior sample into a single estimate is non-trivial, and several approaches exist to achieve such a “consensus” or point estimate. We briefly explore two.

**Frobenius Norm Minimizer.** Suppose we have a series of $M - M_b$ posterior samples (the first $M_b$ are discarded as burn-in) for the vector of cluster assignments $\{z_i\}$. As cluster “labels” change throughout the process of the MCMC, we are interested not in the cluster labels assigned to each individual observation, but rather the co-occurrence of pairs of observations $i$ and $j$.

Take the first sample $m = M_b + 1$ of $\{z_i\}_{i=1}^N$. Using this sample, construct $\Omega_m$, an $N$-by-$N$ symmetric matrix for which $\Omega_m(i,j) = 1$ if $z_i = z_j$ and $\Omega_m(i,j) = 0$ otherwise. We call this the co-occurrence matrix. Similarly calculate the co-occurrence matrix for all $M$ samples of $\{z_i\}_{i=1}^N$. 

27
Algorithm 2.2 Blocked Gibbs Sampler for DPMM

input

- hyperparameters \( \{ \gamma_1, \gamma_2 \} \)
- probability densities \( \{ P_0, f \} \)
- observed data \( y = \{ y_i \}_{i=1}^N \)
- number of samples \( M \)
- burn-in number \( M_b \)
- max. number of clusters \( K_{\max} \)

fix base distribution \( P_0 \); hyperprior \( \alpha \sim \text{Gamma}(\gamma_1, \gamma_2) \)
sample \( \alpha \sim \text{Gamma}(\gamma_1, \gamma_2) \)
for \( k = 1, \ldots, (K_{\max} - 1) \) do
  sample \( V_k \sim \text{Beta}(1, \alpha) \)
  compute \( \pi_k = V_k \prod_{i=k}^{\infty} (1 - V_i) \)
  sample \( \theta^{(k)} \sim P_0 \)
  compute \( \pi_{K_{\max}} = 1 - \sum_{k'=1}^{K_{\max}-1} \pi_{k'} \)
  sample \( \theta^{(K_{\max})} \sim P_0 \)
for \( m = 1, \ldots, M \) do
  for \( i = 1, \ldots, N \) do
    sample \( z_i \) with Multinomial probabilities
    \[
    p(z_i = k) = \frac{\pi_k f(y_i | \theta^{(k)})}{\sum_{k'=1}^{K_{\max}} \pi_{k'} f(y_i | \theta^{(k')})} \quad k = 1, \ldots, K_{\max}
    \]
    count \( n_k = \sum_{i=1}^N 1_{z_i=k} \quad \forall k \in \{1, \ldots, K_{\max}\} \)
  for \( k = 1, \ldots, (K_{\max} - 1) \) do
    sample \( V_k \sim \text{Beta} \left( 1 + n_k, \alpha + \sum_{k'=k+1}^N n_{k'} \right) \)
    compute \( \pi_k = V_k \prod_{i=k}^{\infty} (1 - V_i) \)
    sample \( \theta^{(k)} \sim P_0 \cdot \prod_{i=1}^N \left( f(y_i | \theta^{(k)}) \right)^{1_{z_i=k}} \)
    compute \( \pi_{K_{\max}} = 1 - \sum_{k'=1}^{K_{\max}-1} \pi_{k'} \)
    sample \( \theta^{(K_{\max})} \sim P_0 \cdot \prod_{i=1}^N \left( f(y_i | \theta^{(K_{\max})}) \right)^{1_{z_i=K_{\max}}} \)
    sample \( \alpha \sim \text{Gamma}(\gamma_1 + K_{\max} - 1, \gamma_2 - \sum_{k'=1}^{K_{\max}-1} \log(1 - V_{k'})) \)
  store sample \( \rho_m = (\pi, z, \theta, \alpha) \), where \( \pi = \{ \pi_{k'} \}_{k'=1}^{K_{\max}} \), \( z = \{ z_i \}_{i=1}^N \), \( \theta = \{ \theta^{(k')} \}_{k'=1}^{K_{\max}} \)
  discard samples \( \{ \rho_1, \ldots, \rho_{M_b} \} \)

28
Then, we take the mean co-occurrence matrix

\[ \langle \Omega \rangle = (M - M_b)^{-1} \sum_{i=M_b+1}^{M} \Omega_m \]

Finally, we choose the sample co-occurrence matrix \( m^* \) that minimizes the Frobenius distance to \( \langle \Omega \rangle \), i.e.,

\[ m^* = \arg \min_m \| \Omega_m - \langle \Omega \rangle \|_F \]

where

\[ \| A_{(N \times N)} \|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} |A(i,j)|^2} \]

Then, \( \Omega_m \) is our estimate for the co-occurrence matrix across \( N \) observations. [23, 80]

**MAP estimate.** We similarly construct the mean co-occurrence matrix \( \langle \Omega \rangle \) as above. Each element \((i, j)\) of this symmetric mean matrix gives the proportion of samples from the posterior in which \( i \) and \( j \) are in the same cluster.

Then, for each element \((i, j)\) of \( \langle \Omega \rangle \), we round the element to the nearest integer. This has the effect of clustering \( i \) and \( j \) together \((\langle \Omega \rangle(i, j) = 1)\) if, according to the proportion of sample co-occurrences, they were more likely than not to be clustered together, and \( \langle \Omega \rangle(i, j) = 0 \) otherwise. This “more likely than not” principle gives the maximum *a posteriori* (MAP) estimate of clustering or co-occurrence, or \( \hat{\Omega}_{\text{MAP}} \). [120, 46]

### 2.3.2 Variational Methods

In contrast to MCMC, variational inference methods are deterministic. The computation of the posterior distribution over latent variables \( w \) having observed data \( x \) is reformulated as an optimization problem. The objective is to minimize the Kullback-Leibler (KL) divergence between the true (desired) posterior \( p(w|x) \) and its *variational* approximation \( q_v(w) \) characterized by some free (variational) parameters \( v \). [10]
The KL divergence is defined as
\[
\text{KL}(q_r(w) \mid p(w \mid x)) = \int \log \frac{q_r(w)}{p(w \mid x)} q_r(w) dw
\]
(2.8)
or equivalently
\[
\text{KL}(q_r(w)\|p(w \mid x)) = \mathbb{E}_q[\log q_r(w)] - \mathbb{E}_q[\log p(w, x)] + \log p(x)
\]
(2.9)
Minimizing Eq. 2.9 with respect to \( v \) is equivalent to maximizing the evidence lower bound (ELBO),
\[
\mathbb{E}_q[\log p(w, x)] - \mathbb{E}_q[\log q_r(w)]
\]
(2.10)

For posterior variational inference on the DPMM latent variables \( z, \theta, \pi \), given observed data \( y \), Blei and Jordan [9] write the ELBO
\[
\mathbb{E}_q[\log p(V)] + \mathbb{E}_q[\log p(\theta)] + \sum_{i=1}^{N} (\mathbb{E}_q[\log p(z_i \mid V)] + \mathbb{E}_q[\log p(y_i \mid z_i)]) - \mathbb{E}_q[\log q(V, \theta, z)]
\]
(2.11)
drawing on the stick-breaking construction like the blocked Gibbs sampler. Observe that we introduce \( V \) in place of \( \pi \). We know from the stick-breaking construction in Eq. 2.6 that \( V_k \sim \text{Beta}(1, \alpha) \) and that \( \pi_k = V_k \prod_{i=k} \pi_i \) for all \( k \); therefore, a distribution parameterized by \( \pi \) can be equivalently parameterized by \( V \), as Blei and Jordan do.

They propose the factorized form for the last (variational) term
\[
q(V, \theta, z) = \prod_{k=1}^{K_{\text{max}}-1} q_V(V_{k'}) \prod_{k'=1}^{K_{\text{max}}} q_\theta(\theta^{(k')}) \prod_{i=1}^{N} q_z(z_i)
\]
(2.12)
where \( q_V \) are Beta distributions for \( V \), \( q_\theta \) are exponential family distributions (in the family of \( P_\theta \)) for \( \theta \), and \( q_z \) are multinomial distributions for \( z \). Then, a straightforward coordinate ascent algorithm is used to optimize the factorized
components of the variational posterior in their respective free parameters. Note that the present work will employ MCMC – in particular, the Blocked Gibbs Sampler – as has been common in the literature on Dirichlet Process mixtures, as well as for simplicity in avoiding variational approximations. However, we provide the above brief overview of variational methods for completeness.
Chapter 3

Models, Methods, and Results

We now return to the mobile health (mHealth) setting encountered in Chapter 1, with $N$ users, each labelled $i \in \{1, 2, ..., N\}$. We wish to apply the framework of the Dirichlet Process mixture model to aid us in treating users optimally. In this often noisy, limited data setting, recall that the process of learning an optimal treatment policy for a particular user $i$ is often prohibitively slow. Therefore, our objective is to harness data from other users in an effective and responsible manner (which we refer to as “pooling”) in order to speed this learning for each user $i$.

