Latent Variable Models for Understanding User Behavior in Software Applications

by

Ardavan Saeedi

B.S., Sharif University of Technology (2007)
S.M., Sharif University of Technology (2009)
S.M., University of British Columbia (2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

Author ..... Signature redacted
Department of Electrical Engineering and Computer Science
December 1, 2017

Certified by ..... Signature redacted .....
Professor Joshua B. Tenenbaum
MIT, Department of Brain and Cognitive Sciences
Thesis Supervisor

Certified by ..... Signature redacted .....
Professor Ryan P. Adams
Princeton University, Department of Computer Science
Thesis Supervisor

Accepted by ..... Signature redacted
Professor Leslie A. Kolodziejski
Department of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Latent Variable Models for Understanding User Behavior in Software Applications

by

Ardavan Saeedi

Submitted to the Department of Electrical Engineering and Computer Science
on December 1, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Understanding user behavior in software applications is of significant interest to software developers and companies. By having a better understanding of the user needs and usage patterns, the developers can design a more efficient workflow, add new features, or even automate the user's workflow. In this thesis, I propose novel latent variable models to understand, predict and eventually automate the user interaction with a software application. I start by analyzing users' clicks using time series models; I introduce models and inference algorithms for time series segmentation which are scalable to large-scale user datasets. Next, using a conditional variational autoencoder and some related models, I introduce a framework for automating the user interaction with a software application. I focus on photo enhancement applications, but this framework can be applied to any domain where segmentation, prediction and personalization is valuable. Finally, by combining sequential Monte Carlo and variational inference, I propose a new inference scheme which has better convergence properties than other reasonable baselines.

Thesis Supervisor: Professor Joshua B. Tenenbaum
Title: MIT, Department of Brain and Cognitive Sciences

Thesis Supervisor: Professor Ryan P. Adams
Title: Princeton University, Department of Computer Science

Acknowledgments

I would like to thank my PhD thesis committee Ryan Adams, Joshua Tenenbaum, Matthew Hoffman, Leslie Kaelbling and Tamara Broderick. I am honored to call professor Ryan Adams my mentor and advisor. In the past three years, his patience and curiosity has astonished me and his endless support, encouragement, and guidance has been invaluable on both academic and personal levels. I am very grateful for him and for being given the chance to learn from him. I would like to express my deepest appreciations to professor Joshua Tenenbaum for hosting me in his lab during my time at MIT; I benefited from the intellectual environment of the computational cognitive science lab and learned a lot from other lab members. I am thankful to Matthew D. Hoffman for introducing me to the field of user modeling, for his guidance, and for many fun discussions on modeling and inference. I would like to thank Professors Leslie Kaelbling and Tamara Broderick for reading my thesis and sharing their time and insightful comments with me .

I would like to thank all my friends and colleagues at MIT and Harvard from whom I have learned a lot directly and indirectly. Thanks to Karthik Narasimhan and Tejas Kulkarni with whom I spend most of my time during my PhD. I greatly enjoyed our long and thoughtful discussions on AI and machine learning. I am also very grateful for being able to work with Kayhan Batmanghelich, Johnathan Huggins, Matthew Johnson, Sam Gershman, Yura Perov and the Harvard Intelligent Probabilistic Systems (HIPS) group.

I am thankful to CSAIL and to MIT for all the chances they gave me to explore my passion without limits. The abundance of resources and talent has always aspired me. I am very grateful for being surrounded by the smartest and most driven people in the world. Thank you Rahul Krishnan for giving feedback on various papers and presentations. I thoroughly enjoyed our discussions on machine learning and healthcare. Thank you Ali Vakilian, Amin Rahimian, Diego Cifuentes, Reza Mohammadi, Sajjad Mohammadi, Mikhil Ranka and Mohammad Rashidian. I enjoyed hearing about your research and discussing various topics from science to culture with you over lunch or dinner.

Last but not least, I am forever grateful to my parents and my brother for their love and support, especially my mother who has always been there for me and helped me grow as a person. I thank my father for having a significant impact on my personality and curiosity during his short life. I would like to express my sincere gratitude to Asma for her unconditional love and support and for helping me to become more persistent on my goals. I feel I have become a more complete person with her.

Contents

[1]This chapter is based on the work by Saeedi et al. (2016) (http://proceedings.mlr.press/v48/saeedi16.pdf).

[2]This chapter is based on the work by Huggins et al. (2015) (http://proceedings.mlr.press/v37/hugginsa15.pdf).
[3]This chapter is based on the work by Batmanghelich et al. (2016) (http://aclweb.org/anthology/P/P16/P16-2087.pdf)
[4]This chapter is based on the work by Saeedi et al. (2017a) (https://arxiv.org/pdf/1704.04997.pdf).

[5]This chapter is based on the work by Saeedi et al. (2017b) (http://jmlr.org/papers/v18/15-615.html).

List of Figures

List of Tables

CHAPTER 1

Introduction

User modeling has been an active field of research in both human-computer interaction (HCI) and machine learning communities in recent years. Significant growth in the amount of available users' data, improvement in computational resources and the fierce competition between user-based businesses are all among the reasons behind this recent attention.

Growth in the user data is the result of growing avidity towards collecting more data with the hope of better understanding the users. Proprietary software applications are gathering more and more data with high time-resolution and from different modalities; data such as mouse movement, event clicks, and location are now routinely collected by many software applications. Gaining insight into such unprecedented data, requires developing new tools and models.

This growth in data collection has been happening in parallel with the growth in available inexpensive and powerful computational resources. In fact the boom in data agglomeration is partially fueled by this massive growth; inexpensive and powerful hardware means more data can be stored and processed. On the other hand, analyzing more data necessitates developing more affordable and powerful hardware; hence, a virtuous cycle has been formed. Another positive consequence of improvement in hardware is more powerful and flexible software applications. For instance, photo editing software applications are now available with countless features on mobile devices. Professional-grade photo-editing software has also advanced by becoming more powerful at the price of becoming more complicated.

Software companies are competing to attract more users by adding more features to their products. This trend has resulted in feature overload; there are more features in the software than an average user is able to use. As an example, after loading a photo in a photo editing software application, the user is confronted with an array of cryptic sliders like "clarity", "temp", and "highlights" (see Fig. 1-1). Understanding the users' behavior, allows recommendation of more relevant and personalized features. Thus, user modeling is now an imperative part of designing most user-based

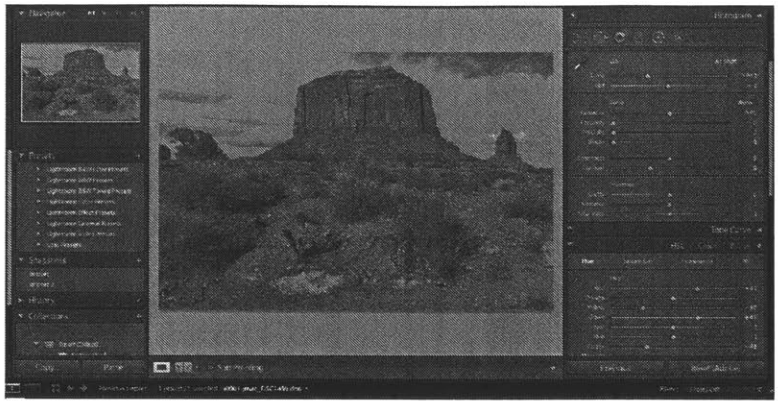


Figure 1-1: **User interface for Adobe Lightroom.** The flexible and powerful interface enables experts to achieve impressive results; however, novices may struggle to complete even basic tasks.

software applications.

Although the above reasons may explain the recent interest in applying machine learning techniques to user modeling, we should mention that this topic is not entirely new. The literature on it can be traced back to the paper by Brown and Burton (1978), where they proposed a framework called *procedural networks* for representing the knowledge underlying a skill. They applied the framework to debugging a student's behavior in mathematical problems, which can be considered as a special case of user modeling (Webb et al., 2001). There is more related work from the '80s and '90s on the problem of student modeling; Webb et al. (2001) covered a comprehensive list of related work from that period. The advent of the world-wide-web and e-commerce resulted in accelerated growth in the area of user modeling with machine learning techniques. For some early works see, for instance, Widmer and Kubat (1996); Pazzani and Billsus (1997); Ungar and Foster (1998).

In general, user modeling refers to methods and algorithms used to gain insight about a user based on previous interactions (Bjorkoy 2010; Webb et al. 2001; Pazzani and Billsus 2007). User models may be developed for different purposes; some common ones are as follows:

- **Discovering common usage patterns across different users** (*e.g.,* Bae et al. 2017; Kosmalla et al. 2015; Liu et al.; Wang et al. 2016): Log files of software application contain user actions and their corresponding time-stamps; however, in many applications, the boundaries between different tasks (*i.e.,* a sequence of events) are not clear and the number of tasks done in each work session is unknown a priori. In other words, the tasks are unobserved and should be inferred from the observed event sequence (see Fig. 1-2). Segmentation and various time series models are commonly applied to this task.

- **Identifying different user groups based on their usage patterns** (*e.g.,* Zhao et al. 2016;

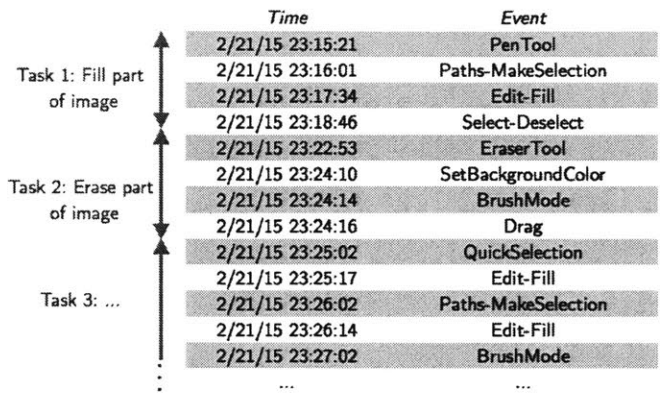


Figure 1-2: **A sample log file from Adobe Photoshop.** Two sample columns in a log file from Adobe Photoshop which include timestamps and the events. Note that the boundaries between the tasks are not evident from only the log file.

Nikolenko and Alekseyev 2016; Geyik et al. 2015): In most software applications, there is a latent heterogeneity between users; that is, users may belong to different usage pattern groups. Inferring and identifying user clusters can be useful from different perspectives. From a business perspective, it can help in identifying what types of users are more engaged with the application or willing to pay for it. From a design point of view, knowing what types of features different groups are using can potentially lead to an enhanced user interface and more optimized workflow.

- **Personalization and user-dependent recommendation** (*e.g.,* Hong et al. 2016; Rabbi et al. 2017): Personalization is known to be a possible solution to the problems of information and feature overload (Bjorkoy, 2010). By recommending the features best suited for each user's interests and abilities, personalization can alleviate these problems and improve the user's experience. In general, recommendation models may not be user-dependent. Based on the level of dependency on each user, recommendation models can be population-based (*i.e.,* one-fits-all), semi-population-based, or individual-based[1] (Hong et al., 2016).

Latent variable models are a natural fit for these tasks since they all involve some unobserved variable. In discovering the common usage patterns, this variable can be the unobserved task, in identifying user groups this variable can be the hidden group, and for the personalization the user's type can be the unobserved variable. Latent variable models such as Gaussian mixture, factor analysis, and topic models have been applied to various HCI studies; for a review of these studies, see Robertson and Kaptein (2016). In this thesis, we propose novel latent variable models for tackling

[1] A more detailed treatment of these approaches is provided in Chapter 6.

the three goals of user modeling which we mentioned above.

Generally, latent variable models are a broad class of probabilistic models that are composed of two parts: 1) the observed variables, which we can directly measure and 2) the unobserved variables, which we assume exist and can affect the observed variables. The main advantage of using these models is that we can incorporate our prior knowledge when designing a model. Graphical models are the standard tool for designing these models; they allow for flexible and powerful models by providing a framework for prior knowledge incorporation. As we will see in later chapters, in most cases more a flexible model means a more challenging inference procedure for the latent variables. We will discuss some widely used approaches for learning and inference in these models in Chapter 2.

Discovering common usage patterns

For the first goal, we focus on identifying common tasks from user event traces (*i.e.,* log files). A task is a group of events. For instance, for the task of deleting part of an image, the events can be opening the file, selecting the appropriate part of the image, deleting, and finally saving the image. Having a good understanding of the tasks done by the users can potentially help in designing better workflows and predicting the user's intention.

We may impose different assumptions on the structure of the latent variables corresponding to the tasks. For instance, if we consider a sequential structure and assume that the transition and event emission probabilities are dependent only on the current task, then we are essentially assuming a hidden Markov model (HMM). HMMs and their variants have been applied to time series segmentation problems in HCI (*e.g.,* Yin et al. 2008; Bui et al. 2002). Having a sequential model structure is advantageous if we are interested in answering questions about the time-dependent aspect of the data. Using these types of models, we can potentially answer questions such as where the user is switching from one task to another (*i.e.,* change-point detection problem) or what is the most likely next task given the current task. We propose two latent variable models with sequential structures in Chapters 3 and 4.

In Chapter 3, we introduce the segmented infinite HMM (siHMM), a hierarchical infinite HMM (iHMM) that supports a simple, efficient inference scheme. The iHMM is a Bayesian nonparametric variant of HMM that we will explain in Chapter 2. The siHMM is well suited to segmentation problems, where the goal is to identify points at which a time series transitions from one relatively stable regime to a new regime. Conventional iHMMs often struggle with such problems, since they have no mechanism for distinguishing between high- and low-level dynamics. Hierarchical HMMs (HHMMs) can do better, but they require much more complex and expensive inference algorithms. The siHMM retains the simplicity and efficiency of the iHMM but outperforms it on user segmentation problems, achieving performance that matches or exceeds that of a more complicated HHMM.

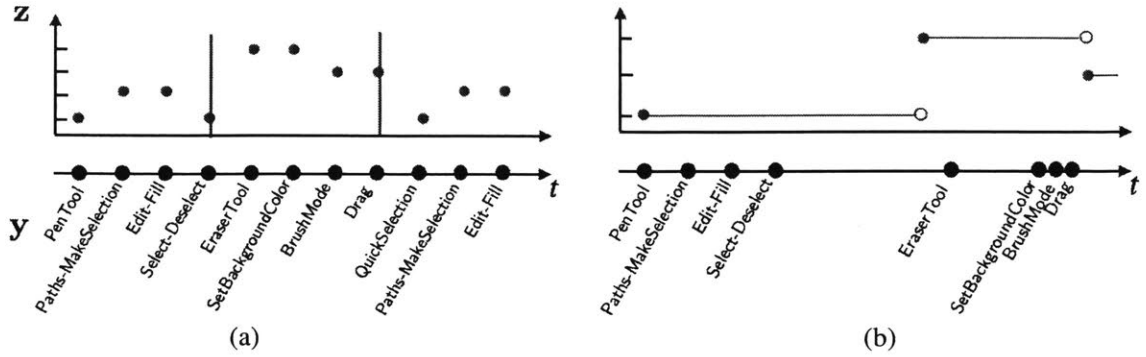


Figure 1-3: **Sample observations and inferred states for siHMM and Markov Jump Process (MJP) (a)** Observations (denoted by $\mathbf{y}$) for the siHMM model are assumed to be equidistant and discrete-time. Hence, we ignore the exact timestamp of sample observations in Fig. 1-2. The inferred states are the index of the hidden states (denoted by $\mathbf{z}$) and the boundaries between the tasks. Note that we are inferring both the dynamics within each task and the high-level change points from one task to another. **(b)** In the MJP model, we assume that observations can occur on continuous-time points. That is, for the sample observations in Fig. 1-2, the events are not distributed evenly over time. We infer the index of the hidden states and the sojourn times in each state.