![Desired outcome of pooling methods (schematic)](image)

Figure 3.1: Desired outcome of pooling methods (schematic)

Figure 3.1 represents the average reward across all $N$ users over the $T = 100$
decision times that constitute the course of a hypothetical mobile health “study.” We are interested in comparing the performance of a pooling algorithm (in which data is shared across users in some principled fashion) to a powerful baseline such as Thompson Sampling, which operates individually on each user \( i \) using \( i \)’s own data alone. Notice that even in the mHealth setting, with high variance (noise) and infrequent data from each user, if we let an algorithm such as Thompson Sampling run for sufficiently long, it would converge to an optimal policy, hence achieving the highest possible reward across users. Therefore, we do not expect that pooling will help achieve a higher total reward (i.e., converge to a more optimal policy) in the long run than a no-pooling algorithm. However, we do seek an outcome in which pooling will accelerate the convergence to this optimal policy (or maximum possible reward). This implies, as in Figure 3.1, that at some finite time \( t \) (for \( t \) sufficiently small), pooling should have achieved higher reward on average than individualized Thompson Sampling – in other words, it is closer to the optimal treatment policy. This is of great significance in mHealth, wherein one can only follow a user for a finite, often brief, period of time, particularly since users are likely to disengage quickly if they do not observe results.

It is important to establish that the scenario illustrated in 3.1 cannot always be perfectly achieved. Pooling depends entirely on the inference task of learning which users are similar, in order to share data among them. Since any pooling or learning algorithm is stochastic, there are bound to be cases in which we incur some degree of error in inference. This error will propagate: incorrect inference will yield some pooling of data across the wrong set of users, resulting in the orange curve (pooling) in 3.1 leveling off slightly below the blue curve. One may view this as paying a small price in total reward (or the policy converged to) in the long run, in exchange for quicker learning and higher rewards early on in the study.

Broadly, we will use the DPMM to model the degree of similarity across users. Thereafter, our task will be to first perform inference on the DPMM in order to determine which users are “similar” and subsequently pool data among these users. We will assess the performance of pooling algorithms by examining the mean reward across \( N \) users over the course of a study.
3.1 Statistical Formulation

We are in the contextual bandit mHealth setting, with $N$ users followed over $T$ decision timesteps. Recall that in a contextual bandit setting, the agent observes a state or context at each timestep, selects an action from a set of arms or action space, and observes a reward. In our setting, the action space consists of two arms: action 1 (treat or intervene) and action 0 (do not treat or intervene). In the context of HeartSteps, our motivating mHealth example from Chapter 1: the treatment is a physical activity suggestion; the reward is measured by a user’s step count over the next 30 minutes; the context or state includes features such as temperature, the user’s prior 30-minute step count. [57, 66] Additionally, at each timestep, we record an indicator of whether each user is available to receive treatment.

This is a multi-task problem: learning how to treat each user optimally is a bandit “task,” and we have $N$ such problems. We aim to use pooling to leverage information across the bandit tasks in order to improve the performance of each.

3.1.1 Dirichlet Process

In order to design a pooling method, we formalize the similarity across users or tasks. We posit that the $N$ users come from $K$ distinct groups or “clusters,” where $K$ is not known or assumed beforehand. We use $z_i$ to denote the cluster that user $i$ belongs to.

We use the Chinese Restaurant Process (CRP) formulation of the Dirichlet Process (see Chapter 2) with concentration parameter $\alpha > 0$ to model the mechanism by which the cluster assignments $z_i$ are generated. Note that the CRP is equivalent to any other formulation of the Dirichlet Process. We use the DP because it does not require to assume the number of user clusters a priori, instead enabling us to learn the number of clusters from the data.

3.1.2 Cluster- and Individual-Specific Parameters

Cluster-Specific Parameters. Suppose $z_i = k$, where $1 \leq k \leq K$. Each user $i$ has a vector of treatment effect coefficients $\beta^{(k)}$ (which is shared identically across all users in $i$’s cluster $k$). We assume that the $\beta^{(k)}$ for all $k$ are drawn from the
base distribution \( P_0 \) of the Dirichlet Process; that is, users are first assigned to clusters, and then a treatment effect \( \beta^{(k)} \) is drawn for each cluster \( k \in \{1, \ldots, K\} \). The treatment effects \( \beta \) are unknown and must be learned by the agent.

Note of caution: we will very occasionally, for the sake of concreteness, switch between referring to \( \beta_i \) and \( \beta^{(k)} \). The former is used to refer to a particular user’s treatment effect, while the latter refers to a treatment effect shared by all users in cluster \( k \). Our model assumes that users in a cluster all share a treatment effect, i.e., \( \beta_i = \beta_j \) if and only if \( z_i = z_j \).

**Individual-Specific Parameters.** Each user \( i \) has a vector of individual baseline reward coefficients \( \omega_i \). This vector is also unknown and must be learned along with treatment effects \( \beta \).

### 3.1.3 Reward Model

At each timestep \( t \), each user \( i \)

- has a context (state) modeled by a \( p \)-dimensional vector \( S_{i,t} \), as well as an indicator \( I_{i,t} \) encoding whether they are available for treatment,
- if \( I_{i,t} = 1 \), is assigned a binary action \( A_{i,t} \), corresponding either to treatment \( (A_{i,t} = 1) \) or no treatment \( (A_{i,t} = 0) \) according to the current policy \( \Pi_{i,t} \),
- receives a reward \( R_{i,t} \).

We use the following linear model for the reward:

\[
R_{i,t} = g(S_{i,t})^\top \omega_i + A_{i,t} \cdot f(S_{i,t})^\top \beta^{(k)} + \varepsilon_{i,t}
\]

where \( k = z_i \), i.e., user \( i \) belongs to cluster \( k \), and the noise term \( \varepsilon_{i,t} \sim \mathcal{N}(0, \sigma^2) \). Note that we take the noise terms to be independent and identically distributed (i.i.d.) across all users \( i \) and all timesteps \( t \). We assume \( \sigma^2 \) to be known for simplicity; this is justifiable in that users are often followed for some “exploration period” without the use of an intervention, during which the baseline noise in each user’s rewards may be estimated. Still, we test sensitivity to this assumption in 3.3.7.
The linear model for reward allows for simplicity as well as robust, interpretable Bayesian Gaussian linear regression in learning the treatment effect parameters \( \{\beta^{(k)}\} \); hence, the linear model is common in the contextual bandit literature. \[20\]

This construction gives a hierarchical Dirichlet Process mixture model: users are assigned to their clusters via the Dirichlet Process; each cluster shares a common \( \beta^{(k)} \) parameter, which gives rise to the rewards \( \{R_{i,t}\} \) we observe.

Figure 3.2 depicts this model. Shaded nodes denote observed variables (including state, action, and reward); unshaded nodes represent unobserved or latent variables (including cluster membership \( z_i \) and true treatment effect \( \beta^{(k)} \)). Parameters
enclosed in circles are continuous; those enclosed in squares are discrete; terms in diamond shapes are distributions; unenclosed parameters are fixed or known parameters.

One final assumption is that of stationarity: we assume any parameters pictured outside the \( T \) plate (i.e., not subscripted by \( t \)) in 3.2 do not change over time.

Mathematically, we describe the hierarchical model as:

\[
\begin{align*}
\{z_i\}_{i=1}^N &\sim \text{CRP}(\alpha, N) \quad \forall i \in \{1, \ldots, N\} \quad z_i \in \{1, \ldots, K\} \\
\beta^{(k)} &\sim P_0 \quad \forall k \in \{1, \ldots, K\} \\
R_{i,t} \mid z_i = k, \beta^{(k)}, A_{i,t}, S_{i,t} &\sim \mathcal{N}\left(\langle g(S_{i,t})\rangle^T \omega_i + A_{i,t} \cdot f(S_{i,t})^T \beta^{(k)}, \sigma^2\right) \forall t \in \{1, \ldots, T\}
\end{align*}
\]

(3.1)

The goal of the agent is to maximize the user’s reward \( R_{i,t} \) at each \( t \) by selecting the optimal action \( A_{i,t} \) (in other words, utilizing the optimal policy). As the next section will elucidate, the agent’s policy for action selection depends entirely on its learning of treatment effects \( \beta^{(k)} \). Therefore, to speed this learning of the \( \{\beta^{(k)}\} \), we aim to infer the underlying cluster membership \( \{z_i\} \) among users and share data across users within the same cluster. The model in Eq. 3.1 justifies this: users who share the same \( \beta^{(k)} \) also share the same generative model for reward \( R_{i,t} \), so all reward data from these users is useful to infer the underlying \( \beta^{(k)} \).