Chapter 4 covers another sequential model for time series segmentation based on Markov jump processes (MJPs). In contrast to siHMM, where we assume time steps are discrete, MJPs are continuous-time models that are applicable in settings where event timestamps are available and consecutive events are not equidistant. These flexible and powerful models are used to model a wide range of phenomena from disease progression to RNA path folding. However, learning and inference in these models are challenging. Maximum likelihood estimation of parametric models leads to degenerate trajectories and inferential performance is poor in nonparametric models. We take a small-variance asymptotics (SVA) approach to overcome these limitations. SVA has shown promising results on scaling-up Bayesian nonparametric models (see Roychowdhury et al. 2013a; Jiang et al. 2012). Our experiments on user trace segmentation and other applications demonstrate that our model is competitive with or outperforms widely used MJP inference approaches in terms of both speed and reconstruction accuracy.

Fig. 1-3 illustrates the inferred hidden states in both siHMM and MJP for the sample observations in Fig. 1-2. If we assume that knowing the exact timestamps is unnecessary and a hierarchical structure exists over the dynamics of the hidden states, then the siHMM is a better choice between the two models. However, if we assume the exact continuous-time timestamps are informative in our dataset and not much gain occurs in a multilevel modeling, then the MJP is more suitable for the dataset. We will discuss the merits of each of these models in more detail in Chapters 3 and 4.

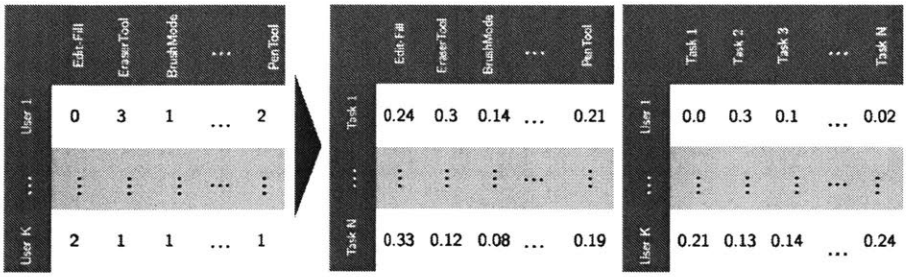


Figure 1-4: **A sample result from applying topic models to log files.** We introduce a nonstandard variant of the topic models that can incorporate distributional representation of the events (see text).

If we ignore the sequential structure of the events in the log files, then each user can be represented as a bag of events. One assumption that we can make is that each user is a mixture of distinct distributions (one for each task). This assumption leads to topic model structure for the latent variables. Topic models have also been applied to user data; see, for instance Geyik et al. (2015). Based on a topic model, we can answer questions such as what are the frequent events in each task, or what are the tasks that each user is interested in. For instance, Fig. 1-4 demonstrates an example of inferring the set of events in each task and the tasks that each user spends more time on. These models are limited in the sense that they generally ignore the sequential aspect of the data so they cannot be readily applied to settings where sequential models such as HMMs are useful.

We apply topic models to user traces in Chapter 5. We introduce a nonstandard variant of topic models that can benefit from the distributional representation of the events (*e.g.,* commands in a photo editing software). We obtain distributional representations from Adar et al. (2014). These high-dimensional vector representations can capture semantic regularities in the event space; they exhibit semantic consistency over directional metrics such as cosine similarity. For instance, vector representation of similar commands such as "eraser tool" and "background eraser tool" will have a high cosine similarity value. Traditional topic models cannot account for semantic regularities in the command space. Neither categorical nor Gaussian observational distributions used in existing topic models are appropriate to leverage such correlations. In Chapter 5, we propose using the von Mises-Fisher distribution to model the density of commands over a unit sphere. Such a representation is well-suited for directional data. Experiments demonstrate that our method outperforms competitive approaches in terms of topic coherence on different datasets while offering efficient inference.

Identifying user groups

We focus on identifying different user groups in a photo editing software application where instead of a log file, we have the images and their corresponding edits for each user. The edits are repre-

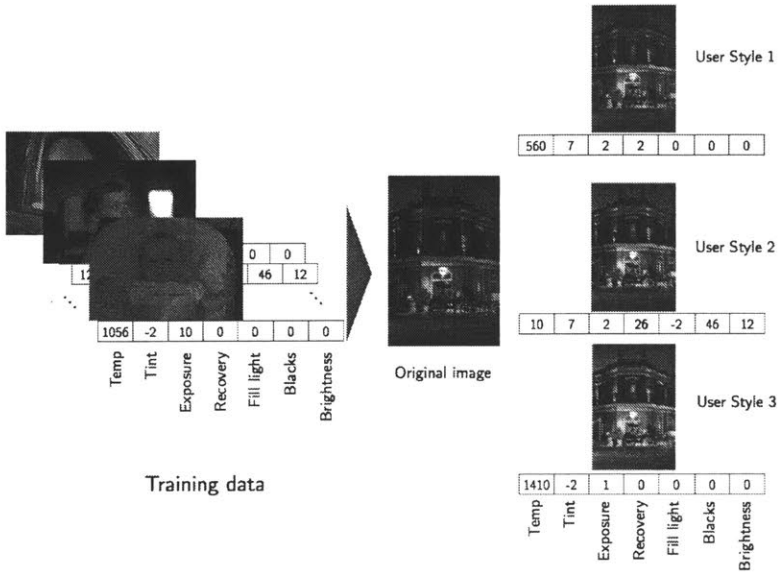


Figure 1-5: **Identifying user groups based on images and their corresponding edits.** The training data contains a set of images and their edits. The inferred user styles can be applied to different images and produce different styles. In the above example, inferred user styles 1 and 3 are from novice users which typically do not utilize various sliders. The inferred user style 2 corresponds to an expert user.

sented as a vector of slider values (*e.g.,* vector of sliders such as "Brightness" or "Contrast"). Our goal is to infer different user styles applied to the same or similar image. That is due to the fact that in photo enhancement, different experts may make very different aesthetic decisions when faced with the same image, and a single expert may make different choices depending on the intended use of the image (or on a whim). We therefore want a system that can propose multiple diverse and high-quality edits. In Chapter 6, we develop a statistical model that meets these objectives. Our model builds on recent advances in neural network generative modeling and scalable inference, and uses hierarchical structure to learn editing patterns across many diverse users. Empirically, we find that our model outperforms other approaches on this challenging multimodal prediction task.

Fig. 1-5 illustrates a synthetic example of training data and the inferred user styles from our model applied to an image. Identified user styles can be applied to a new image and produce a diverse set of edits applied to a single image. These styles may correspond to styles of a novice or multiple different experts. Without identifying different user styles, this problem will be reduced to predicting a single edit for an image. However, in many applications including photo enhancement there is no single "correct" edit for a given input (e.g. image). Most work predicting photo edits as-

sume a single best edit, given an image. See for instance models by Yan et al. (2016) or Bychkovsky et al. (2011) where the authors use a gaussian process regression or a deep neural network to learn a mapping from the image to the slider values. These methods approximate a parametric function by minimizing a squared (or a similar) loss. A thorough review of related models is provided in Chapter 6.

Personalization and user-dependent recommendation

There is an extensive literature on personalization and recommendation in user models; tools such as collaborative filtering (Su and Khoshgoftaar, 2009), k-NN, and kernel-based algorithms (Ghazanfar et al., 2012) have been widely used for these tasks. Particularly, various latent variable models have been proposed in the literature for personalization. For example, Ovsjanikov and Chen (2010) introduce a modified topic model for personalization in the settings with high sparsity and volatility. Hu et al. (2014) proposed a two-step procedure for personalization using a latent Dirichlet allocation (LDA) model. They first infer user interests and form a user profile which is a vector of topic proportions for each user in an LDA model. Next, they use the user's interest profile to conduct a nearest neighbor search over all the other users and make recommendations based on the top 100 users. For more examples of these types of models see Brusilovski et al. (2007).

We develop a latent variable model for personalization in photo editing applications in Chapter 6. Similar to the previous task, we assume we have a set of images and their corresponding edits by a user. Our aim is to predict the edits applied to a new image, given a history of edits from a user (and also other users). The model is an extension of the style identification model which we introduced above. We show that in addition to identifying user groups, the model is capable of adapting to a user's aesthetic preferences.

Limitations of the models and some possible solutions

Despite outperforming other baselines, the latent variable models in Chapters 3 to 6 may suffer from poor inference performance. This is due to the fact that the variational inference is known to have difficulties in approximating distributions if non-flexible variational distributions are used. To improve the performance of variational inference in these types of models, we introduce Discrete Particle Variational Inference (DPVI) in Chapter 7. DPVI is a new approach that combines key strengths of Monte Carlo and variational techniques. DPVI is based on a novel family of particle-based variational approximations that can be fit using simple, fast, deterministic search techniques. Like Monte Carlo, DPVI can handle multiple modes, and yields exact results in a well-defined limit. Like unstructured mean-field, DPVI is based on optimizing a lower bound on the partition function; when this quantity is not of intrinsic interest, it facilitates convergence assessment and debugging.

In addition to application of user trace segmentation, DPVI performance is illustrated and evaluated via experiments on several other parametric and nonparametric models. Results show that DPVI can offer appealing time/accuracy trade-offs as compared to multiple alternatives. We discuss further limitations of the proposed models along with some possible future research directions in Chapter 8.

CHAPTER 2

Background

In this chapter, we provide a brief overview of the methods and models which different chapters of this thesis use extensively. In particular, we explain various approaches for inference in latent variable models and also review some Bayesian nonparametric models we utilized to build our proposed latent variable models.

2.1 Inference for latent variable models

Consider a Bayesian model $p(\mathbf{y}|\mathbf{z})p(\mathbf{z})$ in which $\mathbf{y}$ denote the observations and $\mathbf{z}$ denote the latent variables. In Bayesian inference, we are typically interested in approximating the posterior probability distribution $p(\mathbf{z}|\mathbf{y})$ over latent variables $\mathbf{z} = \{z_1, \ldots, z_N\}$, where the target distribution is known only up to a normalizing constant $\mathcal{Z}$: $p(\mathbf{z}|\mathbf{y}) = f(\mathbf{z})/\mathcal{Z}$. We will refer to $f(\mathbf{z}) \geq 0$ as the *score* of $\mathbf{z}$ and $\mathcal{Z}$ as the *partition function*. Most approximate inference algorithms fall into two classes: Monte Carlo methods and variational methods. Monte Carlo methods generate samples from approximations to the posterior distribution that grow more accurate as the technique is given more compute time. Variational methods (Wainwright and Jordan, 2008) treat probabilistic inference as an optimization problem over a set of distributions. This set is typically constrained (*e.g.,* to factorized conjugate exponential distributions), thereby attaining efficiency at the expense of bias.

2.1.1 Importance sampling and sequential Monte Carlo (SMC)

A general way to approximate $p(\mathbf{z}|\mathbf{y})$ is with a weighted collection of K particles, $\{\mathbf{z}^1, \ldots, \mathbf{z}^K\}$:

$$p(\mathbf{z}|\mathbf{y}) \approx q(\mathbf{z}) \triangleq \sum_{k=1}^{K} w^k \delta_{\mathbf{z}^k}, \tag{2.1}$$

where $\mathbf{z}^k = \{z_1^k, \ldots, z_N^k\}$, and δ_X is the Dirac measure at X. Importance sampling is a Monte Carlo method that stochastically generates particles from a proposal distribution, $\mathbf{z}^k \sim r(\cdot)$, and computes the weight according to $w^k \propto f(\mathbf{z}^k)/r(\mathbf{z}^k)$. Importance sampling has the property that the particle approximation converges to the target distribution as $K \to \infty$ (Robert and Casella, 2004).

Sequential Monte Carlo (SMC) methods such as particle filtering (Doucet et al., 2001) apply importance sampling to stochastic dynamical systems (where n indexes time) by sequentially sampling the latent variables at each time point using a proposal distribution $r(z_n|z_{n-1})$. As a concrete example, consider the following stochastic dynamical system also known as a hidden Markov model (HMM). The joint probability density can be written as:

$$p(\mathbf{z}, \mathbf{y}) = p(z_1) \prod_{n=1}^{T} p(z_n|z_{n-1}) p(y_n|z_n). \tag{2.2}$$

Our goal is to compute the posterior $p(z_{1:T}|y_{1:T})$. However, for most models with non-linear non-Gaussian transition and observation distributions, the posterior does not admit a closed form and we need to resort to approximate inference methods. In an SMC scheme, particles are extended over time by a sequence of propose and resample steps. At time $n = 1$, we sample a set of K particles from $r(z_1)$ using standard importance sampling. For $n > 1$, we first resample the particles based on their weight:

$$a_{n-1}^k \sim \text{Cat}(\mathbf{W_{n-1}}), \tag{2.3}$$

where $W_n \triangleq (W_n^1, \cdots, W_n^K)$ denotes the normalized importance weights, Cat denotes the categorical distribution and a_{n-1}^k represents the index of the "parent" at time $n-1$ of particle $z_{1:n}^k$ for $n = 2, \cdots, T$. Next, we propose new set of particles conditioned on the resampled particles:

$$z_n^k \sim r(z_n|z_{n-1}^{a_{n-1}^k}), \tag{2.4}$$

and extend each particle with the new sample: $z_{1:n}^k = (z_{1:n-1}^{a_{n-1}^k}, z_n^k)$. At time T, we have the following approximation to the posterior: $p(z_{1:T}|y_{1:T}) \approx \sum_{k=1}^K W_T^k \delta_{z_{1:T}^k}$.

Up to this point, we assumed the parameters for the transition $p(z_n|z_{n-1})$ and emission $p(y_n|z_n)$ distributions are known; for models with unknown parameters $\theta \in \Theta$ we can assume a prior $p(\theta)$ over the parameter (vector) and approximate the posterior $p(\theta, z_{1:T}|y_{1:T})$. A Markov chain Monte Carlo (MCMC) that alternates between sampling θ and $z_{1:T}$ is a natural choice for this purpose. To apply MCMC we need proposal distributions for θ given $z_{1:T}$ and vice versa. Sampling exactly from $p(\theta|z_{1:T}, y_{1:T})$ is typically feasible. However, exact sampling from $p_\theta(z_{1:T}|y_{1:T})$ is only possible

for few models with restrictive assumptions (e.g. linear Gaussian models). For most models, we need to design a proposal distribution for sampling from $p_\theta(z_{1:T}|y_{1:T})$.

One popular approach in such scenarios is to use Particle MCMC (PMCMC) methods introduced by Andrieu et al. (2010a). These methods use an SMC algorithm to propose samples for $p_\theta(z_{1:T}|y_{1:T})$. The main advantage of these methods is the asymptotic exactness of sampling from $p_\theta(z_{1:T}|y_{1:T})$ which is due to the SMC algorithm properties. For a comprehensive overview of PMCMC methods see Andrieu et al. (2010a).