### 3.2 Algorithms

Having established a model for the environment in which the agent operates, the question remains: what learning algorithm can the agent follow in order to achieve the optimal action selection policy \( \Pi^* \) or maximize rewards \( \{R_{i,t}\} \)? Specifically, how do we incorporate pooling into such a learning algorithm?

#### 3.2.1 Baseline: Individual RL

As a baseline for comparison, we use INDIVIDUAL (Algorithm 3.1), a Thompson Sampling algorithm that does not pool across users. In this setting, no clusters are assumed, so each user has their own \( \beta_i \) instead of \( \beta^{(k)} \) shared within a cluster. The task of the learner is to learn \( \beta_i \forall i \) and use this to maximize reward \( R_{i,t} \) at every
timestep, with some allowance for exploration arising from the stochasticity of Thompson Sampling. As seen in Chapter 1, stochasticity comes from posterior uncertainty at time \( t \) regarding the true value of the treatment effect \( \beta_i \).

We adopt a Bayesian approach to learning each \( \beta_i \) via Bayesian linear regression. We begin with a Gaussian prior, \( \beta_i \sim \mathcal{N}(0, \Sigma_\beta) \), \( \forall i \). We take \( \Sigma_\beta \) to be the diagonal matrix \( \frac{1}{2} \mathbf{I} \) (parameterized by a single \( \frac{1}{2} \)). At each time \( t \), we update the posterior distribution for each \( \beta_i \) (via Eq. 3.2 below), which we denote \( \mathcal{N}(\mu_{i,t}, \Sigma_{i,t}) \).

Until some defined timestep \( T_{pool} \), we use pure exploration, i.e., select action \( A_{i,t} = 1 \) with some constant probability \( p_e \). For the purposes of simulation, we set \( p_e = 0.5 \), such that on average, users receive treatment once every two decision times. Note of caution: the subscript “pool” is not meant to suggest pooling in this context; we use it for the sake of consistency with our proposed algorithm in 3.2.2, in which we will also employ pure exploration until the timestep at which we perform pooling.

### 3.2.2 Proposed Algorithm

We now propose an algorithm, DPMM-Pooling (Algorithm 3.3) to introduce pooling within the Thompson Sampling framework above.

Recall that we use a blocked Gibbs Sampler (see Chapter 2) to perform inference on a Dirichlet Process mixture model, which generates an MCMC sample approximating the posterior distribution over cluster membership for a user \( i \), given all the observed data \( \{R_{i,t}\}_{t=1}^T \) thus far. Our blocked Gibbs Sampler applied to the mHealth DPMM, which we use to perform cluster inference for DPMM-Pooling, is summarized in Algorithm 3.2.

**Walkthrough of Algorithm 3.2.** The blocked Gibbs Sampler first iteratively samples cluster-specific probabilities \( \pi_k \) arising from \( V_k \) in the stick-breaking construction, for all clusters \( k = 1, ..., K_{\max} \). Recall that \( K_{\max} \) is our assumed upper bound (“block”) on the number of clusters. We then sample \( M = 1000 \) times from the joint posterior over three sets of parameters: \( V_k, \beta^{(k)} \) (for all \( k \)) and \( z_i \) (for all \( i \)). As Algorithm 3.2 is a Gibbs sampler, it iteratively samples cluster assignments \( z_i \), followed by cluster parameters \( V_k \) and \( \beta^{(k)} \). Specifically,
Algorithm 3.1 INDIVIDUAL

input

- $b$, the dimension of $\beta$ and $f(S_{i,t})$
- $w$, the dimension of $\omega$ and $g(S_{i,t})$
- $s^2_\beta$, the prior variance of $\beta$
- time horizon $T$
- number of users $N$
- $\sigma^2$, the measure of noise
- $T_{\text{pool}}$, the time $T_{\text{pool}}$ at which we stop pure exploration
- probability $p_\alpha$ of selecting $A_{i,t} = 1$ in exploration period

set Gaussian prior on treatment effects: $\beta_i \sim \mathcal{N}(0_b, \Sigma_\beta = s^2_\beta I_b)$, $\forall i$
set Gaussian prior on baseline reward coefficients: $\omega_i \sim \mathcal{N}(0_w, \Sigma_\omega = I_w)$, $\forall i$

initialize $\mu_{i,1} \leftarrow 0_b$, $\Sigma_{i,1} \leftarrow \Sigma_\beta$, $\hat{\Sigma}_{i,1} \leftarrow 0_{b \times w}$, $\bar{\Sigma}_{i,1} = \left( \begin{array}{c} \Sigma_\omega \\ 0_{w \times w} \end{array} \right)$, $\forall i$

for $t = 1, \ldots, T$

for $i = 1, \ldots, N$

observe user $i$’s context $S_{i,t}$ and availability for treatment $I_{i,t}$

if $I_{i,t} = 1$ then

if $T < T_{\text{pool}}$ then

sample $A_{i,t} = 1$ with probability $p_\alpha$

else

sample $\beta_i$ from current marginal posterior, i.e. $\beta_i \sim \mathcal{N}(\mu_{i,t}, \Sigma_{i,t})$

set $A_{i,t} = 1$ if $f(S_{i,t})^\top \beta_i > 0$ and $A_{i,t} = 0$ otherwise

if $A_{i,t} = 1$ then

deliver treatment/intervention to user $i$

else

do nothing

else

do nothing

observe reward $R_{i,t}$

concatenate $\phi(S_{i,t}) = (g(S_{i,t}), f(S_{i,t}))$

update joint posterior for $\omega_i, \beta_i$ given $R_{i,t}$ via:

\[
\begin{aligned}
\bar{\Sigma}_{i,t+1} &\leftarrow \left( \bar{\Sigma}_{i,t}^{-1} + \frac{1}{\sigma^2} \sum_{t' = 1}^{t} I_{i,t'} \cdot \phi(S_{i,t'}) \phi(S_{i,t'})^\top \right)^{-1} \\
\bar{\mu}_{i,t+1} &\leftarrow \bar{\Sigma}_{i,t} \left( \bar{\Sigma}_{i,t}^{-1} \bar{\mu}_{i,1} + \frac{1}{\sigma^2} \sum_{t' = 1}^{t} I_{i,t'} \cdot R_{i,t'} \cdot \phi(S_{i,t'}) \right)
\end{aligned}
\]

(3.2)

set $\mu_{i,t+1}$ to the last $b$ elements of $\bar{\mu}_{i,t+1}$

set $\Sigma_{i,t+1}$ to the bottom-right corner matrix of size $b \times b$ in $\bar{\Sigma}_{i,t+1}$
Each user’s cluster assignment $z_i$ is iteratively sampled from a Multinomial distribution, given and holding fixed the current samples of $\pi_k$ and $\beta^{(k)}$.

Each potential cluster’s treatment effect $\beta^{(k)}$ is sampled from a Gaussian posterior. Since a Gaussian base distribution $P_0$ is the prior for $\beta^{(k)}$, and observed rewards are Gaussian given $\beta$, the posterior for $\beta^{(k)}$ is also always Gaussian for all $k$.

The parameter $V_k$ is sampled from a Beta distribution as per the stick-breaking construction. Then, $\pi_k$ is computed as a deterministic function of $V_k$ following this construction.

Finally, the concentration parameter $\alpha$ is sampled from a Gamma distribution. We begin with a Gamma hyperprior $\alpha \sim \text{Gamma}(\gamma_1, \gamma_2)$ and iteratively update this “hyperposterior” via conjugacy.

Finally, the first $M_b = 400$ of burn-in samples are discarded, so as to ensure that all MCMC samples used in subsequent stages of analysis originate from the desired posterior distribution, i.e., after the MCMC has stabilized to sample from its equilibrium distribution.

In this case, following from the hierarchical model in Eq. 3.1, the users’ treatment effect coefficients $\beta$ are assumed to be equivalent to one of many cluster coefficients $\beta^{(k)}$. Therefore, we cluster the current posterior mean estimates for each user’s $\beta$ coefficients at time $T_{pool}$ using the blocked Gibbs sampler. This posterior mean for each user is denoted $\hat{\beta}_i$ in Algorithm 3.2.