2.1.2 Variational inference

Variational methods (Wainwright and Jordan, 2008) define a parametrized family of probability distributions $\mathcal{Q}$ and then choose $q \in \mathcal{Q}$ that maximizes the *negative variational free energy*:

$$\mathcal{L}[q] = \int_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{z}, \mathbf{y})}{q(\mathbf{z})} \mathrm{d}\mathbf{z}. \tag{2.5}$$

The negative variational free energy is related to the partition function $\mathcal{Z}$ and the KL divergence through the following identity:

$$\log \mathcal{Z} = \mathrm{KL}[q||p] + \mathcal{L}[q], \tag{2.6}$$

where

$$\mathrm{KL}[q||p] = \int_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{y})}. \tag{2.7}$$

Since $\mathrm{KL}[q||p] \geq 0$, the negative variational free energy is a lower bound on the log partition function, achieving equality when the KL divergence is minimized to 0. Maximizing $\mathcal{L}[q]$ with respect to q is thus equivalent to minimizing the KL divergence between q and p. Note that we are assuming continuous latent variables; for discrete variables, the integrations should be replaced with summations in the above equations.

Unlike the Monte Carlo methods described in the previous section, variational methods do not in general converge to the target distribution, since typically p is not in $\mathcal{Q}$. The advantage of variational methods is that they guarantee an improved bound after each iteration, and convergence is easy to monitor (unlike most Monte Carlo methods). In practice, variational methods are also often more computationally efficient.

Depending on the variational distribution family that we consider, the optimization procedure for equation Eq. (2.5) can be different. For instance, restricting the family to exponential distributions, may result in closed-form update rules of mean field variational inference. More flexible

parametric density estimators such as multilayer perceptron (MLP) are also possible; we will discuss them in Section 2.1.3.

2.1.3 Variational autoencoders (VAEs)

The VAE, introduced by Kingma and Welling (2013), has been successfully applied to various models with continuous latent variables and a complicated likelihood function (*e.g.*, a neural network with nonlinear hidden layers). In these settings, posterior inference is typically intractable, and even approximate inference may be prohibitively expensive to run in the inner loop of a learning algorithm. The VAE allows this difficult inference to be amortized over many learning updates, making each learning update cheap even with complex likelihood models.

As an instance of such models, consider modeling a set of M i.i.d. observations $\mathbf{Y} = \{\mathbf{y}^{(m)}\}_{m=1}^{M}$ with the following generative process: $\mathbf{z}^{(m)} \overset{\text{iid}}{\sim} h$ and $\mathbf{y}^{(m)} \sim f(g_\theta(\mathbf{z}^{(m)}))$, where $\mathbf{z}^{(m)}$ is a latent variable generated from a prior $h(\mathbf{z})$ (e.g., $\mathcal{N}(0, I)$) and the likelihood function

$$p_\theta(\mathbf{y}^{(m)}|\mathbf{z}^{(m)}) = f(\mathbf{y}^{(m)}; g_\theta(\mathbf{z}^{(m)})) \tag{2.8}$$

is a simple distribution f whose parameters $g_\theta(\mathbf{z}^{(m)})$ can be a complicated function of $\mathbf{z}^{(m)}$. For example, $p_\theta(\mathbf{y}^{(m)}|\mathbf{z}^{(m)})$ might be $\mathcal{N}(\mathbf{y}^{(m)}; \mu(\mathbf{z}^{(m)}; \theta), \Sigma(\mathbf{z}^{(m)}; \theta))$ where the mean and the covariance depend on $\mathbf{z}^{(m)}$ through a multi-layer perceptron (MLP) richly parameterized by weights and biases θ.

In the VAE framework, the posterior density $p_\theta(\mathbf{z}|\mathbf{y})$ is approximated by a recognition network $q_\phi(\mathbf{z}|\mathbf{y})$, which can take the form of a flexible conditional density model such as an MLP parameterized by ϕ. To learn the parameters of the likelihood function θ and the recognition network ϕ, the following lower bound on the marginal likelihood is maximized with respect to θ and ϕ:

$$\mathcal{L}_{\text{VAE}}(\phi, \theta) \triangleq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{y})||p(\mathbf{z})). \tag{2.9}$$

To compute a Monte Carlo estimate of the gradient of this objective with respect to ϕ, Kingma and Welling (2013) propose a reparameterization trick for sampling from $q_\phi(\mathbf{z}|\mathbf{y})$ by first sampling from an auxiliary noise variable and then applying a differentiable map to the sampled noise. This yields a differentiable Monte Carlo estimate of the expectation with respect to ϕ. Given the gradients, the parameters are updated by stochastic gradient ascent.

2.2 Bayesian nonparametric (BNP) models

Bayesian nonparametric models are flexible generative models that can capture complex latent structures and adapt their complexity to the data. These models provide another view on the problem of model selection in machine learning. Typically, the number of parameters in a model can be determined via some model selection metrics which favors simpler models and can fit the data well. In the BNP models the number of parameters is not fixed a priori; it can grow in the inferred model as more data is observed. In fact complexity is controlled via the prior distribution; with small datasets the posterior is typically simpler. By adding more data points the posterior gets more complex and more accurate.

Being flexible and also an elegant alternative to parametric model selection, makes BNP models ideal candidates for latent variable modeling. There are two main categories of BNP models depending on whether they are based on a prior over functions (*i.e.,* Gaussian process) or a prior over distributions (*i.e.,* Dirichlet process). In this thesis, we focus on latent variable models which are extensions on Dirichlet processes. For a detailed review on BNPs see, for example, Gershman and Blei (2012); Sudderth (2006).

2.2.1 Dirichlet process mixture models (DPMM)

A DPMM, a mixture model with a Dirichlet process (DP) prior, has been commonly used in practice. Having a simple and scaleable posterior sampling scheme can be the reason behind its wide applicability. DP defines a distribution over random probability measures. It is defined by a *concentration parameter* α and a base measure H over a measurable space Θ. More concretely a DP is defined as follows:

Definition 2.2.1 (Dirichlet process)**.** Assume $\alpha > 0$ and a probability space $(\Theta, \mathcal{F}, H)$. We say G is distributed according to a Dirichlet process with parameters α and H and write $G \sim \text{DP}(\alpha, H)$ if for every measurable partition $\{T_1, \ldots, T_K\}$ of Θ:

$$\bigcup_{k=1}^{K} T_k = \Theta, \qquad T_i \cap T_j = 0 \;\; i \neq j, \tag{2.10}$$

we have

$$(G(T_1), G(T_2), \ldots, G(T_K)) \sim \text{Dir}(\alpha H(T_1), \alpha H(T_2), \ldots, \alpha H(T_K)). \tag{2.11}$$

Intuitively, this means if we sample from a DP and sum over the probabilities in a region $T_k \in \Theta$ then there will be a mass of $H(T_k)$ in that region on average. The concentration parameter α controls the inverse variance; a high α means more concentrated mass around the $H(T_k)$.

There are various representations of the DP. The one described above is not a constructive representation; in what follows we provide two more representations of DP which are constructive and can be used in devising inference schemes. We define a DPMM based on these constructive representations.

DPMM generates data from the following process (Antoniak, 1974; Escobar and West, 1995):

$$\begin{aligned} \pi|\alpha &\sim \text{GEM}(\alpha), \qquad \theta_k|H \sim H, \\ z_n|\pi &\sim \pi, \qquad y_n|z_n, (\theta_k)_{k=1}^{\infty} \sim F(\theta_{z_n}), \end{aligned} \tag{2.12}$$

where $\text{GEM}(\alpha)$ is the stick-breaking distribution with concentration parameter $\alpha \geq 0$ defined by

$$\pi_i = \prod_{j<i}(1-v_j)(v_i), \qquad v_j \overset{iid}{\sim} \text{Beta}(1,\alpha) \tag{2.13}$$

and H is a base distribution over the parameter θ_k of the observation distribution $F(y_n|\theta_{z_n})$. Fig. 2-1(a) demonstrates the graphical model for the DPMM.

Since the Dirichlet process induces clustering of the parameters θ into K distinct values, we can equivalently express this model in terms of a distribution over cluster assignments, $z_n \in \{1,\ldots,C\}$. The distribution over z is given by the Chinese restaurant process (Aldous, 1985):

$$P(z_n = c|z_{1:n-1}) \propto \begin{cases} t_c & \text{if } c \leq C_+ \\ \alpha & \text{if } c = C_+ + 1, \end{cases} \tag{2.14}$$

where t_c is the number of data points prior to n assigned to cluster c and C_+ is the number of clusters for which $t_c > 0$.

2.2.2 Hierarchical Dirichlet process (HDP)

HDP introduced by Teh et al. (2006a) is a hierarchical variant of Dirichlet process that can be applied to grouped data (see Fig. 2-1 for its graphical model). We assume that there are M groups and denote the nth observation in group m by y_{mn}. The generative process is as follows:

$$\begin{aligned} \beta|\gamma &\sim \text{GEM}(\gamma), \qquad \pi_m \overset{\text{i.i.d.}}{\sim} \text{DP}(\alpha,\beta), \qquad \theta_k|H \sim H, \\ z_{mn}|\pi_m &\sim \pi_m, \qquad y_{mn}|z_{mn}, (\theta_k)_{k=1}^{\infty} \sim F(\theta_{z_{mn}}), \end{aligned} \tag{2.15}$$

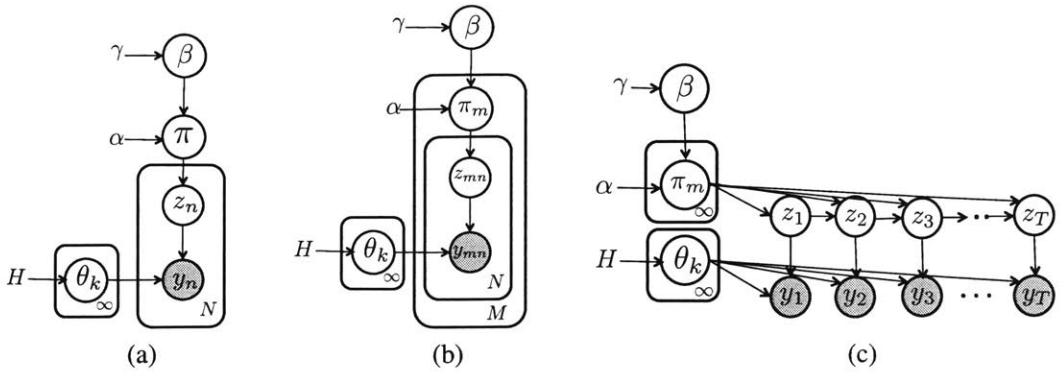


Figure 2-1: **Graphical models for (a)** DPMM (Section 2.2.1), **(b)** HDP (Section 2.2.2) and **(c)** HDP-HMM (Section 2.2.2).

where β and π_m are probability measures over positive integers and each π_m is independently distributed according to DP(α, β). A sample from DP(α, β) can also be shown as:

$$w \sim \text{GEM}(\alpha), \qquad x_k \overset{iid}{\sim} \beta, \qquad \pi_m = \sum_{k=1}^{\infty} w_k \delta_{x_k}. \tag{2.16}$$

HDP can be extended to sequential domains. Applying HDP to an HMM results in a model which is called HDP-HMM (Teh et al., 2006b) or infinite HMM (iHMM). This is a dynamic mixture variant of the HDP, in which there is a mixture component for each possible hidden state. An iHMM generates data from the following process:

$$\begin{aligned} &\beta|\gamma \sim \text{GEM}(\gamma), \qquad \pi_m \overset{\text{i.i.d.}}{\sim} \text{DP}(\alpha, \beta), \qquad \theta_k|H \sim H, \\ &z_{t+1}|z_t, (\pi_m)_{m=1}^{\infty} \sim \pi_{z_t}, \qquad y_t|z_t, (\theta_k)_{k=1}^{\infty} \sim F(\theta_{z_t}), \end{aligned} \tag{2.17}$$

where z_t is the hidden state at time t (see Fig. 2-1(c)).

Like the DPMM, the iHMM induces a sequence of cluster assignments. The distribution over cluster assignments is given by the Chinese restaurant franchise (Teh et al., 2006b). Letting t_{jc} denote the number of times cluster j transitioned to cluster c, z_t is assigned to cluster c with probability proportional to $t_{z_{t-1}c}$, or to a cluster never visited from z_{t-1} ($t_{z_{t-1}c} = 0$) with probability proportional to α. If an unvisited cluster is selected, z_t is assigned to cluster c with probability proportional to $\sum_j t_{jc}$, or to a new cluster (*i.e.,* one never visited from any state, $\sum_j t_{jc} = 0$) with probability proportional to γ.

2.2.3 Hierarchical Gamma-exponential process (HΓEP)

HΓEP, the hierarchical variant of the gamma-exponential process (ΓEP), can be used for building continuous-time sequential models. We denote the Moran gamma process with base measure H and rate parameter γ by $\Gamma\mathrm{P}(\gamma, H)$ (Kingman, 1993). We recall that the Moran gamma process is a distribution over measures. If $\mu \sim \Gamma\mathrm{P}(\gamma, H)$ is a random measure distributed according to a Moran gamma process with base measure H on the probability space $(\Omega, \mathcal{F})$ and rate parameter γ, then for all measurable partitions of Ω, $(A_1, \ldots, A_\ell)$, μ satisfies

$$(\mu(A_1), \ldots, \mu(A_\ell)) \sim \mathrm{Gam}(H(A_1), \gamma) \times \cdots \times \mathrm{Gam}(H(A_\ell), \gamma). \tag{2.18}$$

The HΓEP generates a state/dwell-time sequence $z_0, t_1, z_1, t_2, z_2, t_3, z_3, \ldots$ (with z_0 assumed known) according to (Saeedi and Bouchard-Côté, 2011):

$$\mu_0 \sim \Gamma\mathrm{P}(\gamma_0, \alpha_0 H_0), \qquad \mu_m \mid \mu_0 \overset{\text{i.i.d.}}{\sim} \Gamma\mathrm{P}(\gamma, \mu_0), \tag{2.19}$$

$$z_k \mid z_{k-1}, (\mu_m)_{m=0}^{\infty} \sim \bar{\mu}_{z_{k-1}}, \qquad t_k \mid z_{k-1}, (\mu_m)_{m=0}^{\infty} \sim \mathrm{Exp}(\|\mu_{z_{k-1}}\|, \tag{2.20}$$

where H_0 is the base probability measure, α_0 is a concentration parameter, $\bar{\mu}_m \triangleq \mu_m / \|\mu_m\|$, and $\|\mu\|$ denotes the total mass of the measure μ.

2.2.4 Truncated variational inference in DPs and HDPs

Inference in DPMM and HDP models can be done via Monte Carlo methods such as Gibbs sampling and also via variational methods. For the purpose of this thesis, we only focus on truncated variational methods since they are more amenable to scaling up to large datasets.

Performing variational inference in BNP models is challenging due to the fact that these models can potentially have infinite number of variational parameters. Hence, most variational approaches in these models are based on truncation. Truncation can occur in two different ways: 1) truncating the two stick-breaking distributions in the definition of HDP (used by Hoffman et al. 2013) and 2) truncating the support of $q(z_t)$ which is the variational distribution for the hidden states in an HDP model (used by Bryant and Sudderth 2012). We only use the second scheme in this thesis but we explain both here for the sake of comparison. To be more concrete, we compare the inference schemes in the context of an HDP topic model.

Truncating the stick-breaking process for the variational distribution The first scheme introduced by Hoffman et al. (2013), assumes the following generative model for an HDP topic model for D documents (which is different from Eq. (2.15)):

1. Draw breaking proportions at the corpus level, $\beta_j \overset{iid}{\sim} \text{Beta}(1, \gamma)$ for $j \in \{1, 2, 3, \cdots\}$.

2. Draw infinite number of topics, $\theta_k \sim \text{Dir}(\eta)$ for $k \in \{1, 2, 3, \cdots\}$.

3. For each document:

   (a) Draw document-level topic indices, $c_{di} \sim \text{Mult}(\sigma(\beta))$ for $i \in \{1, 2, 3, \cdots\}$ where $\sigma_i(\beta) = \prod_{j<i}(1 - \beta_j)(\beta_i)$.

   (b) For each word n:

      i. Draw breaking proportions at the document level, $\pi_{di} \sim \text{Beta}(1, \alpha)$.

      ii. Draw topic assignment $z_{dn} \sim \text{Mult}(\sigma(\pi_d))$.

      iii. Draw word $w_n \sim \text{Mult}(\theta_{c_{dz_{dn}}})$.

The variables in this model that need to be inferred are $v_j, \theta_k, c_{di}, \pi_{di}$, and z_{dn}. Since the model contains infinite number of hidden variables, a naive variational approach requires optimizing over potentially infinite number of variational parameters. To address this issue, one approach introduced by Blei et al. (2006) is to truncate the stick-breaking distributions in the variational distribution at two levels. Considering truncation levels T and K for the document-level and corpus-level sticks, we have the following variational family:

$$q(\theta, \beta, c, \pi, z) = \prod_{k=1}^{K} q(\theta_k) q(\beta_k) \prod_{d=1}^{D} \prod_{i=1}^{T} q(c_{di}) q(\pi_{di}) \prod_{n=1}^{N_d} q(z_{dn}) \tag{2.21}$$

where N_d is the number of words for document d.

For high enough truncation levels T and K, the variational posterior will use as many as topics that are needed and does not necessarily use all the possible T topics; hence, the model is not finite. Also note that we are only truncating the variational distributions and not the generative model.

"Direct assignment" truncation The truncation scheme that we use in this thesis, is the called the direct assignment truncation and it has been used by Johnson and Willsky (2014), Bryant and Sudderth (2012), and Liang et al. (2007). As the name suggests, in the generative process for the topic model, we directly assign words to corpus-level topics. That is, the generative model is the same as Eq. (2.15) and we have the following variational distribution:

$$q(\theta, \beta, \pi, z) = q(\beta) \prod_{k=1}^{\infty} q(\theta_k) \prod_{d=1}^{D} q(\pi_d) \prod_{n=1}^{N_d} q(z_{dn}) \tag{2.22}$$

The individual distributions for variational distributions are:

$$q(\beta) = \delta_{\beta^*}(\beta), \qquad q(\theta_k|\lambda_k) = \text{Dir}(\theta_k|\lambda_k), \tag{2.23}$$
$$q(\pi_d|\nu_d) = \text{Dir}(\pi_d|\nu_d), \qquad q(z_{dn}|\phi_{dn}) = \text{Cat}(z_{dn}|\phi_{dn}),$$

where $\delta_{\beta^*}(\beta)$ is the degenerate distribution at the point β^* and $\lambda_k, \nu_d, \phi_{dn}$ are the variational parameters. We truncate $q(z_{dn})$, the topic indicator distribution for each word, at the threshold K. That is, $q(z_{dn} = k) = 0$ for $k > K$. With this assumption, we can ignore the topic distributions with indices greater than K and only update the remaining parameters. In other words, other factors $q(\beta), q(\theta_k)$ and $q(\pi_d)$ are different from their priors over the first K components. Hence, the variational parameters for $q(\pi_d)$ and $q(\beta)$ instead of countably infinite elements have $K+1$ elements: $q(\pi_d) = q((\pi_{d1}, \pi_{d2}, \cdots, \pi_{dK}, \pi_{d,\text{rest}})) = \text{Dir}(\nu_{d1}, \cdots, \nu_{d,K+1})$ and $q(\beta) = (\beta_1^*, \beta_2^*, \cdots, \beta_K^*, \beta_{\text{rest}}^*)$ where $\pi_{d,\text{rest}} = 1 - \sum_{k=1}^K \pi_{dk}$ and $\beta_{\text{rest}}^* = 1 - \sum_{k=1}^K \beta_k^*$.

The main advantage of the direct assignment truncation scheme is that it requires less bookkeeping. Another advantage is that as opposed to the stick-breaking truncation, the family of approximations is nested over K. This means an automatic truncation level adjustment can be done as proposed by Bryant and Sudderth (2012). However, the disadvantage of this approach is the update to $q(\beta)$ is not conjugate given the other factors. We simplified the updates by using a point estimate for β. It has been empirically shown (see Johnson and Willsky, 2014; Bryant and Sudderth, 2012; Liang et al., 2007) that the updates to the global topic weights have much less impact compared to the improvements to the topic distributions. One reason for this behavior could be that the main effect of β is to share sparsity across all document topics; hence, a point approximation for $q(\beta)$ can work reasonably well in practice for large datasets.

Updating the variational parameters requires taking the gradient of Eq. (2.5) with respect to the variational parameters. This yields the following update equations for ϕ_{dn}, λ_k, and ν_d:

$$\phi_{dnk} \propto \exp\{\mathbb{E}_q(\theta_k)[\log \theta_{kn}] + \mathbb{E}_q(\pi_d)[\log \pi_{dk}]\} \tag{2.24}$$

$$\lambda_{kn} \leftarrow \eta + \sum_{j=1}^{D} c_{n(j)} \phi_{jnk} \tag{2.25}$$

$$\nu_{dk} \leftarrow \alpha\beta_k + \sum_{n=1}^{N_d} c_{n(d)} \phi_{dnk} \tag{2.26}$$

where $c_{n(j)}$ is the number of times word n appears in document j. To update the β parameters, we

need to take the gradient of the variational objective Eq. (2.5) with respect to β^*:

$$\nabla_{\beta^*}\mathcal{L} = \nabla_{\beta^*}\left\{\mathbb{E}_{q(\pi)}\left[\ln\frac{p(\beta,\pi)}{q(\beta)q(\pi)}\right]\right\} \tag{2.27}$$

$$= \nabla_{\beta^*}\left\{\ln p(\beta^*) + \sum_{d=1}^{D}\mathbb{E}_{q(\pi_d)}\ln p(\pi_d|\beta^*)\right\}, \tag{2.28}$$

where from Eqs. (2.15) and (2.23) we have:

$$\frac{\partial}{\partial\beta_k^*}\ln p(\beta^*) = 2\sum_{i\geq k}\frac{1}{1-\sum_{j<i}\beta_j^*} - (\gamma-1)\sum_{i\geq k}\frac{1}{1-\sum_{j\leq i}\beta_j^*}, \tag{2.29}$$

and

$$\frac{\partial}{\partial\beta_k^*}\mathbb{E}_{q(\pi)}[\ln p(\pi_d|\beta^*)] = \gamma\psi(\nu_{dk}) - \gamma\psi(\nu_{d,K+1}) + \gamma\psi(\gamma\sum_{j=1}^{K+1}\beta_j^*) - \gamma\psi(\beta_k^*). \tag{2.30}$$

To update the gradients we use a backtracking line search so that for each gradient step we have $\beta^* \geq 0$.

CHAPTER 3

Segmenting user behavior traces with an efficient Bayesian nonparametric approach[1]

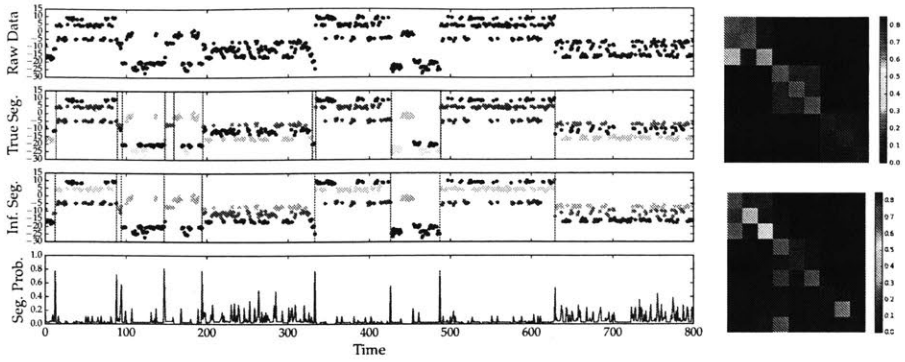
3.1 Introduction

User behavior traces collected from a software application may evolve at multiple timescales. For instance, in a photo editing application, the tasks such as filling part of an image or erasing part of it (see Fig. 1-2), evolve at the high-level. At the low-level and within the tasks, there are some non-exchangeable events. That is, to correctly perform each task, some order of the events needs to be preserved. In erasing part of an image, the eraser tool needs to be selected first, then the background color should be chosen and finally the tool should be dragged over the image.

Such hierarchically structured sequences characterized by multiple timescales is not limited to user trace segmentation and may arise in many domains, including natural language (Lee et al., 2013), handwriting (Lake et al., 2014), and motion recognition (Heller et al., 2009). For example, it is natural in motion recognition to model the sequence of high-level actions (such as walking to a chair, then sitting down) and steps within the actions (*e.g.,* bending one's knees then leaning back to sit down) at two different levels.

In this chapter, we will focus on the problem of *segmentation*, in which the goal is to identify points at which a time series transitions from one relatively stable regime to a new regime. In the photo editing example, the segmentation problem would be to identify when a user transitioned from one type of task (*e.g.,* filling part of an image) to another (*e.g.,* erasing part of an image), without necessarily identifying what they were doing. This is one of the easiest problems in time-series modeling that involves multiple timescales, but (as we will see) it is quite hard for (i)HMMs, which have no mechanism for distinguishing between high- and low-level dynamics.

[1]This chapter is based on the work by Saeedi et al. (2016) (http://proceedings.mlr.press/v48/saeedi16.pdf).



(a) (b)

Figure 3-1: **siHMM applied to a synthetic dataset.** (a) A sample of a synthetic dataset with true and inferred segmentations. Top three plots: the y-axis shows 1-d observations, colors denote true or inferred hidden state, vertical lines denote true or inferred segment boundaries. Bottom plot: inferred posterior probability of a segment boundary. (b) **Top** True and **Bottom** inferred transition matrices

The iHMM and its variants (*e.g.,* Fox et al. 2008; Saeedi and Bouchard-Côté 2011) have been among the most successful Bayesian nonparametric models, with applications from speech recognition (Fox et al., 2011) to biology (Beal and Krishnamurthy, 2012). However, despite their success in modeling time series with complicated low-level dynamics, their application to time series with multiple timescales has been limited.

The hierarchical HMM (HHMM) is a generalization of the HMM that naturally deals with dynamics at multiple timescales (Fine et al., 1998; Murphy and Paskin, 2002). But this generality comes at a price: these models lack the simple predictive distributions and efficient inference schemes that make (i)HMMs so popular. And the available nonparametric versions of the HHMM (*e.g.,* Heller et al., 2009; Stepleton et al., 2009) are complex to implement and not readily amenable to efficient inference.

We propose the segmented iHMM (siHMM), a simple extension to the iHMM that does not suffer from the above problems and can discover segment boundaries in time series with dynamics at two timescales. Unlike the HHMM, our model does not explicitly model higher-level state; instead, it assumes dynamics that evolve according to a standard iHMM except for occasional change-point events that kick the model into a new randomly chosen hidden state, disrupting the low-level dynamics of the iHMM. Because it relies on a very simple model of high-level dynamics, inference in the siHMM has time and implementation complexity similar to that of the iHMM, and well below that of a typical HHMM. We show that this simple change-point extension is sufficient to encourage the iHMM to model time-series data characterized by multiple regimes of low-level dynamics.

Although our model is limited by the depth of the hierarchy, in many practical applications of HHMMs (*e.g.,* Olivera et al. 2004; Nguyen et al. 2005; Xie et al. 2003) a two-level analysis of the dynamics is sufficient.

Below, we describe two versions of the model. The first version, which we call the feature-independent model, enjoys conditional conjugacy and therefore has simple Gibbs and variational inference algorithms. The second version, which we call the feature-based model, can incorporate domain knowledge without requiring a complex new machinery. We present a stochastic variational inference (SVI) scheme for the feature-based model; the derivation for the feature-independent model is similar and straightforward.

In addition to applying the model to segmenting user behavior traces, we apply the model to two different tasks: biometric sensor data labeling and automatic behavioral segmentation of fruit fly. As mentioned, segmenting user behavior traces is of significant importance in understanding the behavior of software application users; it can help in identifying and simplifying the complex common patterns among the users (*e.g.,* Adar et al. 2014; Han et al. 2007; Horvitz et al. 1998). Labeling sensor data gathered in everyday life settings can be used not only to understand physical activities (*e.g.,* Ermes et al. 2008; Pärkkä et al. 2006), but also to detect psychological and emotional states (*e.g.,* Picard et al. 2001; Healey and Picard 2005). Such labels can aid in implementing effective health and well-being related interventions, understanding user behavior, and designing affective interfaces. Finally, for the fruit fly behavior segmentation, we use a dataset from Kain et al. (2013); the results of this task can be used to better understand how the nervous system generates behavior.

We empirically compare our model with two main baselines: 1) a two-level Bayesian nonparametric hierarchical HMM (HHMM) introduced by Johnson (2014) that models high-level dynamics as an infinite hidden semi-Markov model (HSMM) and sub-dynamics as an iHMM, and 2) the iHMM. In each of these tasks, we show that our model outperforms the nonparametric HHMM (despite being substantially simpler and faster) and the iHMM.

3.2 Model

Our model can be viewed as a generalization of an iHMM where the transition probability from each state k is a mixture of two distributions: 1) a state-dependent transition probability distribution π_k, as in an iHMM, and 2) a state-independent probability distribution π_0. Which of these two distributions generates a hidden state z_t at a given time t depends on the hidden state z_{t-1} and observation y_{t-1} at the previous time $t-1$.

We say that the transitions caused by π_0 define the boundaries of a *segment*. The model implicitly assumes that the low-level dynamics within a segment are more structured and predictable than

the higher-level dynamics that govern transitions between segments, since it can throw much more modeling power at these low-level dynamics. For instance, In the user trace data in a photo editing application, the dynamics of a task may be highly structured and predictable, whereas the dynamics that govern whether a user transitions from filling part of an image to erasing or applying a filter may be much less predictable.

3.2.1 Feature-independent model

The feature-independent model assumes the following generative process. At time step $t = 0$, we initialize the process by sampling a hidden state z_0 from a distribution π_0. Given a hidden state z_t, we generate an observation y_t from a conditional observation distribution $F(\theta_{z_t})$ where θ_{z_t} is the parameter corresponding to the hidden state z_t: $y_t|z_t \sim F(\theta_{z_t})$.