Recall also from Chapter 2 the several methods used to achieve a single set of “consensus” cluster assignments from an MCMC sample. We use the maximum a posteriori (MAP) estimate for the co-occurrence matrix $\Omega$. As seen in Chapter 2, this estimator is useful in that it follows directly from the model and is hence statistically intuitive, as opposed to other methods such as the Frobenius norm minimizer. This explains the last line of Algorithm 3.2.

**Walkthrough of Algorithm 3.3.** In DPMM-Pooling, we act under pure exploration (each user receives treatment with probability 0.5 at every decision time), as in Algorithm 3.1. We explore until a specified timestep $T_{pool}$, at which point we infer cluster membership across users. After $T_{pool}$, we proceed with “pooled”
Thompson sampling, i.e., using a variant of Algorithm 3.1 in which all data within a user’s cluster $k$ is used to (1) compute the posterior for its shared $\beta^{(k)}$ and (2) select actions for the user.
Algorithm 3.2 Proposed Blocked Gibbs Sampler for mHealth

\begin{algorithm}
\textbf{input}
\begin{itemize}
  \item $\{\hat{\beta}_i\}_{i=1}^{N}$ and $\{\hat{\Sigma}_i\}_{i=1}^{N}$, the posterior mean and covariance matrix of user $i$'s $\beta$ coefficients at $t = T_{\text{pool}}$
  \item base distribution $P_0 = \mathcal{N}(0, \Sigma_{\phi})$ from Algorithm 3.3
  \item hyperparameters $\gamma_1, \gamma_2$
  \item number of samples $M$
  \item burn-in number $M_b$
  \item max. number of clusters $K_{\text{max}}$
  \item number of users $N$
\end{itemize}

\textbf{fix} base distribution $P_0$; hyperprior $\alpha \sim \text{Gamma}(\gamma_1, \gamma_2)$
\textbf{sample} $\alpha \sim \text{Gamma}(\gamma_1, \gamma_2)$
\textbf{for} $k = 1, \ldots, (K_{\text{max}} - 1)$ \textbf{do}
\begin{itemize}
  \item sample $V_k \sim \text{Beta}(1, \alpha)$
  \item compute $\pi_k = V_k \prod_{l=k}^{1}(1 - V_l)$
  \item sample $\beta^{(k)} \sim P_0$
  \item compute $\pi_{K_{\text{max}}} = 1 - \sum_{k'=1}^{K_{\text{max}}-1} \pi_{k'}$
  \item sample $\beta^{(K_{\text{max}})} \sim P_0$
\end{itemize}
\textbf{for} $m = 1, \ldots, M$ \textbf{do}
\begin{itemize}
  \item \textbf{for} $i = 1, \ldots, N$ \textbf{do}
  \begin{itemize}
    \item define density $f_{i,k} = \mathcal{N}(\beta^{(k)}, \hat{\Sigma}_i)$
    \item sample $z_i$ with Multinomial probabilities
    \begin{equation}
    \mathbb{P}(z_i = k) = \frac{\pi_k \cdot f_{i,k}(\hat{\beta}_i | \beta^{(k)}, \hat{\Sigma}_i)}{\sum_{k'=1}^{K_{\text{max}}} \pi_{k'} \cdot f_{i,k'}(\hat{\beta}_i | \beta^{(k')}, \hat{\Sigma}_i)} \quad k = 1, \ldots, K_{\text{max}}
    \end{equation}
  \end{itemize}
  \item count $n_k = \sum_{i=1}^{N} 1_{z_i=k} \forall k \in \{1, \ldots, K_{\text{max}}\}$
  \item \textbf{for} $k = 1, \ldots, (K_{\text{max}} - 1)$ \textbf{do}
  \begin{itemize}
    \item sample $V_k \sim \text{Beta}(1 + n_k, \alpha + \sum_{k'=k+1}^{N} n_{k'})$
    \item compute $\pi_k = V_k \prod_{l=k}^{1}(1 - V_l)$
    \item sample $\beta^{(k)} \sim P_0 \cdot \prod_{i=1}^{N} (f_{i,k}\hat{\beta}_i | \beta^{(k)}, \hat{\Sigma}_i)^{1_{z_i=k}}$
    \item compute $\pi_{K_{\text{max}}} = 1 - \sum_{k'=1}^{K_{\text{max}}-1} \pi_{k'}$
    \item sample $\beta^{(K_{\text{max}})} \sim P_0 \cdot \prod_{i=1}^{N} (f_{i,K_{\text{max}}}\hat{\beta}_i | \beta^{(K_{\text{max}})}, \hat{\Sigma}_i)^{1_{z_i=K_{\text{max}}}}$
    \item sample $\alpha \sim \text{Gamma}(\gamma_1 + K_{\text{max}} - 1, \gamma_2 - \sum_{k'=1}^{K_{\text{max}}-1} \log(1 - V_{k'}))$
  \end{itemize}
  \item store sample $\rho_m = (\pi, z, \beta, \alpha)$, where $\pi = \{\pi_{k'}\}_{k'=1}^{K_{\text{max}}}, z = \{z_i\}_{i=1}^{N}, \beta = \{\beta^{(k')}\}_{k'=1}^{K_{\text{max}}}$
  \item discard samples $\{\rho_1, \ldots, \rho_{M_b}\}$
\end{itemize}
\textbf{return} MAP estimate $\hat{\Omega}_{\text{MAP}}$ of co-occurrence as outlined in 2.3.2
\end{algorithm}
3.3 Experiments

We now turn to simulation in order to evaluate the performance of our algorithm, DPMM-Pooling (Algorithm 3.3) in comparison to INDIVIDUAL (Algorithm 3.1). We also compare to Oracle Pooling, an algorithm identical to DPMM-Pooling except in that the true clusters are assumed to be known to the agent, i.e., clusters need not be inferred. This provides a performance ceiling for DPMM-Pooling (represents the best possible performance with pooling if we incurred no error in inferring clusters among users).

3.3.1 Generative Environment

We simulate $N = 100$ users in $K^*$ clusters for $T = 150$ decision times. Users are divided equally among the $K^*$ clusters so as to maximize information within each cluster. For example, in our case with $N = 100$, when $K^* = 2$, each cluster consists of 50 users; if $K^* = 3$, the cluster memberships are $\{33, 33, 34\}$.

Each cluster is assigned a $b$-dimensional treatment effect $\beta^{(k)}$ (we take $b = 3$).

Each user is assigned a vector of baseline reward coefficients $\omega_i$ drawn from distribution $\mathcal{N}(\mathbf{0}_w, \mathbf{I}_w)$. We take $w = 7$.

At each timestep $t$, each user $i$ is assigned a state $S_{i,t}$, a vector of dimension $w + 1 = 6$. The $S_{i,t}$ are drawn i.i.d. from the multivariate normal distribution $\mathcal{N}(\mathbf{1}, \mathbf{I}_6)$. The agent observes this state and constructs

- a feature vector $f(S_{i,t})$ of dimension $b = 3$, simply consisting of the scalar quantity 1 (as a bias term for our Bayesian linear regression) followed by the last two elements of $S_{i,t}$.

- a feature vector $g(S_{i,t})$ of dimension $w = 7$, which consists of the scalar quantity 1 (as a bias term) followed by $S_{i,t}$ in its original form (i.e., includes all three elements of $f$ with four additional scalar elements).

With probability 0.5, the user $i$ is available for treatment at time $t$, i.e., $P(I_{i,t} = 1) = 0.5$. We choose 0.5 as data in mHealth is sparse due to limited user availability.

If the user is available ($I_{i,t} = 1$), the agent then follows its learning algorithm (either Algorithm 3.1 or 3.3) to select the action $A_{i,t}$. The agent observes reward $R_{i,t}$, which
Algorithm 3.3 DPMM-POOLING

input

- $b$, the dimension of $\beta$ and $f(S_{i,t})$
- $w$, the dimension of $\omega$ and $g(S_{i,t})$
- $s_0^2$, the prior variance of $\beta$
- time horizon $T$
- number of users $N$

- measure of noise $\sigma^2$
- time $T_{\text{pool}}$ at which we pool and stop pure exploration
- probability $p_o$ of selecting $A_{i,t} = 1$ in exploration period

set Gaussian prior on treatment effects: $\beta_i \sim \mathcal{N}(0_b, \Sigma_\beta = s_0^2 I_b), \ \forall i$

set Gaussian prior on baseline reward coefficients: $\omega_i \sim \mathcal{N}(0_w, \Sigma_\omega = I_w), \ \forall i$

initialize $\mu_{i,1} \leftarrow 0_b, \Sigma_{i,1} \leftarrow \Sigma_\beta, \bar{\mu}_{i,1} \leftarrow 0_{b \times w}, \bar{\Sigma}_{i,1} = \left( \frac{\Sigma_\omega}{0_{b \times w} \Sigma_\beta} \right) \forall i$

for $t = 1, \ldots, T_{\text{pool}}$ do

for $i = 1, \ldots, N$ do

<table>
<thead>
<tr>
<th>observe</th>
<th>user $i$’s context $S_{i,t}$ and availability for treatment $I_{i,t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>$I_{i,t} = 1$ then</td>
</tr>
<tr>
<td>sample</td>
<td>action $A_{i,t}$ from a Bernoulli distribution with parameter $p_o$</td>
</tr>
<tr>
<td>if</td>
<td>$A_{i,t} = 1$ then</td>
</tr>
<tr>
<td>deliver</td>
<td>treatment/intervention to user $i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>observe</th>
<th>reward $R_{i,t}$</th>
</tr>
</thead>
</table>

run Algorithm 3.2 to sample $M = 1000$ times from DPMM posterior, fixing a set of hard cluster assignments $\{z_i\}$ using MAP estimator $\hat{\Omega}_{\text{MAP}}$

initialize cluster-specific joint posteriors for each cluster $k$ at $t = T_{\text{pool}}$ via:

\[
\begin{align*}
\bar{\Sigma}_{k,t} & \leftarrow \left( \bar{\Sigma}_{k,t}^{-1} + \frac{1}{\sigma^2} \sum_{t'=1}^{t} \sum_{i:z_i=k} I_{i,t'} \cdot \phi(S_{i,t'}) \phi(S_{i,t'})^\top \right)^{-1} \\
\bar{\mu}_{k,t} & \leftarrow \bar{\Sigma}_{k,t} \left( \bar{\Sigma}_{k,t}^{-1} \bar{\mu}_{k,t} + \frac{1}{\sigma^2} \sum_{t'=1}^{t} \sum_{i:z_i=k} I_{i,t'} \cdot R_{i,t'} \cdot \phi(S_{i,t'}) \phi(S_{i,t'})^\top \right)
\end{align*}
\] (3.3)

set $\mu_{i,t+1}$ to the last $b$ elements of $\bar{\mu}_{i,t+1}$

set $\Sigma_{i,t+1}$ to the bottom-right corner matrix of size $b \times b$ in $\bar{\Sigma}_{i,t+1}$

for $t = T_{\text{pool}} + 1, \ldots, T$ do

for $i = 1, \ldots, N$ do

<table>
<thead>
<tr>
<th>observe</th>
<th>user $i$’s context $S_{i,t}$, availability $I_{i,t}$, and cluster assignment $k = z_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>$I_{i,t} = 1$ then</td>
</tr>
<tr>
<td>sample</td>
<td>$A_{i,t} = 1$ with probability $P(f(S_{i,t})^\top \beta^{(k)} &gt; 0)$ using current posterior, $\beta^{(k)} \sim \mathcal{N}(\mu_{k,t}, \Sigma_{k,t})$</td>
</tr>
<tr>
<td>if</td>
<td>$A_{i,t} = 1$ then</td>
</tr>
<tr>
<td>deliver</td>
<td>treatment/intervention to user $i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>observe</th>
<th>reward $R_{i,t}$</th>
</tr>
</thead>
</table>

update full posterior for each cluster $k$ given $\{R_{i,t}\}_{i=1}^N$ via Eq. 3.3
is drawn from $\mathcal{N}(g(S_{t,i})^\top \omega_i + A_{t,i} \cdot f(S_{t,i})^\top \beta^{(k)}, \sigma^2)$. Notice that this is identical to the reward model assumed by the agent, though we will investigate the performance of our learning algorithms under some degree of model misspecification.

We vary the following parameters across simulations:

- the noise in reward $\sigma^2$ (recall that this is known to the agent),
- the distance between $\beta^{(k)}$ between clusters,
- the time $T_{\text{pool}}$ at which cluster inference and pooling occurs,
- the true number of clusters $K^\ast$.

### 3.3.2 Evaluation Metrics

We average over $L = 100$ trials or experiments in each simulation in order to erase effects of noise $\sigma^2$. In each trial, we set a new random seed for the environment (generating rewards) as well as a new seed per individual user. Within a trial, the same set of $N = 100$ seeded users undergo INDIVIDUAL, Pooling, and Oracle Pooling. We evaluate DPMM-Pooling using the following three metrics for each simulation:

- **Percent Learning Acceleration**: This measures the extent to which pooling accelerates learning (or the reward trajectory). It is computed as the mean percentage improvement in reward (for DPMM-Pooling compared to INDIVIDUAL) over the first ten timesteps after $T_{\text{pool}}$, i.e.,

$$
\frac{1}{L} \sum_{l=1}^{L} \frac{1}{10} \sum_{t=T_{\text{pool}}+1}^{T_{\text{pool}}+10} \frac{1}{N} \sum_{i=1}^{N} \frac{R_{i,t,l}^{\text{DPMM-Pooling}} - R_{i,t,l}^{\text{INDIVIDUAL}}}{R_{i,t,l}^{\text{INDIVIDUAL}}}
$$

where $R_{i,t,l}^{\text{alg}}$ denotes the reward for user $i$ at time $t$ in trial $l$ under algorithm $\text{alg}$.

- **Percent Loss in Long-Run Reward**: With a pooling algorithm, we face a trade-off between accelerating the reward trajectory earlier in the trial and achieving high long-run reward by a pooling algorithm. This is expected due to some degree of incorrect pooling due to error in cluster inference. This metric measures said tradeoff with the percentage loss in reward (for
DPMM-Pooling compared to INDIVIDUAL) over the last 10 timesteps of the trial, i.e.,

\[
\frac{1}{L} \sum_{l=1}^{L} \frac{1}{10} \sum_{t=T-10}^{T} \frac{1}{N} \sum_{i=1}^{N} \frac{R_{l,t,l}^{\text{INDIVIDUAL}} - R_{l,t,l}^{\text{DPMM-Pooling}}}{R_{l,t,l}^{\text{INDIVIDUAL}}}
\]

- **Percent Loss versus Oracle**: Error in cluster inference can also be viewed as a loss in performance in comparison to Oracle Pooling, in which the agent knows the true cluster assignments and can pool data strictly within true clusters of users with perfect fidelity. Error with respect to the oracle is computed as the percentage loss in long-run reward (i.e., average reward over the last 10 timesteps) for DPMM-Pooling as compared to Oracle Pooling, i.e.,

\[
\frac{1}{L} \sum_{l=1}^{L} \frac{1}{10} \sum_{t=T-10}^{T} \frac{1}{N} \sum_{i=1}^{N} \frac{R_{l,t,l}^{\text{Oracle Pooling}} - R_{l,t,l}^{\text{DPMM-Pooling}}}{R_{l,t,l}^{\text{Oracle Pooling}}}
\]

### 3.3.3 Varying Noise (\(\sigma^2\))

Recall that pooling is motivated by high noise in a mobile health setting, which generates the need to leverage as many users’ data as possible in selecting actions for a particular user \(i\). We vary \(\sigma^2 = \{5, 25, 50, 100\}\) to evaluate the relative performance of pooling across noise levels in the environment.

We generate two clusters of 50 users each, one with \(\beta = [3, 3, 3]\) and the other with \(\beta = [-3, -3, -3]\). We hold fixed \(T_{\text{pool}} = 50\) and \(\sigma_0^2 = 1\).

Shaded regions in all figures represent 95% confidence intervals for the mean reward statistics depicted. Across all tables, evaluation metrics are reported as a mean \(\pm 1\) standard error (across \(L = 100\) trials).

From Figure 3.3 and Table 3.1, it is clear that pooling accelerates learning significantly more in high-noise settings. However, in exchange, we pay the price of lower long-run reward (compared to INDIVIDUAL and Oracle Pooling). Since cluster inference is likely difficult in high noise settings (where distinct clusters are less clearly discernible to the blocked Gibbs sampler), we incur this loss in
Table 3.1: Evaluation metrics across varying $\sigma^2$.