Next, we sample a variable s_t, which we call the segmentation variable, from a Bernoulli distribution with a parameter ω_{z_t}. This is a state-dependent variable which has a conjugate beta prior with hyperparameters a_0 and b_0. Here, $s_t = 1$ denotes the beginning of a new segment:

$$\omega_i \sim \text{Beta}(a_0, b_0); \quad s_t|z_t, y_t \sim \text{Bern}(\omega_{z_t}).$$

We denote the probability of creating a new segment at time t by p_t^{seg}. If $s_t = 0$, we sample the next state z_{t+1} from a state-dependent distribution π_{z_t} (as in the iHMM), otherwise, we ignore the current state z_t and sample z_{t+1} from a distribution π_0:

$$z_{t+1} \mid z_t, s_t = 0 \sim \pi_{z_t}, \quad z_{t+1} \mid z_t, s_t = 1 \sim \pi_0.$$

The transition matrix has the same generative process as the iHMM:

$$\beta \sim \text{GEM}(\gamma); \qquad \pi_i \sim \text{DP}(\alpha, \beta); \qquad \theta_i \sim H;$$

where H is the prior distribution over θ, and γ and α are the concentration parameters of the stick-breaking and the Dirichlet process, correspondingly. The graphical model of the feature-independent siHMM is depicted in Fig. 3-2.

An illustration of the model applied to a synthetic dataset (explained in Section 3.5.1) is provided in Fig. 3-1. The model is able to approximately recover the block-diagonal structure of the true transition matrix. Even though the model does not explicitly encourage block-diagonal structure, the sparsity induced by the DP prior on π_k is sufficient to encourage the model to push inter-segment dynamics into π_0 and recover the block-diagonal intra-segment dynamics.

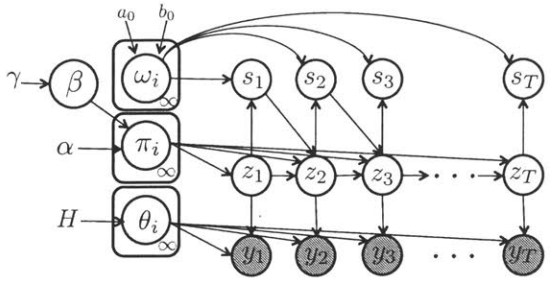


Figure 3-2: Graphical representation of the feature-independent siHMM.

3.2.2 Feature-based model

In some tasks such as segmenting software user traces or tagging fruit fly behavior, there is a rich domain knowledge available for improving the model. For instance, in segmenting user traces, features like $\mathbb{1}(\text{action} = \texttt{save})$ or $\mathbb{1}(\text{action} = \texttt{close})$ may indicate the end of a segment. We modify the model in a way that we can add features declaratively. Although due to lack of conjugacy, deriving the Gibbs sampler is not straightforward anymore, in Section 3.3, we derive an efficient SVI algorithm for this model.

The difference between this version of the model and the feature-independent version is in the conditional distribution of the segmentation variable s_t (see Fig. 3-3 for the graphical model). Here, the parameter of the Bernoulli distribution is

$$\sigma(\phi \cdot \mathrm{f}(y) + \omega_{z_t}) = \frac{1}{1 + \exp\left(-\phi \cdot \mathrm{f}(y) - \omega_{z_t}\right)}$$

where ϕ is the weight vector for the data-dependent features, $\mathrm{f}(y)$ is the feature function which consists of all observation-dependent features, and ω_{z_t} is the feature weight for hidden state z_t. To simplify the notation, we assume that the observation-dependent features only depend on the observation at a single time step. Hence, we have

$$s_t | z_t, y_t \sim \mathrm{Bern}(\sigma(\phi \cdot \mathrm{f}(y_t) + \omega_{z_t})).$$

We do not assume a prior for the feature weights; instead, we use a point estimate for them in our SVI algorithm.

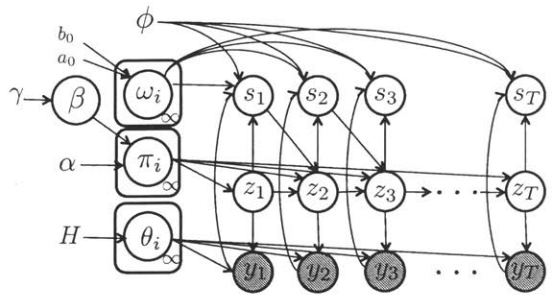


Figure 3-3: Graphical representation of the feature-based siHMM.

3.3 Stochastic variational inference

To keep the notation uncluttered, we assume that we have a dataset $\mathbf{y}$ of K sequences all with the same length T and write: $\mathbf{y} = \{y^k_{1:T}\}_{k=1}^K$, $\mathbf{z} = \{z^k_{1:T}\}_{k=1}^K$, $\mathbf{s} = \{s^k_{1:T}\}_{k=1}^K$. For inference, we use the stochastic variational inference (SVI) algorithm (Hoffman et al., 2013) and approximate the posterior with a truncated variational distribution introduced by Johnson and Willsky (2014). We approximate the posterior $p(\mathbf{z}, \mathbf{s}, \beta, \phi, \omega, \pi, \theta|\mathbf{y})$ with a mean-field family distribution $q(\mathbf{z}, \mathbf{s})q(\beta)q(\omega)q(\phi)q(\pi)q(\theta)$. In the language of SVI, $\mathbf{z}$ and $\mathbf{s}$ are local variables and β, ω, ϕ, π, and θ are global variables. We maximize the marginal likelihood lower bound $\mathcal{L}$:

$$\mathcal{L} \triangleq \mathbb{E}_q \left[\ln \frac{p(\mathbf{z}, \mathbf{s}, \beta, \omega, \phi, \pi, \theta, \mathbf{y})}{q(\mathbf{z}, \mathbf{s})q(\beta)q(\omega)q(\phi)q(\pi)q(\theta)} \right]$$

by using stochastic natural gradient ascent over the global factors and standard mean-field updates for the local factors. At each iteration of SVI, we sample a minibatch of M sequences from the dataset and update its local factors; next, given the expectation with respect to the local factors, we update the global factors by taking a step of size ρ in the approximate natural gradient direction. To further simplify the notation, we assume that the minibatch is a single sequence and drop the superscript for y, z and s. Next, we explain the variational factors for each of the variables.

3.3.1 Variational factors

For $q(z_{1:T}, s_{1:T})$, the "direct assignment" truncation used by Johnson and Willsky (2014), sets $q(z_{1:T}, s_{1:T}) = 0$ if for any of z_1 to z_T we have $z_t = k$ and $k > K$; here K is the truncation level. Since by using this truncation the update to $q(\beta)$ conditioned on the other factors is not conjugate anymore, similar to that we described in Section 2.2.4, we use a point estimate for $q(\beta)$: $q(\beta) = \delta_{\beta^*}(\beta)$. We adopt the same point estimate approach for the parameters of the sigmoid function; hence, $q(\phi) = \delta_{\phi^*}(\phi)$ and $q(\omega_z) = \delta_{\omega_z^*}(\omega_z)$.

With this truncation scheme, we can write the prior over π_i as $p((\pi_{i1}, \ldots, \pi_{iK}, \pi_{i,\text{rest}})) = \text{Dir}(\alpha\beta_1, \ldots, \alpha\beta_K, \alpha\beta_{\text{rest}})$. Here, $\pi_{i,\text{rest}} = 1 - \sum_{k=1}^{K} \pi_k$ and $\beta_{\text{rest}} = 1 - \sum_{k=1}^{K} \beta_k$. The optimal $q((\pi_{i1}, \ldots, \pi_{iK}, \pi_{i,\text{rest}}))$ is of the form of $\text{Dir}(\tilde{\alpha}_i)$ where $\tilde{\alpha}$ is the parameter of the variational distribution (Hoffman et al., 2013).

We assume that the prior over θ is in an exponential family with natural parameter η, and it is a conjugate prior for the likelihood function $f(y_t|\theta)$. This implies that the optimal variational distribution $q(\theta)$ is also in the same family with some other natural parameter denoted by $\tilde{\eta}$. More formally, we have: $h(\theta_i) \propto \exp\{\langle \eta_i, t_\theta(\theta_i) \rangle\}$ and $q(\theta_i) \propto \exp\{\langle \tilde{\eta}_i, t_\theta(\theta_i) \rangle\}$ where t_θ is the sufficient statistic function of $p(\theta)$.

3.3.2 SVI update equations

For the variational updates, we need to take expectations with respect to each of the variational distributions. For the expectations with respect to $q(z_{1:T}, s_{1:T})$, a modification of the standard HMM forward-backward algorithm with the following forward F and backward B messages can be used:

$$F(z_t, s_t) \triangleq f(y_t|\theta_{z_t}) p(s_t|z_t, y_t) \times \sum_{z_{t-1}, s_{t-1}} F(z_{t-1}, s_{t-1}) p(z_t|s_{t-1}, z_{t-1});$$

$$B(z_t, s_t) \triangleq \sum_{z_{t+1}, s_{t+1}} B(z_{t+1}, s_{t+1}) \times f(y_{t+1}|\theta_{z_{t+1}}) p(s_{t+1}|z_{t+1}, y_{t+1}) p(z_{t+1}|s_t, z_t).$$

These messages can be computed in $O(TK^2)$. In fact, the augmented transition matrix that we need to compute for these forward-backward messages has the following form:

$$\begin{array}{c} \\ s_t = 0 \\ s_t = 1 \end{array} \begin{array}{c} \begin{array}{cc} s_{t+1} = 0 & \quad s_{t+1} = 1 \end{array} \\ \left[\begin{array}{c|c} (1 - p^{\text{seg}}_{t+1})\Pi & p^{\text{seg}}_{t+1}\Pi \\ \hline (1 - p^{\text{seg}}_{t+1})\mathbf{1}\pi_0^T & p^{\text{seg}}_{t+1}\mathbf{1}\pi_0^T \end{array}\right] \end{array}$$

where Π is a $K \times K$ transition matrix and $\mathbf{1}$ is an all-ones vector of size K. The matrix operation for computing a message requires $2K^2$ operations for the upper half of the matrix ($s_t = 0$) and $2K + K$ for the lower half ($s_t = 1$). This is because all the rows of each $K \times K$ block in the lower half are the same. Hence, the total number of operations for message passing is $T(2K^2 + 3K)$. Furthermore, the total memory required to compute these operations is $O(TK)$.

For updating the local factors, instead of π_{ij} and $f(y_t|\theta_{z_t})$ in Eq. (3.1), we compute the forward-backward messages using: $\tilde{\pi}_{ij} \triangleq \exp\{\mathbb{E}_{q(\pi)} \ln \pi_{ij}\}$ and $\tilde{L}_{ti} \triangleq \exp\{\mathbb{E}_{q(\theta_i)} \ln f(y_t|\theta_{z_t}, z_t = i)\}$

The expectations of the sufficient statistics with respect to $q(z_{1:T}, s_{1:T})$ are:

$$\begin{aligned}
\tilde{t}^{ij}_{\text{trans}} &\triangleq \mathbb{E}_{q(z_{1:T},s_{1:T})} \sum_{t=2}^{T} \mathbb{1}[z_{t-1} = i, z_t = j, s_{t-1} = 0, s_t = .]; \\
\tilde{t}^{0j}_{\text{trans}} &\triangleq \mathbb{E}_{q(z_{1:T},s_{1:T})} \left[\mathbb{1}[z_1 = j] + \sum_{t=2}^{T} \mathbb{1}[z_t = j, s_{t-1} = 1] \right]; \\
\tilde{t}^{j}_{y} &\triangleq \mathbb{E}_{q(z_{1:T},s_{1:T})} \sum_{t=1}^{T} \mathbb{1}[z_t = j, s_t = .] t^{j}_{y}(y_t).
\end{aligned} \tag{3.1}$$

Given these expected sufficient statistics for S sequences and minibatch size M, we can write the update equations for the parameters of the global variational factors $q(\pi)$ and $q(\theta)$ with a scaling factor $m \triangleq S/M$:

$$\begin{aligned}
\tilde{\eta}_i &\leftarrow (1-\rho)\tilde{\eta}_i + \rho(\eta_i + m.\tilde{t}^{i}_{y}) \\
\tilde{\alpha}_i &\leftarrow (1-\rho)\tilde{\alpha}_i + \rho(\alpha_i + m.\tilde{t}^{i}_{\text{trans}}) \\
\tilde{\alpha}_0 &\leftarrow (1-\rho)\tilde{\alpha}_0 + \rho(\alpha_0 + m.\tilde{t}^{0}_{\text{trans}}).
\end{aligned}$$

For the global factors $q(\omega)$, $q(\phi)$ and $q(\beta)$, we use a point estimate; hence, we only need the gradient of $\mathcal{L}$ with respect to β^*, ϕ^* and ω^*. For ∇_{β^*} we follow the derivation in Section 2.2.4 and obtain the gradient to use in a truncated gradient step on β^*. To estimate ϕ^*, we have:

$$\begin{aligned}
\nabla_{\phi^*}\mathcal{L} &= \nabla_{\phi^*} \left\{ \mathbb{E}_{q(z_{1:T}s_{1:T})} \left[\ln \frac{p(\phi, z_{1:T}, s_{1:T}, y_{1:T})}{q(\phi)q(z_{1:T}s_{1:T})} \right] \right\} \\
&= \mathbb{E}_{q(z_{1:T},s_{1:T})} \left[\sum_{t=1}^{T} s_t \mathrm{f}(y_t) - p(s_t = 1 | y_t, z_t) \mathrm{f}(y_t) \right].
\end{aligned}$$

We have a similar equation for ω^*_z. This gradient computation has time complexity of $O(TK)$.

3.4 Related models

There are few models similar to our model in terms of extending iHMM to multiple timescales. Infinite hierarchical HMM (iHHMM), introduced by Heller et al. (2009), is a nonparametric model that allows the HHMM to have potentially unbounded depth. Hence, the model can infer the number of levels in the hierarchy. In iHHMM, the bottom level is the observed sequence and each level is a sequence of hidden variables dependent on the level above. As the authors suggested, more efficient inference algorithms are needed in order to make their model useful for practical applications.

The block-diagonal iHMM (Stepleton et al., 2009) is a generalization of the iHMM that assumes a nearly block-diagonal structure on the transition matrix of the iHMM. Each block corresponds to

a "sub-behavior" and the model can partition the data sequences according to these sub-behaviors. The model first partitions the infinite number of hidden states into an infinite number of blocks by using an additional stick-breaking process. Then, it increases the probability of transition between the states of a block by modifying the Dirichlet process prior over the transitions. Hence, as the block size becomes smaller, the model behavior converges to that of iHMM. For inference, as the authors explained, achieving a fast mixing rate in their proposed inference algorithm requires implementing a nontrivial bookkeeping-intensive method. In contrast, our model is much simpler and easier to implement inference for, but it can also discover transition matrices with approximately block-diagonal structure; the segmentation events provide a mechanism for transitioning from one group of connected states to another.