<table>
<thead>
<tr>
<th>Parameter $\sigma^2$</th>
<th>Metric</th>
<th>% Learning Acceleration</th>
<th>% Loss in Long-Run Reward</th>
<th>% Loss v. Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td></td>
<td>0.31 ± 0.11</td>
<td>0.03 ± 0.13</td>
<td>0.84 ± 0.09</td>
</tr>
<tr>
<td>25.0</td>
<td></td>
<td>2.63 ± 0.12</td>
<td>0.25 ± 0.08</td>
<td>1.40 ± 0.09</td>
</tr>
<tr>
<td>50.0</td>
<td></td>
<td>6.90 ± 0.14</td>
<td>0.37 ± 0.05</td>
<td>2.61 ± 0.08</td>
</tr>
<tr>
<td>100.0</td>
<td></td>
<td>12.02 ± 0.27</td>
<td>2.89 ± 0.09</td>
<td>4.50 ± 0.07</td>
</tr>
</tbody>
</table>

Figure 3.3: Varying noise level $\sigma^2$.

3.3.4 Varying Inter-Cluster $\beta$ Distance

We consider the impact of the distance between clusters’ $\beta$ parameters on the evaluation metrics of interest. We conjecture that increasing this distance makes
clusters more easily discernible to the DPMM-Pooling algorithm, thereby increasing the accuracy of cluster inference and pooling.

We hold $s_0^2 = 1$ and $T_{pool} = 50$ fixed.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-Cluster $\beta$ Distance</td>
<td>$\sigma^2$</td>
</tr>
<tr>
<td>{2, 2, 2}</td>
<td>25.0</td>
</tr>
<tr>
<td>{6, 6, 6}</td>
<td>25.0</td>
</tr>
<tr>
<td>{2, 2, 2}</td>
<td>50.0</td>
</tr>
<tr>
<td>{6, 6, 6}</td>
<td>50.0</td>
</tr>
</tbody>
</table>

Table 3.2: Evaluation metrics across varying inter-cluster $\beta$ distance.

The empirical relationship between the inter-cluster $\beta$ distance and our evaluation...
tion metrics is interesting, particularly with respect to $\sigma^2$. In both cases of $\sigma^2$, increasing the distance between the clusters’ $\beta$ parameters improves both learning acceleration and loss in long-run reward. This is intuitive as clusters become more easily discernible, so cluster inference is more accurate, and pooling provides useful data to accelerate learning on each user. Interestingly, for higher $\sigma^2$, the advantage of a larger inter-cluster $\beta$ distance increases. This can be explained in conjunction with 3.3.3; as pooling accelerates the reward trajectory more in higher noise environments, making clusters more easily discernible (by increasing the distance between their $\beta$ parameters) is likely to confer greater advantages in these high-noise environments.

### 3.3.5 Varying $T_{\text{pool}}$

The question of when to perform pooling is critical. We hypothesize that pooling too early may not leverage sufficient information, hence leading to poor cluster inference and performance, while pooling too late may not be beneficial as the INDIVIDUAL algorithm may have already sufficiently learned and achieved a near-optimal treatment policy.

We test $T_{\text{pool}} = \{20, 30, 50\}$ holding $\sigma^2 = 25.0$ and $s_0^2 = 1$ fixed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{pool}}$</td>
<td>% Learning Acceleration</td>
</tr>
<tr>
<td>20</td>
<td>3.47 ± 0.31</td>
</tr>
<tr>
<td>30</td>
<td>4.56 ± 0.25</td>
</tr>
<tr>
<td>50</td>
<td>2.62 ± 0.12</td>
</tr>
</tbody>
</table>

Table 3.3: Evaluation metrics across varying $T_{\text{pool}}$.

In the case of $T_{\text{pool}}$, the same tradeoff holds between accelerating learning early on and achieving high average reward in the long-run.

In terms of learning acceleration, we see that pooling too early ($T_{\text{pool}} = 20$) or too late ($T_{\text{pool}} = 50$) can both be detrimental. This is intuitive in that pooling too early produces inaccurate clustering estimates, resulting in lower rewards immediately as data is pooled across dissimilar users. On the other hand, pooling too late fails to
accelerate learning, as by the large $T_{\text{pool}}$, INDIVIDUAL has observed enough data to learn a reasonable policy. Therefore, there exists some optimal intermediate period in which pooling is most beneficial to accelerate learning.

In terms of long-run reward, waiting longer to pool can only reduce long-run...
loss, both with respect to INDIVIDUAL and Oracle Pooling.

### 3.3.6 Varying True Number of Clusters $K^*$

Recall that the results above all correspond to the case of $K^* = 2$ underlying clusters of users. We test the performance of DPMM-Pooling across settings with $K^* = \{1, 5, 10\}$ ground-truth clusters. In each case, $N = 100$ users are split randomly and uniformly across the $K^*$ clusters.

For each value of $K^*$, we average over $L = 100$ trials or experiments as we have done above. In each trial, we set a new random seed for the environment (generating both the $\beta^{(k)}$ parameter for each cluster $k$ and the observed rewards) as well as a new seed per individual user. Within a trial, the same set of $N = 100$ seeded users undergo INDIVIDUAL, Pooling, and Oracle Pooling.

We hold $\sigma^2 = 100.0$ and $T_{\text{pool}} = 20$ fixed in order to isolate the effects of $K^*$, although the same trends were verified to hold across settings of $\sigma^2$ and $T_{\text{pool}}$. We use $T_{\text{pool}} = 20$ as opposed to $T_{\text{pool}} = 50$ as in other sections because the effects of $K^*$ are most apparent in this case, when cluster inference and pooling are performed using only a few timesteps of observed data.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K^*$</td>
<td>$T_{\text{pool}}$</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 3.4: Evaluation metrics across varying $K^*$.

Figure 3.6 illustrates the performance of DPMM-Pooling across values of $K^*$. Evidently, in a high-noise setting, pooling is less beneficial as the number of ground-truth clusters increases. As the observed rewards generated by a large
Figure 3.6: Varying $K^*$. 

number of clusters are likely to overlap significantly in a high-noise setting, cluster inference is limited in its accuracy; therefore, pooling for a higher $K^*$ provides less learning acceleration and incurs greater loss in long-run reward.
However, Figure 3.7 (in which $T_{pool} = 30$ for $K^* = 10$) and the last line of Table 3.4 demonstrate that waiting longer to pool can address this problem posed by the existence of many ground-truth clusters. Increasing $T_{pool}$ is not a trivial fix, however, as it comes at the cost of lower learning acceleration, as seen in 3.3.5.

### 3.3.7 Investigating Reward Model Misspecification

Thus far, we have presented results under the assumption that the algorithm (DPMM-Pooling or INDIVIDUAL) is cognizant of the true underlying reward-generating model. We now test the performance of DPMM-Pooling under two specific cases of model misspecification.

For simplicity, we display all model misspecification results in the two-cluster
setting. The $K^* = 2$ clusters have $\beta = [-3, -3, -3]$ and $[3, 3, 3]$ respectively. Similar trends hold across a range of configurations for $K^*$ and $\beta$.

**Misspecified Noise ($\sigma^2$).** As discussed in 3.3.3, the degree of noise in the reward $\sigma^2$ is critical to the performance of pooling. Yet, in DPMM-Pooling, we assume this noise parameter is known to the algorithm. Figure 3.8 presents results for DPMM-Pooling when $\sigma^2$ is either under- or over-estimated by the algorithm, i.e., the value of $\sigma^2$ assumed by the algorithm does not match that in the simulation environment which generates rewards $R_{it}$. The left panels of Figure 3.8 are cases of correct $\sigma^2$ specification, while the right panels showcase misspecification.

![Figure 3.8: Cases of misspecified and correctly specified $\sigma^2$.](image)
Table 3.5: Evaluation metrics for cases of misspecified and correctly specified $\sigma^2$.

As in Figure 3.8, we notice that DPMM-Pooling provides less of an advantage in the case of misspecified $\sigma^2$ and, in some cases, incurs a larger loss in long-run reward. However, even under extreme misspecification of $\sigma^2$, pooling accelerates learning early in the course of the intervention.

**Misspecified DPMM.** Our particular hierarchical formulation of the DPMM in 3.1 and 3.2 assumes that every user within a cluster $k$ shares a single treatment effect parameter $\beta^{(k)}$. This is unlikely to match a real-world setting, in which users are highly heterogenous.

We test DPMM-Pooling under a simple violation of this assumption; we construct a parameter $\beta_i$ for each user, which is a noised version of the $\beta^{(k)}$ corresponding to the user’s cluster. Specifically, for each user, we take the shared parameter $\beta^{(k)}$ corresponding to the user’s true cluster and add i.i.d. Gaussian noise with mean $0_k$ and covariance matrix $\eta \cdot I_d$, where we vary the scalar degree of heterogeneity $\eta$ within a cluster.