Another related model is a two-level Bayesian nonparametric HMM introduced by Johnson (2014) that models the high-level dynamics or the superstates as an HDP-HSMM. As a generalization of iHMM, HDP-HSMM can model the dwell time in each state by sampling that from a state-specific duration distribution once a state is entered. For a formal definition of the HDP-HSMM, see Johnson (2014). Given each superstate j, observations are generated according to an iHMM with parameters $\{\beta^j, \pi_k^j, \theta_k^j\}_{k=1}^{\infty}$. where x_t denotes the substate at time step t. Compared to our model, this model is much more flexible; however, the computation of forward-backward messages is less efficient. Moreover, it requires more bookkeeping for all superstates and their substates, and uses separate truncation levels for the superstates and each of the iHMMs correpsonding to them. (One can set a large truncation level for all of them, but this means paying a huge computational cost for iHMMs that only require a few states. In contrast, in the siHMM we only need to set a single truncation level for the whole model.) We refer to this model as the sub-iHMM and use it as a baseline since it is a flexible Bayesian nonparametric model that supports two-level dynamics.

Finally, the iHMM or sticky HDP-HMM can be used for segmentation by treating changes in hidden state as segment boundaries (Fox et al., 2011). In the sticky HDP-HMM, a self-transition bias encourages consecutive observations to be associated with the same state. The model can capture segments in tasks such as speaker diarization; however, within a given state there are no dynamics. In other words, in the sticky HDP-HMM, the observations within a segment are independent of each other. This makes the model less appropriate for segmenting sequences with predictable intra-segment dynamics. For instance, in a software application workflow an action like `selection` needs to come before an action like `move selection`. In our iHMM experiments, we include a "sticky" self-transition bias [2].

[2]We do not include a separate column in Table 3.2 for the sticky HPD-HMM; instead, we find the best variational lower bound for all settings of hyperparameters of the iHMM including a hyperparameter for self-transition bias.

3.5 Experiments

We evaluate the performance of our feature-independent and feature-based models on synthetic and real datasets. We use a synthetic dataset to illustrate the advantages of our model compared to baselines.

In order to further demonstrate the capabilities of our model, we apply it to three real tasks from human-computer interaction and two other domains: sensor data analysis, and biology. The properties of the datasets used for these tasks are provided in Table 3.1. In this table, the maximum number of labels is the total number of segment labels for each dataset [3]. All datasets are accompanied with the "gold standard" labels; hence, the ground truth is available and we can compute the labeling error of our model.

As mentioned in Section 3.4, two reasonable baselines for our model are the two-level Bayesian nonparametric HMM and iHMM. We report the labeling error (or normalized Hamming distance in the case of the synthetic dataset) and predictive log-likelihood for our model and these baselines. To choose among different hyperparameter settings, we use the variational lower bound (VLB) as our objective measure. We show that our model, while being simpler and efficient in terms of inference, is competitive with or outperforms these baselines. For all experiments on siHMM we try both the feature-independent and feature-based models; we report the results separately in Table 3.2 to show the effect of including observation features in the model. In the experiments, we only try the hidden state and the observation as the features for a given time step. However, more sophisticated features can be made from the observation(s). For the details of all experiments see the supplementary material.

3.5.1 Synthetic data

We generate a synthetic dataset with 5000 data points from 3 different transition matrices, each with 3 hidden states. Each row of each transition matrix is sampled from a modified $\mathrm{Dir}(1)$ with self-transition bias of 1. The observations are sampled from normal distributions with non-conjugate separate priors $\mathcal{N}(0, 10)$ and $\Gamma^{-1}(2, 1)$ on their mean and variance parameters, respectively. The goal is to find the points where we change regimes and also to determine the dynamics within each segment of the sequence. At each time step with probability 0.05, we switch the regime.

Fig. 3-1 shows a sample sequence and the result of running 100 passes of SVI over the whole dataset for that sequence. For running SVI, we split the dataset into 20 sequences and use a batch of size 2. We randomly sample a sequence with length 750 to calculate the predictive log-likelihood. We report the error over the dataset for the hyperparameter setting with the highest VLB.

[3] For the synthetic dataset, maximum number of labels is the number of hidden states used to generate the observations

We run SVI with 10 different seeds for 100 iterations over a set of hyperparameters (see the supplementary material for all the settings which we considered). To compute the normalized Hamming distance between the inferred states and the true states, we use the Munkres algorithm (Munkres, 1957). The algorithm assigns indices to the inferred sequence so that it maximizes the overlap with the true sequence. Table 3.2 gives the computed distance over the dataset for the hyperparameter setting with the highest VLB. We also report the predictive log-likelihood over the held-out set.

Fig. 3-4 shows the histogram of the normalized Hamming distance and also the predictive log-likelihood for runs with different hyperparameter settings. In terms of the Hamming distance, the siHMM performs slightly better than the iHMM and outperforms the sub-iHMM for most settings. The same conclusion holds for the predictive log-likelihood. Furthermore, Fig. 3-1 shows that our model can do reasonably well in finding the regime change points and also the states within each segment of the sequence. We choose a threshold of 0.5 for the posterior segmentation probability to identify a time point as a change point. This threshold is just a natural default; it can be tuned to trade off sensitivity and specificity. Finally, as shown in the last row of Fig. 3-1, the model is able to provide an estimate for the posterior probability of beginning a segment at time t.

Table 3.2 shows the normalized Hamming distance for the synthetic dataset; siHMM outperforms both iHMM and sub-iHMM. This might be because the synthetic dataset is specifically generated without any state-dependent high-level dynamics. Our model, which does not assume any dynamics for the segments, performs better than the other baselines which implicitly or explicitly assume that.

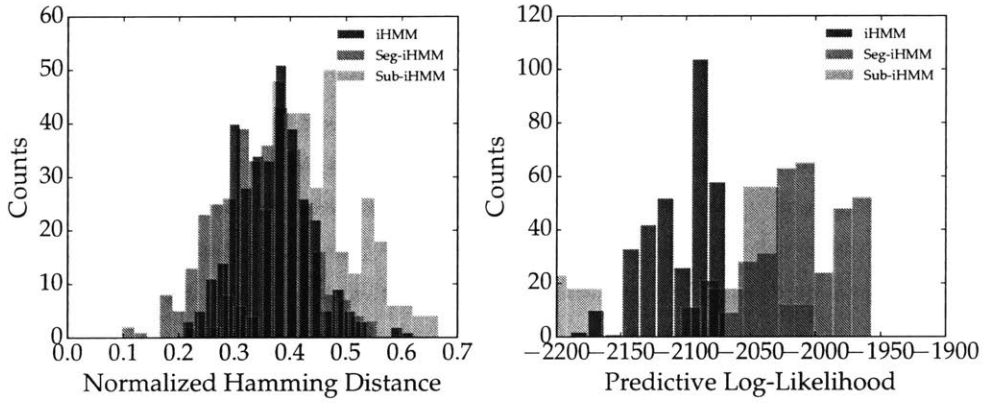


Figure 3-4: **Histogram of normalized Hamming distance and predictive log-likelihood.** These are over different settings of hyperparameters for the synthetic data set.

Dataset	# Points	Held-out	Max. # Labels
Synthetic	$5e^3$	15%	9
Users	$1.4e^4$	10 %	23
Sensors	$1.2e^4$	10 %	10
Drosophila	$1e^4$	15 %	12

Table 3.1: Datasets used for experiments (description in text)

	(Normalized Hamming Distance / Error% & Predictive LL)			
Data Set	iHMM	sub-iHMM	siHMM (WoF)	siHMM (WF)
Synthetic	$(0.21, -2.07 \times 10^3)$	$(0.23, -2.24 \times 10^3)$	$(0.15, -2.05 \times 10^3)$	$(\mathbf{0.13}, -2.19 \times 10^3)$
Users	$(30\%, -3.87 \times 10^3)$	$(24\%, -2.51 \times 10^3)$	$(25\%, -3.71 \times 10^3)$	$(\mathbf{16\%}, -3.52 \times 10^3)$
Sensors	$(27\%, -3.25 \times 10^3)$	$(22\%, -3.95 \times 10^3)$	$(\mathbf{18\%}, -3.21 \times 10^3)$	$(34\%, -3.35 \times 10^3)$
Drosophila	$(36\%, -6.58 \times 10^4)$	$(41\%, -6.7 \times 10^4)$	$(37\%, -7.18 \times 10^4)$	$(\mathbf{34\%}, -6.71 \times 10^4)$

Table 3.2: **Labeling error and predictive log-likelihood for various different datasets. Key:** iHMM = infinite HMM; sub-iHMM = a two level hierarchical infinite HMM; siHMM (WoF) = feature-independent siHMM; siHMM (WF) = feature-based siHMM; Synthetic = synthetic data segmentation and hidden state inference tasks for which we report the normalized Hamming distance instead of error rate; Users = user trace segmentation task; Sensors = labeling sensor data task; Drosophila = segmenting fruit fly behavior task.

3.5.2 Segmenting user behavior traces

Having a good understanding of the tasks done by users can potentially help in designing better workflows in software applications. As a reminder, log files of software applications contain user actions and their corresponding time-stamps; however, it is not clear only from these log files how many different tasks have been done by a user in a single work session. A task consists of multiple events and each work session consists of multiple tasks. Our two-level hierarchical model can be used for detecting the boundaries between tasks (*i.e.,* segments). We believe that for the software applications in which the tasks are less predictable compared to the actions within each task, our model is a good fit.

We collect log files of users who follow 23 different tutorials in a photo editing software. The dataset contains 14000 data points and 59 unique events in total. The events are directly read from the users log files and an action at time t is the observation at that time point. We randomly choose a sequence of size 1400 and form a held-out set. We split the sequences into subsequences of size 1000 and apply 100 passes of SVI to the dataset with both feature-based and feature-independent models.

The labels (*i.e.,* the tutorial numbers that the user followed) for each segment of the dataset are available; hence, we can test the siHMM on predicting the labels for each segment. This task is more

involved than segmentation, as we also need to group the substates within the inferred segments. We use a simple K-means clustering on the empirical transition matrix which is generated from counting the transitions in each inferred segment. This approach works well in practice; however, more sophisticated methods are possible for grouping the substates. Table 3.2 provides the prediction error (computed using the Munkres algorithm) for siHMM and the baselines. The performance of our feature-independent model is significantly better than iHMM and comparable with that of the more flexible (and also computationally intensive) sub-iHMM. Adding the observation feature to the model reduces the error to 16%. As mentioned in Section 3.2.2, in this dataset there are observations that can signal a change-point in the dynamics.

3.5.3 Segmenting and classifying human behavior from sensor data

Through the emergence of pervasive computing and affordable wearable sensors, in-situ measurement of different bio-signals has become possible. This powerful source of data can be utilized for several purposes, including activity recognition and task identification. Toward this goal, an efficient algorithm for analyzing this large amount of data – which is gathered 24/7 – is essential. In this section, we use siHMM to model the data collected via Empatica E4 wristband (Empatica, 2015), a wearable device that can collect Electrodermal activity (EDA) (Boucsein, 2012), blood volume pulse (BVP), acceleration, and body temperature. EDA refers to changes in electrical properties of the skin caused by sudomotor innervation (Boucsein, 2012). EDA is an indication of physiological or psychological arousal and has been utilized to objectively measure affective phenomena and sleep quality (Sano et al., 2015).

Segmenting the sensor data can help psychophysiological activity recognition. For instance, it can help in finding stressful periods objectively in order to detect the roots of stress in a person's lifestyle. However, manual labeling for large amounts of user data (days or months) is time-consuming and even invalid if not reported in a timely manner.

We use a dataset with 12000 time steps, collected from a single user, and model the (normalized) observations (*i.e.,* EDA, BVP and acceleration in 3 dimensions) by a multivariate Gaussian distribution. The hyperparameter setting is similar to that of Section 3.5.4.

The labeling error in Table 3.2 shows that feature-independent siHMM, while performing comparably to sub-iHMM, outperforms iHMM by a relative error reduction of 50%. Section 3.5.3 demonstrates the inferred segments for siHMM and the baselines. It seems that the single observation feature that we are using for the experiment does not help in this dataset; however, more sophisticated features may help improving the segmentation.

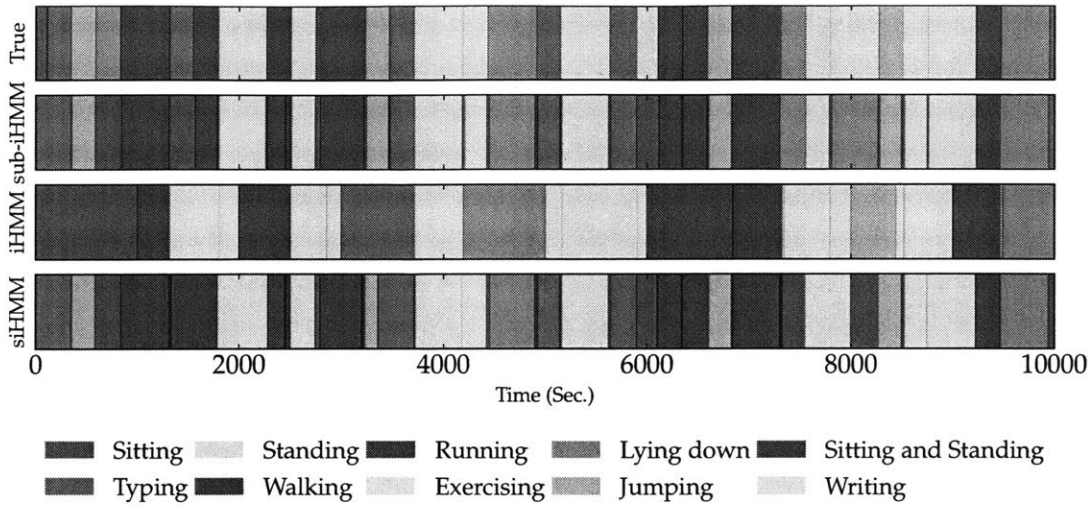


Figure 3-5: **Segments and labels for data collected from sensors.** From top to bottom: True, sub-iHMM, iHMM, siHMM.

3.5.4 Segmenting fruit fly behavior

Automating scientific experiments on live animals has attracted significant attention recently (see, for instance, Kain et al. 2013; Wiltschko et al. 2015; Crall et al. 2015; Freeman et al. 2014). With the advent of high throughput and more accurate devices, the need for automatic analysis of large amounts of collected data is felt more today. In neuroscience and biology, a large amount of behavioral data is collected from live animals in order to understand how the brain generates activity and how the underlying mechanisms have evolved (Kain et al., 2013). Typically the first step in analyzing this data, is finding and categorizing different types of behavior; this step can be done manually by experts but it is time-consuming and sometimes error-prone.

An automatic framework has been proposed by Kain et al. (2013) for tracking the leg movements and classifying the behavior of fruit flies. The behavior is recorded by tracking each leg of a fruit fly moving upon a track ball. The collected raw data is the x and y coordinates of 6 legs and the three rotational components of the rotating ball (i.e., a 15-dimensional vector in real time). After some post-processing and adding some higher-order features (e.g., derivatives of each of the 15 raw data vectors), they expand the dimensions to 45 and apply a KNN classifier to classify each frame as a part of 12 possible behavioral labels. Our goal is to use this dataset and categorize the frames in an unsupervised way. A frequent assumption in the behavioral sciences is that a small set of stereotyped motifs describe most animal activities (Berman et al., 2014). In other words, actions within a behavioral segment (e.g., actions required for grooming) should be more structured, compared to the behavioral segments themselves. Given the capabilities of the siHMM,

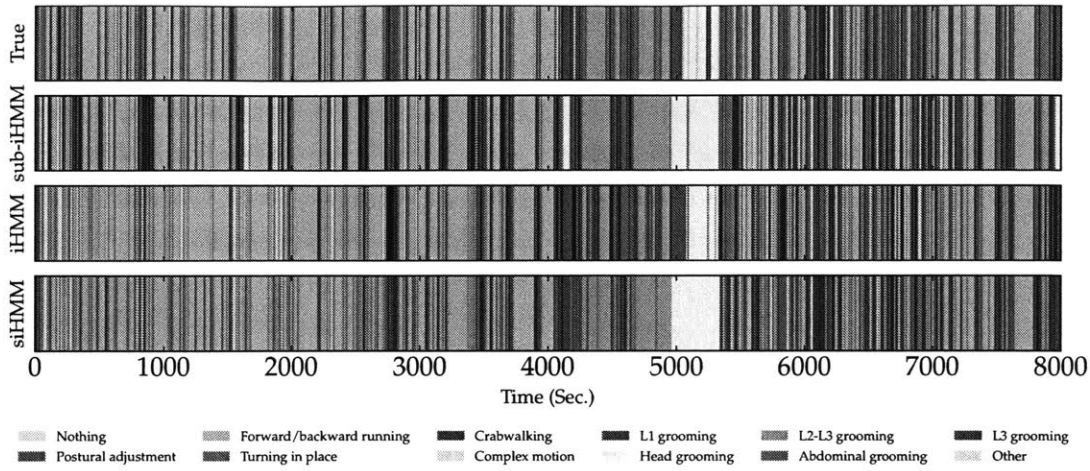


Figure 3-6: **A sample segmentation from the fruit fly dataset.** From top to bottom: True, sub-iHMM, iHMM, siHMM.

it is a reasonable choice for applying to this dataset.

The dataset contains 10000 data points; our held-out set is a randomly chosen subsequence with length 1500, and we apply SVI for 100 passes over both the feature-independent and feature-based models. For the observations, we use a multivariate Gaussian likelihood and a conjugate Normal/inverse-Wishart prior.

As in Section 3.5.2, we group the inferred substates with K-means and assign labels to the segments. The results, presented in Table 3.2, show that feature-based siHMM performs on a par with the iHMM and outperforms sub-iHMM by a relative error reduction of 17%. This may emphasize the importance of adding data-driven features to the model. Fig. 3-6, shows a sample of the dataset and its segmentation by different methods.

3.6 Conclusion

We proposed a new Bayesian nonparametric model, siHMM, for modeling dynamics at two timescales in time series. Our model is a simple extension to the widely used iHMM and has an efficient inference scheme. Although our model is less flexible than other nonparametric models for hierarchical time series, we showed that it can perform reasonably well in practice. One potential application of our model is using the inferred state-independent transition vector (π_0) for summarizing a sequence. For instance, in the user behavior analysis, this vector may represent a user fingerprint and users can be grouped based on it. For a better understanding of this feature and the behavior of our model in

other applications, a more comprehensive comparison with other models is useful.

CHAPTER 4

Markov jump processes for modeling user behavior traces[1]

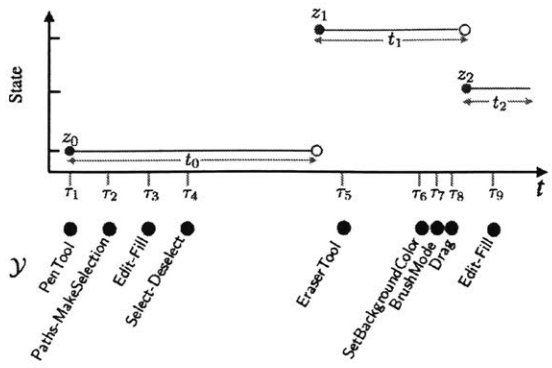
4.1 Introduction

In Chapter 3, we ignored the fact that the events in the log files occur at irregular time intervals. Furthermore, we assumed they happen at discrete time points. These unrealistic assumptions helped us in developing a tractable model for user trace segmentation. However, a more realistic setting would be a continuous-time generative model for the latent variables in which we explicitly model the duration time in each task and switching between tasks can happen at any time point between the observations. In this chapter, we focus on Markov Jump Processes (MJPs) which are simple models with these properties.

MJPs are continuous-time, discrete-state Markov processes in which state durations are exponentially distributed according to state-specific rate parameters. A stochastic matrix controls the probability of transitioning between pairs of states.

MJPs have been used to construct probabilistic models either when the state of a system is observed directly, such as with disease progression (Mandel, 2010) and RNA path folding (Hajiaghayi et al., 2014), or when the state is only observed indirectly, as in corporate bond rating (Bladt and Sørensen, 2009). In analyzing user behavior data, similar to the model in Chapter 3, segmenting the trace can be done by inferring the latent state of the user. Once the user enters a new state the software application can guide the user by proposing relevant tools for the new state. In contrast to the model in Chapter 3, we do not assume any dynamics in each segment; hence, our model is not hierarchical. That is a simplifying assumption but considering a hierarchical structure for MJPs will make our algorithms harder to scale to large datasets; we leave this setting for future work.

[1]This chapter is based on the work by Huggins et al. (2015) (http://proceedings.mlr.press/v37/hugginsa15.pdf).



Notation
M: number of states
π: initial state distribution
P: state transition matrix, with entries $p_{zz'}$
λ_z: transition rate for state z
$\mathcal{U} = (z_0, t_0, z_1, t_1, \ldots, z_{K-1}, t_{K-1}, z_K)$: MJP trajectory
$\mathcal{Z}$: the states corresponding to $\mathcal{U}$
$\mathcal{T}$: the times corresponding to $\mathcal{U}$
$\mathcal{O} = \{(\tilde{t}_i, \tilde{z}_i)\}$: observation times and states of DOMJP*
$\boldsymbol{\tau} = (\tau_1, \ldots, \tau_L)$: observation times of HMJP
$\mathcal{Y} = (y_1, \ldots, y_L)$: observations of HMJP
ρ_{zn}: probability of observing $y_\ell = n$ when in state z

Figure 4-1: **An example of and notation used in HMJPs. Left:** Illustrative example for an HMJP (Section 4.3.2) with three hidden states ($M = 3$) and two possible observation values ($N = 2$). The observations $\mathcal{Y}$, their times $\boldsymbol{\tau}$, an (arbitrary) sample MJP trajectory $\mathcal{U} = (z_0, t_0, z_1, t_1, z_2, t_2)$. **Right:** Notation used for parametric MJPs. *DOMJP = directly observed MJP.

In addition to user modeling, datasets with continuous-time observations and potentially discrete latent states can be found in various domains. For example, consider the important clinical task of analyzing physiological signals of a patient in order to detect abnormalities. Such signals include heart rate, blood pressure, respiration, and blood oxygen level. For an ICU patient, an abnormal state might be the precursor to a cardiac arrest event while for an epileptic, the state might presage a seizure (Goldberger et al., 2000). How can the latent state of the patient be inferred by a Bayesian modeler, so that, for example, an attending nurse can be notified when a patient enters an abnormal state? MJPs offer one attractive approach to analyzing such physiological signals.

Applying an MJP model to user traces or physiological signals presents a challenge: the number of states is unknown and must be inferred using, for example, Bayesian nonparametric methods. However, efficient inference in nonparametric MJP models is a challenging problem, where existing methods based on particle MCMC scale poorly and mix slowly (Saeedi and Bouchard-Côté, 2011). Current optimization-based methods such as expectation maximization (EM) are inapplicable if the state size is countably infinite; hence, they cannot be applied to Bayesian nonparametric MJP models, as we would like to do for physiological signals.

Furthermore, although MJPs are viewed as more realistic than their discrete-time counterparts in many fields (Rao and Teh, 2013), degenerate solutions for the maximum likelihood (ML) trajectories for both directly and indirectly observed cases (Perkins, 2009), and non-existence of the ML transition matrix (obtained from EM) for some indirectly observed cases (Bladt and Sørensen, 2009)

present inferential challenges. Degenerate ML trajectories occur when some of the jump times are infinitesimal, which severely undermines the practicality of such approaches. For instance, a trajectory which predicts a patient's seizure for an infinitesimal amount of time is of limited use to the medical staff. Fig. 4-3 shows an example of the degeneracy problem.

In this chapter, we take a *small-variance asymptotics* (SVA) approach to develop an optimization-based framework for efficiently estimating the most probable trajectories (states) for both parametric and nonparametric MJP-based models. Small-variance asymptotics has recently proven to be useful in estimating the parameters and inferring the latent states in rich probabilistic models. SVA extends the well-known connection between mixtures of Gaussians and k-means: as the variances of the Gaussians approach zero, the *maximum a posteriori* solution to the mixture of Gaussians model degenerates to k-means solution (Kulis and Jordan, 2012). The same idea can be applied to obtain well-motivated objective functions that correspond to a latent variable model for which scalable inference via standard methods like MCMC is challenging. SVA has been applied to (hierarchical) Dirichlet process mixture models (Kulis and Jordan, 2012; Jiang et al., 2012), Bayesian nonparametric latent feature models (Broderick et al., 2013), hidden Markov models (HMMs), and infinite-state HMMs (Roychowdhury et al., 2013b).

We apply the SVA approach to both parametric and Bayesian nonparametric MJP models to obtain what we call the *JUMP-means* objective functions. In the parametric case, we derive a novel objective function which does not suffer from maximum likelihood's solution degeneracy, leading to more stable and robust inference procedures in both the directly observed and hidden state cases. Infinite-state MJPs (iMJPs) are constructed from the hierarchical gamma-exponential process (HΓEP) (Saeedi and Bouchard-Côté, 2011). In order to apply SVA to iMJPs, we generalize the HΓEP to obtain the first deterministic procedure (we know of) for inference in Bayesian nonparametric MJPs.

We evaluate JUMP-means on several synthetic and real-world datasets in both the parametric and Bayesian non-parametric cases. JUMP-means performs on par with or better than existing methods, offering an attractive speed-accuracy tradeoff. We obtain significant improvements in the non-parametric case, gaining up to a 20% reduction in mean error on the task of observation reconstruction. In summary, the JUMP-means approach leads to algorithms that 1) are applicable to MJPs with Bayesian nonparametric priors; 2) provide non-degenerate solutions for the most probable trajectories; and 3) are comparable to or outperform other standard methods of inference both in terms of speed and reconstruction accuracy.

4.2 Background

4.2.1 Markov jump processes

A *Markov jump process* (MJP) is defined by (a) a finite (or countable) state space, which we identify with the integers $[M] \triangleq \{1, \ldots, M\}$; (b) an initial state probability distribution π; (c) a (stochastic) state transition matrix P with $p_{zz} = 0$ for all $s \in [M]$; and (d) a state dwell-time rate vector $\boldsymbol{\lambda} \triangleq (\lambda_1, \ldots, \lambda_M)$. The process begins in a state $z_0 \sim \pi$. When the process enters a state s, it remains there for a dwell time that is exponentially distributed with parameter λ_z. When the system leaves state s, it transitions to state $z' \neq z$ with probability $p_{zz'}$.

A *trajectory* of the MJP is a sequence of states and a dwell time for each state, except for the final state: $\mathcal{U} \triangleq \mathcal{U}_T \triangleq (z_0, t_0, z_1, t_1, \ldots, z_{K-1}, t_{K-1}, z_K)$. Implicitly, K (and thus $\mathcal{U}$) is a random variable such that $t_{K-1} < T$ and the system is in state z_K at time T. Let $\mathcal{Z} \triangleq \mathcal{Z}_T \triangleq (z_0, z_1, \ldots, z_K)$ and $\mathcal{T} \triangleq \mathcal{T}_T \triangleq (t_0, t_1, \ldots, t_{K-1})$ be the sequences of states and times corresponding to $\mathcal{U}$. The probability of a trajectory is given by

$$p(\mathcal{U} \mid \pi, P, \boldsymbol{\lambda}) = \mathbb{1}[t_\cdot \leq T] e^{-\lambda_{z_K}(T - t_\cdot)} \pi_{z_0} \times \prod_{k=1}^{K} \lambda_{z_{k-1}} e^{-\lambda_{z_{k-1}} t_{k-1}} p_{z_{k-1} z_k}, \tag{4.1}$$

where $t_\cdot \triangleq \sum_{k=0}^{K-1} t_k$ and $\mathbb{1}[\cdot]$ is the indicator function. In many cases when the states are directly observed, the initial state and the final state are observed, in which case it is straightforward to obtain a likelihood from Eq. (4.1).

A hidden state MJP (HMJP) is an MJP in which the states are observed indirectly according to a likelihood model $p(y \mid z)$, $z \in [M]$, $y \in Y$, where Y is some observation space. The times of the observations $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_L)$ are chosen independent of $\mathcal{U}$, so the probability of the observations $\mathcal{Y} \triangleq (y_1, \ldots, y_L)$ is given by $p(\mathcal{Y} \mid \mathcal{U}, \boldsymbol{\tau}) = \prod_{\ell=1}^{L} p(y_\ell \mid z_{\tau_\ell})$, where, with an abuse of notation, we write z_τ for the state of the MJP at time τ.

4.2.2 Previous approaches to MJP inference

There are a number of existing approaches to inference and learning in MJPs. An expectation-maximization (EM) algorithm can be derived, but it cannot be applied to models with countably infinite states, so it is not suitable for iMJPs (Lange, 2014) (iMJPs are detailed in Section 4.4). Moreover, with discretely observed data, the maximum-likelihood estimate with finite entries for the transition matrix obtained from EM may not exist (Bladt and Sørensen, 2005).

Maximum likelihood inference amounts to finding $\max_{\mathcal{U}} \ln p(\mathcal{U} \mid \pi, P, \boldsymbol{\lambda})$, which can be carried out efficiently using dynamic programming (Perkins, 2009). However, maximum likelihood solutions for the trajectory are degenerate: only an infinitesimal amount of time is spent in each state, except for the state visited with the smallest rate parameter (i.e., longest expected dwell time). Such

a solution is unsatisfying and unintuitive because the dwell times are far from their expected values. Thus, maximum likelihood inference produces results that are unrepresentative of the model behavior.

Markov chain Monte Carlo methods have also been developed, but these can be slow and their convergence is often difficult to diagnose (Rao and Teh, 2013). Recently, a more efficient Monte Carlo method was proposed by Hajiaghayi et al. (2014) which is based on particle MCMC (PMCMC) methods (Andrieu et al., 2010b). This approach addresses the issue of efficiency, but since it marginalizes over the jump points, it cannot provide probable trajectories.