We hold $\sigma^2 = 100$ and $T_{pool} = 50$ fixed.
Figure 3.9: Varying $\eta$: the degree of heterogeneity in user-specific ground-truth $\beta$ parameters within a single cluster.

Figure 3.9 illustrates that DPMM-Pooling is relatively robust to violations of the assumption that treatment effects are exactly shared among all users in a cluster. In simulated settings wherein users are noised versions of cluster centers (across
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Metric</th>
<th>% Learning Acceleration</th>
<th>% Loss in Long-Run Reward</th>
<th>% Loss v. Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>η</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td>6.63 ± 0.10</td>
<td>0.22 ± 0.09</td>
<td>0.72 ± 0.07</td>
</tr>
<tr>
<td>1.0</td>
<td></td>
<td>6.47 ± 0.10</td>
<td>0.07 ± 0.07</td>
<td>0.69 ± 0.12</td>
</tr>
<tr>
<td>3.0</td>
<td></td>
<td>5.66 ± 0.12</td>
<td>0.27 ± 0.09</td>
<td>0.62 ± 0.11</td>
</tr>
</tbody>
</table>

Table 3.6: Evaluation metrics for misspecified DPMM (within-cluster heterogeneity in β).

Varying degrees of this perturbation, controlled by parameter η, we see that DPMM-Pooling still accelerates learning, although this acceleration diminishes mildly with increasing heterogeneity. Perturbation at the η = 3 level is a severe case of misspecification, since this is comparable in magnitude to the β parameter itself; however, learning acceleration at the η = 3 level is still reasonably high with a small long-run loss in reward. Increasing heterogeneity does not appear to significantly influence loss in long-run reward.
Chapter 4

Discussion

The way in which RL algorithms are translated from simulated testbeds, as in the present work thus far, to real-world settings, such as clinical trials and interventions, is a non-trivial area of research crossing disciplinary boundaries. [34, 38, 71, 84, 48, 83] The mHealth setting in particular is characterized largely by sparse and noisy data; these key factors guide our interpretation of simulation results. [124, 80, 138]

In this final chapter, we connect our results for DPMM-Pooling to implications for mHealth interventions. We also suggest avenues for future work in pooling within sequential decision-making algorithms such as mHealth.

4.1 Conclusions and Implications

As we saw in Chapter 3, pooling (specifically DPMM-Pooling) is promising in a variety of settings, accelerating the average reward trajectory across users early in an intervention. We now analyze the implications of our results, including the performance sensitivity of DPMM-Pooling to model misspecification, which is likely in the real-world context of mHealth. Further, we examine DPMM-Pooling with a focus on ethical imperatives in the mHealth setting; for example, while pooling may improve the reward trajectory on average across users, what can be said about worst-case performance?

4.1.1 Pooling: A Bias-Variance Tradeoff

Chapter 3 reported the performance of DPMM-Pooling across a range of settings; specifically, assuming correct reward model specification, we varied noise in the
reward model $\sigma^2$, time of pooling $T_{\text{pool}}$, and distance between distinct clusters’ $\beta$ parameters. Our results point to, and can be summarized by, a classic bias-variance tradeoff.

In general, pooling is useful in settings characterized by particularly noisy and/or sparse data. Noise in observed rewards leads to high-variance estimates for each user’s treatment effect parameters $\beta$ and, consequently, high-variance treatment decisions (action selection). Pooling serves to reduce variance in our estimation of each user $i$’s treatment effect $\beta_i$. This is achieved by contributing additional observations originating from other users in the same cluster ($j \neq i, z_j = z_i$), thereby increasing the effective sample size of observed rewards for each user $i$. A more precise estimate of $\beta$ allows for lower-variance treatment decisions at each decision time, since the action selection policy hinges entirely on our estimate of $\beta$. Such a reduction in variance leads to an acceleration in the average reward trajectory across users (tasks) immediately after pooling.

In designing a pooling algorithm to exploit the benefits of reducing variance, we are likely to incur some bias in estimating $\beta_i$ for each user $i$. Errors in cluster inference lead to pooling data across users who are, in reality, dissimilar; using data from dissimilar users introduces bias in our estimation of user $i$’s treatment effect $\beta_i$. This bias leads to poor treatment decisions (action selection) and lower long-run reward as compared to an algorithm that does not pool across users.

The above bias-variance tradeoff encapsulates our simulation results across settings and parameters.

- Higher noise $\sigma^2$ in reward creates a greater need for pooling (to reduce variance); hence, in high-noise settings, we see an acceleration in the reward trajectory immediately after pooling. However, high noise in the observed rewards also leads to greater bias in cluster inference and a loss in long-run reward.

- Pooling early (setting $T_{\text{pool}}$ small) has the same effects as high noise $\sigma^2$.

- When the distance between distinct clusters’ $\beta$ parameters is small, we incur bias in cluster inference that is, as we have seen in Chapter 3, potentially large enough to entirely outweigh the benefits of pooling in terms of reduced
variance. Specifically, when clusters are closer together (less discernible), both learning acceleration and long-run reward are severely hindered for DPMM-Pooling.

For the practical task of designing a mHealth intervention, this central bias-variance tradeoff has many implications. First, it implies that one must deploy an algorithm such as DPMM-Pooling only if it is clear that (1) clusters are likely to exist among users and (2) clusters are relatively discernible from one another, without which we are likely to incur significant loss in immediate and long-run reward.

Furthermore, due to the influence of the parameter $T_{\text{pool}}$, there is much to gain from optimizing the time of pooling according to the level of noise $\sigma^2$ either known or expected in the reward model. Recall also from Chapter 3 that it is beneficial to optimize $T_{\text{pool}}$ according to the number of ground-truth clusters $K^*$ existing among the set of users. Since the DPMM nonparametric formulation allows us to avoid assuming $K^*$ a priori, such an optimization may be undesirable or difficult. Still, this is a useful note even if $K^*$ can be roughly predicted, for example, from past intervention data.

4.1.2 Model Misspecification in mHealth

In Chapter 3, we presented preliminary results for DPMM-Pooling under two cases of model misspecification: (1) the case of misspecified $\sigma^2$, wherein the algorithm (DPMM-Pooling and/or INDIVIDUAL) assumed a value of $\sigma^2$ that did not concur with ground truth; (2) the case of noised cluster centers, in which each user’s $\beta$ parameter is not exactly the shared parameter $\beta^{(k)}$ from its cluster, but rather a noised version of it.

We saw that in the case of extreme misspecification of $\sigma^2$, DPMM-Pooling still accelerates learning early in the course of an intervention, although less effectively than in the case of correctly specified $\sigma^2$. These results are encouraging, as in a real-world setting, (1) noise misspecification is unlikely to be this extreme, and (2) simple modifications can be made to DPMM-Pooling in order to learn $\sigma^2$ from the observed data.

Moreover, we saw that DPMM-Pooling is robust to violations of the assumption
that users within a cluster exactly share the same $\beta^{(k)}$; it performs well under cases where users in fact possess noised versions of their cluster-specific treatment effects $\beta^{(k)}$. This affords the intervention designer much freedom to deploy DPMM-Pooling in an mHealth setting, even if it is unreasonable to expect clusters of users sharing precisely equivalent treatment effects.

Some readers may connect the latter case of misspecification to the literature on mixed effects, in which users have underlying individual treatment effects in addition to shared population-level treatment effects. In mHealth, IntelligentPooling [137] explores pooling using a mixed effects model wherein users’ treatment effect parameters $\beta_i$ are seen as noised versions of a single population treatment effect. Our results show that DPMM-Pooling can perform under weaker assumptions, wherein users’ treatment effect parameters $\beta_i$ are noised versions of cluster-specific parameters, rather than a single population-level parameter.

Overall, DPMM-Pooling is robust to major types of model misspecification and therefore applies more generally (under weaker assumptions) than extant methods.

4.1.3 Ethical Considerations

In considering an application of DPMM-Pooling to a real-world JITAI or mHealth intervention, we consider four ethical questions that arise from our simulation results.

1. **Equipoise.** A principle commonly applied to medical trials, equipoise states that patients should be assigned to different arms (or actions/treatments) of a trial only if there exists genuine uncertainty over the optimality of any single arm or treatment option. [46] In our micro-randomized setting, this principle is followed at each decision time: no user is randomized to treatment if it is believed with certainty that treatment is not beneficial; similarly, no user is randomized to control if it is believed with certainty that the effect of treatment is positive. [45, 24] One can think of extending the principle of equipoise to the act of pooling, which helps speed the learning of a user’s treatment effect (and, therefore, the optimal treatment policy for the user).
We have seen in 4.1.1 that pooling can cause us to incur some bias in estimating a user $i$’s treatment effect $\beta_i$, resulting in a loss in long-run reward. Therefore, to avoid losses in long-run reward, pooling should only be performed if we are unlikely to obtain precise estimates of each user’s $\beta_i$ using a no-pooling algorithm such as INDIVIDUAL, and not otherwise. In summary, much like randomization, pooling should only take place if there exists significant uncertainty in our estimation of users’ treatment effects that may be resolved by sharing data across users.

2. **Efficacy.** Via simulation, we see a non-negative loss in long-run reward with the use of pooling, as compared to baseline algorithms that do not pool. We have further theoretically grounded this loss in a bias-variance tradeoff; in high-noise settings, pooling is beneficial in reducing the variance of treatment effect estimation and, consequently, the variance of treatment decisions. However, errors in cluster inference and pooling in these high-noise settings leads us to incur some bias and, consequently, loss in long-run reward. What follows is an imperative to minimize this loss in long-run reward as far as possible. In 4.2.1, we present preliminary ideas toward this goal.

3. **Privacy.** When pooling information across users, it is essential to anonymize data and obtain informed consent from all eligible users. [7, 31]

4. **Worst-case performance.** All results presented in Chapter 3 have been mean reward statistics and performance metrics, averaged across $L = 100$ experiments and $N = 100$ users. While shaded confidence intervals supplement graphical evidence that DPMM-Pooling indeed outperforms a baseline algorithm such as INDIVIDUAL across a variety of settings, it is still important to consider the worst-case effects of our proposed algorithm. Specifically, how does the algorithm perform for each individual user? While users benefit from pooling on average, do there exist users who are worse off as a result of pooling?

In Figure 4.1, we visualize the empirical distribution (across $N = 100$ users) of the mean reward improvement with DPMM-Pooling (as compared to INDIVIDUAL) over the last 10 timesteps of an intervention. This metric is
averaged across $L = 100$ trials. Users are then ordered by this mean reward improvement statistic for visualization purposes.

We hold $\sigma^2 = 25.0$, $K^* = 2$ and $T_{pool} = 50$ fixed. One cluster is assigned $\beta = [-3, -3, -3]$ while the other is assigned $\beta = [3, 3, 3]$. We choose this combination of parameters to represent a “worst-case” scenario; as seen in Figure 3.5 of Chapter 3, this set of parameters produces highly overlapping confidence intervals for the performance of INDIVIDUAL and DPMM-Pooling, calling into question the worst-case performance of DPMM-Pooling in this particular setting.

Figure 4.1: Empirical distribution of mean reward improvement with pooling for each of $N = 100$ users.

Figure 4.1 reveals that roughly 5% of users are somewhat worse off with DPMM-Pooling as opposed to INDIVIDUAL. The task of setting a maximum permissible percentage of users who are likely to be worse off poses a complex ethical dilemma beyond the scope of the present work. Regardless, this empirical distribution is tentatively encouraging. Note that Figure 4.1 represents simulation results and that a greater proportion of users could be left worse off in a real-world setting due to unforeseen factors.
4.2 Future Work

The present work presents a step towards model-based multi-task learning in sequential decision-making problems. The following avenues, many of which are motivated by the characteristic challenges of mHealth encountered in Chapter 1, demand further investigation.

4.2.1 Dynamic Pooling

So far, in the present work, we perform cluster inference across $N$ users at a single pre-specified time $T_{\text{pool}}$ and thereafter pool all data within each estimated cluster. We have observed certain limitations of this approach.

First, we have seen that fixing a time $T_{\text{pool}}$ is challenging, as pooling too early can result in poor cluster inference; on the other hand, in pooling too late, we forgo the advantages of pooling, i.e., the acceleration of learning that occurs soon after $T_{\text{pool}}$. One potential solution to this dilemma can be achieved by online or dynamic pooling, in which cluster inference is performed not only at time $T_{\text{pool}}$ but once at every interval $\Delta t$ thereafter.

This ensures that errors in cluster inference – resulting in data pooling among dissimilar users – which occur early on in the course of an intervention, when limited data is available, are continuously corrected. Key challenges anticipated with this approach include (1) computational burden, in that MCMC-based cluster inference may not be possible at frequent intervals in a real-world intervention; (2) the task of determining the pooling period $T_{\text{pool}}$ and the interval $\Delta t$ (i.e., we now have one additional tuning parameter.)

Online pooling could also effectively address the challenge of nonstationarity, i.e., the (likely) case in which user-specific treatment effects, or ground-truth clustering patterns across users, change over time. [63, 98, 28] Dynamic pooling over time could ensure that clustering patterns and treatment effects are precisely estimated as they evolve, and that old data is either erased or down-weighted, such that pooling and treatment decisions are not influenced by unrepresentative data from an old data-generating mechanism. The exact mechanism by which old data would be down-weighted remains a non-trivial question to be investigated.
Second, we concluded that, due to error in cluster inference, pooling data within estimated clusters can often lead to some loss in long-run reward, particularly in comparison to an individualized algorithm that does not pool data across users. Future work could address this challenge by adopting an approach in which the degree of pooling, or the weight assigned to data coming from other users \( j \neq i \) in making treatment decisions for user \( i \), diminishes over time. Such an algorithm could effectively harness the benefits of pooling, in accelerating the reward trajectory early in the course of an intervention, without compromising on long-run reward.

With regard to the second suggestion, readers familiar with the RL literature may recognize similarity to certain \( \varepsilon \)-greedy methods for exploration in which \( \varepsilon \) (which controls the probability of exploration) systematically approaches zero over time. [136, 85] A challenge here remains finding an optimal mechanism to systematically taper off pooling, such as by specifying a functional form for the decrease of a parameter (analogous to \( \varepsilon \) in \( \varepsilon \)-greedy) that dictates the degree of pooling.

4.2.2 Interpretability

While using data from users \( j \neq i \) to treat user \( i \) may be empirically useful, the decision to do so in high-stakes settings such as mHealth interventions can still be non-trivial, unless each pooling decision can be rigorously justified. [154] Specifically, those designing interventions would benefit from the ability to interpret clusters of users inferred by an algorithm such as DPMM-Pooling.

As a direct consequence of its Bayesian formulation, DPMM-Pooling offers one layer of interpretability; clusters can be understood as collections of users with high posterior probability of sharing the same treatment effect parameter \( \beta \).

In the context of mHealth interventions, further work may lend an additional layer of interpretability via a supervised learning approach to predict cluster assignments generated by DPMM-Pooling as a function of pre-treatment (baseline) covariates. Such a classifier could lend interpretability to pooling decisions, or even inform simple and intuitive pooling rules, such as: "pool all data within the cluster of all males over the age of 65 with a history of heart attacks." A key
challenge here remains the question of generalizability across interventions: a classifier trained on one intervention cohort, along with any resulting pooling rules, may not necessarily perform well on another cohort.

4.2.3 Social Network-Based Priors

Methods in multi-task RL surveyed in Chapter 2 (including the Gang of Bandits [17] and Horde of Bandits [141] frameworks) often employ graph or network structures to model similarity, and justify information transfer, across tasks (users). [106, 110] Further work could explore incorporating some information from social networks while retaining the Bayesian nonparametric approach of DPMM-Pooling. For example, an extension of this work could mathematize information from social networks (in which each user occupies a node) within the prior on cluster co-occurrence across $N$ users.

4.2.4 Delayed Effects of Interventions

As motivated in Chapter 1, learning an optimal user-specific treatment policy quickly is important partially because of rapid user disengagement, a characteristic challenge in mHealth. [139, 36] The fact that disengagement is to be expected in mHealth – alongside the widespread concern about treatment-induced cognitive burden in the literature [16] – implies that an intervention at decision time $t_1$ may generate delayed negative impacts for $t > t_1$. However, DPMM-Pooling does not consider negative delayed effects when selecting actions at each decision time.

The work on HeartSteps V2 attempts to model and account for such downstream negative impacts in the treatment policy. An extension of the present work may similarly model and account for the negative delayed effects of treatment.


[70] Mike, B. Hyperparameter estimation in dirichlet process mixture models.


[99] SHORTREEED, S. M., LABER, E., SCOTT STROUP, T., PINEAU, J., AND MURPHY, S. A. A multiple imputation strategy for sequential multiple assignment