4.2.3 Small-variance asymptotics

Consider a Bayesian model $p(D \mid Z, \theta, \sigma^2)p(Z, \theta)$ in which the likelihood terms contain a variance parameter σ^2. Given some data D, a point estimate for the parameters θ and latent variables Z of the model can obtained by maximizing the posterior $p(Z, \theta \mid D, \sigma^2) \propto p(D \mid Z, \theta, \sigma)p(Z, \theta)$, resulting in a *maximum a posteriori* (MAP) estimate. In the SVA approach (Broderick et al., 2013), the MAP optimization is considered in the limit as the likelihood variance parameter is taken to zero: $\sigma^2 \to 0$. Typically, the small-variance limit leads to a much simpler optimization than the MAP optimization with non-zero variance. For example, the MAP objective for a Gaussian mixture model simplifies to the k-means objective.

4.3 Parametric MJPs

4.3.1 Directly observed MJP

Consider the task of inferring likely state/dwell-time sequences given $\mathcal{O} = \{(\tilde{t}_i, \tilde{z}_i)\}_{i=1}^{I}$, the times at which the system was directly observed and the states of the system at those times. For simplicity we assume that $\tilde{t}_0 = 0$ and that all times are in the interval $[0, T]$. Let $z(\mathcal{U}, t)$ be the state of the system following trajectory $\mathcal{U}$ at time t. The likelihood of a sequence is

$$\ell(\mathcal{U} \mid \mathcal{O}, P, \boldsymbol{\lambda}) = \mathbb{1}[t_\cdot \leq T] \prod_{i=1}^{I} \mathbb{1}[s(\mathcal{U}, \tilde{t}_i) = \tilde{z}_i] \times \left(\prod_{k=1}^{K} \lambda_{z_{k-1}} e^{-\lambda_{z_{k-1}} t_{k-1}} p_{z_{k-1} z_k} \right) e^{-\lambda_{z_K}(T - t_\cdot)} \tag{4.2}$$

We also place a gamma prior on the rate parameters $\boldsymbol{\lambda}$ (detailed below). Instead of relying on MAP estimation, we apply a small variance asymptotics analysis to obtain a more stable objective function. Following Jiang et al. (2012), we scale the distributions by an inverse variance parameter β and then maximize the scaled likelihood and prior in the limit $\beta \to \infty$ (i.e., as the variance goes to zero).

To obtain the SVA objective, we begin by scaling the exponential distribution $f(t;\lambda) = \lambda \exp(-\lambda t)$, which is an exponential family distribution with natural parameter $\eta = -\lambda$, log-partition function $\psi(\eta) = -\ln(-\eta)$, and base measure $\nu(\mathrm{d}t) = 1$ (Banerjee et al., 2005). To scale the distribution, introduce the new natural parameter $\tilde{\eta} = \beta\eta$ and log-partition function $\tilde{\psi}(\tilde{\eta}) = \beta\psi(\tilde{\eta}/\beta)$. The new base measure $\tilde{\nu}(\mathrm{d}t)$ is uniquely defined by the integral equation (see Banerjee et al., 2005, Theorem 5)

$$\int \exp(\tilde{\eta}t)\tilde{\nu}(\mathrm{d}t) = \exp(\tilde{\psi}(\tilde{\eta})) = \exp(-\beta \ln(\tilde{\eta}/\beta)) = \frac{\beta^\beta}{\tilde{\eta}^\beta}.$$

Choosing $\tilde{\nu}(\mathrm{d}t) = \frac{t^{\beta-1}\beta^\beta}{\Gamma(\beta)}\mathrm{d}t$ satisfies the condition, so we have

$$\begin{aligned} f(t;\lambda,\beta) = \frac{(\beta\lambda)^\beta}{\Gamma(\beta)} t^{\beta-1} e^{-\beta\lambda t} &= \exp(-\beta\lambda t + (\beta-1)\ln t + \beta \ln \lambda\beta - \ln\Gamma(\beta)) \\ &= \exp\left\{-\beta\left(\lambda t - \ln t - \ln\lambda - \frac{\beta\ln\beta - \ln\Gamma(\beta)}{\beta} + \frac{\ln t}{\beta}\right)\right\}. \end{aligned}$$

It can now be seen that $f(t;\lambda,\beta)$ is the density of a gamma distribution with shape parameter β and rate parameter $\beta\lambda$. Hence, the mean of the scaled distribution is $\frac{1}{\lambda}$ and its variance is $\frac{1}{\lambda\beta}$. Letting $F(t;\lambda,\beta)$ denote the CDF corresponding to $f(t;\lambda,\beta)$, we have $1 - F(t;\lambda,\beta) = \frac{\Gamma(\beta,\beta\lambda t)}{\Gamma(\beta)}$, where $\Gamma(\cdot,\cdot)$ is the upper incomplete gamma function.

The multinomial distribution is scaled by the parameter $\hat{\beta} \triangleq \xi\beta$. Writing the likelihood with the scaled exponential families (and dropping indicator variables) yields:

$$\begin{aligned} \ell(\mathcal{U}|\mathcal{O},P,\boldsymbol{\lambda}) \propto \exp\Bigg\{-\beta\Bigg(&\frac{\ln\Gamma(\beta) - \ln\Gamma(\beta,\beta\lambda_{z_K}t.)}{\beta} \\ &+ \sum_{k=0}^{K-1}\left(\xi \ln p_{z_k z_{k+1}} + \lambda_{z_k}t_k - \ln\lambda_{z_k}t_k\right) \\ &+ \sum_{k=0}^{K-1}\left(-\frac{\beta\ln\beta - \ln\Gamma(\beta)}{\beta} + \frac{\ln t_k}{\beta}\right)\Bigg)\Bigg\}. \end{aligned}$$

The modified likelihood is for a jump process which is no longer Markov when $\beta \neq 1$. We also place a $\mathrm{Gam}(\alpha_\lambda, \alpha_\lambda\mu_\lambda)$ prior on each λ_i and set $\alpha_\lambda = \xi_\lambda\beta$.

For the state at the k-th jump we use a 1-of-M representation; that is, z_k is an M-dimensional binary random variable which satisfies $z_{km} \in \{0,1\}$ and $\sum_{m=1}^M z_{km} = 1$. Hence, we have:

$$p(z_k|z_{k-1,j}=1) = \prod_{m=1}^M p_{jm}^{z_{km}}. \tag{4.3}$$

Given the Bregman divergence for a multinomial distribution, $d_\phi(z_k, \bar{p}_j) = \mathrm{KL}(z_k||\bar{p}_j)$ where $\bar{p}_j \triangleq (p_{j1}, \ldots, p_{jM})$, this can be written in terms of exponential family notation in the following form (Banerjee et al., 2005):

$$p(z_k|z_{k-1,j} = 1) = b_\phi(z_k) \exp(-d_\phi(z_k, \bar{p}_j)) \tag{4.4}$$

where $b_\phi(z_k) = 1$. For a scaled multinomial distribution we have $b_{\hat{\beta}\phi}(z_k) \exp(-\hat{\beta} d_\phi(z_k, \bar{p}_j))$, where $\hat{\beta} = \xi\beta$ is the scaling parameter for the multinomial distribution. Writing the trajectory probility with the scaled exponential families yields[2] :

$$\begin{aligned} p(\mathcal{U}|z_0, z_K, P, \boldsymbol{\lambda}) \propto \exp\Bigg\{ -\beta \Bigg(& \frac{\ln\Gamma(\beta) - \ln\Gamma(\beta, \beta\lambda_{z_K} t.)}{\beta} + \xi \sum_{k=0}^{K-1} \mathrm{KL}(z_{k+1}||\bar{p}_{z_k}) \\ & + \sum_{k=0}^{K-1} \left(\lambda_{z_k} t_k - \ln \lambda_{z_k} t_k - \frac{\beta \ln \beta - \ln\Gamma(\beta)}{\beta} + \frac{\ln t_k}{\beta} \right) \Bigg) \Bigg\}, \end{aligned} \tag{4.5}$$

Since $\beta \to \infty$, we can apply the asymptotic expansions for $\Gamma(\cdot)$ and $\Gamma(\cdot, \cdot)$. In particular, applying Stirling's formula and the facts in (DLMF) we have:

$$\begin{aligned} \frac{\beta \ln \beta - \ln\Gamma(\beta)}{\beta} &= \frac{\beta \ln \beta - \beta \ln \beta + \beta + o(\beta)}{\beta} \to 1 \\ \frac{\ln\Gamma(\beta) - \ln\Gamma(\beta, \beta\lambda t)}{\beta} &= \begin{cases} \frac{-\beta - o(\beta) - \beta \ln \lambda t + \beta\lambda t}{\beta} \to \lambda t - \ln \lambda t - 1 & \text{if } t \geq \frac{1}{\lambda} \\ \frac{\beta \ln \beta - \beta - \beta \ln \beta + \beta + o(\beta)}{\beta} \to 0 & \text{if } t < \frac{1}{\lambda} \end{cases} \end{aligned}$$

We also place a $\mathrm{Gam}(\alpha_\lambda, \alpha_\lambda \mu_\lambda)$ prior on each λ_i. With $\alpha_\lambda = \xi_\lambda \beta$, we obtain

$$\begin{aligned} \ln p(\lambda_s \,|\, \alpha_\lambda, \alpha_\lambda\mu_\lambda) &= \alpha_\lambda \ln(\alpha_\lambda\mu_\lambda) + (\alpha_\lambda - 1) \ln \lambda_s - \ln\Gamma(\alpha_\lambda) - \alpha_\lambda\mu_\lambda\lambda_s \\ &= \xi_\lambda \beta \ln \lambda_s - \xi_\lambda \mu_\lambda \beta \lambda_s + \xi_\lambda \beta + o(\beta) \\ &= -\beta(\xi_\lambda \mu_\lambda \lambda_s - \xi_\lambda \ln \lambda_s - 1) + o(\beta). \end{aligned}$$

Hence, when $\beta \to \infty$, obtain

$$\begin{aligned} \min_{\mathcal{U}, \boldsymbol{\lambda}, P} \Bigg\{ & \xi \sum_{k=0}^{K-1} \mathrm{KL}(z_{k+1}||\bar{p}_{z_k}) + \sum_{k=0}^{K-1} (\lambda_{z_k} t_k - \ln \lambda_{z_k} t_k - 1) \\ & + \mathbb{1}[\lambda_{z_K} t. \geq 1](\lambda_{z_K} t. - \ln \lambda_{z_K} t. - 1) + \xi_\lambda \sum_{s=1}^{M} (\mu_\lambda \lambda_s - \ln \lambda_s - 1) \Bigg\} \end{aligned} \tag{4.6}$$

The optimization problem Eq. (4.6) is very natural and offers far greater stability than maximum

[2]This derivation is done by Jonathan Huggins a coauthor in Huggins et al. (2015).

likelihood optimization. As with maximum likelihood, the $\ln p_{z_k z_{k+1}}$ terms penalize transitions with small probability. The term $h(t_k) \triangleq \lambda_{z_k} t_k - \ln \lambda_{z_k} t_k - 1$ is convex and minimized when $t_k = 1/\lambda_{z_k}$, the expected value of the dwell time for state z_k. As $t_k \to 0$, $h(t_k)$ approaches ∞, while for $t_k \gg 1/\lambda_{z_k}$, $h(t_k)$ grows approximately linearly. Thus, times very close to zero are heavily penalized while times close to the expected dwell time are penalized very little. The term $\mathbb{1}[\lambda_{z_K} t_\cdot \geq 1](\lambda_{z_K} t_\cdot - \ln \lambda_{z_K} t_\cdot - 1)$ penalizes the time $t_\cdot$ spent in state z_k so far in the same manner as a regular dwell time when $t_\cdot$ is *greater* than the expected value of the dwell-time. However, when $t_\cdot$ is *less* than the expected value there is no cost, which is quite natural since the system may remain in state z_k for longer than $t_\cdot$ — i.e., there should not be a large penalty for $t_\cdot$ being less than its expected value. Finally, parameters ξ_λ and μ_λ have a very natural interpretation (cf. Eq. (4.10) below): they correspond to *a priori* having ξ_λ dwell times of length μ_λ for each state.

Comparison to maximum likelihood MJP trajectories estimated using maximum likelihood (MLE) are usually trivial, with the system spending almost all its time in a single state (with the smallest λ), with infinitesimal dwell times for the other states. This poor behavior of MLE is due to the fact that the mode of $\mathsf{Exp}(\lambda)$, which is favored by the MLE, is 0, even though the mean is $1/\lambda$.[3] The SVA optimization, on the other hand, does give trajectories that are representative of the true behavior because the SVA terms of the form $\lambda t - \ln(\lambda t) - 1$ are optimized at $1/\lambda$ (i.e., at the mean of $\mathsf{Exp}(\lambda)$). We demonstrate the superior behavior of the SVA in the concrete example of estimating disease progression in patients in Section 4.5.

4.3.2 Hidden state MJP

For an HMJP, the likelihood of a valid trajectory is

$$p(\mathcal{U} \mid \mathcal{X}, \boldsymbol{\tau}, P, \boldsymbol{\lambda}) = \left(\prod_{\ell=1}^{L} p(y_\ell \mid z_{\tau_\ell}) \right) \times \left(\prod_{k=1}^{K} \lambda_{z_{k-1}} e^{-\lambda_{z_{k-1}} t_{k-1}} p_{z_{k-1} z_k} \right) e^{-\lambda_{z_K}(T - t_\cdot)}. \quad (4.7)$$

Hence, the only difference between the directly observed case and the HMJP is the addition of the observation likelihood terms. Because multinomial observations are commonly used in MJP applications, that is the case we consider here. Let N denote the number of possible observations and ρ_{sn} be the probability of observing $y_\ell = n$ when $z_{\tau_\ell} = s$. The observation likelihoods are scaled in the same manner as the transition probabilities, but with $\hat{\beta} = \zeta\beta$. Thus, for the HMJP, we

[3] Note that placing priors on the rate parameters, as we do, does not affect the degeneracy of the ML trajectory.

obtain:

$$\min_{\mathcal{U},\boldsymbol{\lambda},P,\rho} \Bigg\{ \zeta \sum_{\ell=1}^{L} \ln \rho_{z_{\tau_\ell} y_\ell} + \xi \sum_{k=0}^{K-1} \ln p_{z_k z_{k+1}} + \sum_{k=0}^{K-1} (\lambda_{z_k} t_k - \ln \lambda_{z_k} t_k - 1) + \mathbb{1}[\lambda_{z_K} t_{\cdot} \geq 1](\lambda_{z_K} t_{\cdot} - \ln \lambda_{z_K} t_{\cdot} - 1) + \xi_\lambda \sum_{z=1}^{M} (\mu_\lambda \lambda_z - \ln \lambda_z - 1) \Bigg\}. \tag{4.8}$$

4.3.3 Algorithm

Optimizing the JUMP-means objectives in Eq. (4.6) and Eq. (4.8) is non-trivial due to the fact that we do not know the number of jumps in the MJP, and the combinatorial explosion in the sequences with the number of jump points. The terms involving the continuous variables t_k (dwell times) present an additional complexity.

We therefore resort to an alternating minimization procedure to optimize the JUMP-means objective function, similar in spirit to the one used by Roychowdhury et al. (2013b). In each iteration of the optimization process, we first use a modified Viterbi algorithm to obtain the most likely state sequence. Then, we use convex optimization to distribute the jump points optimally with respect to the values from $\boldsymbol{\lambda}$ for the current state sequence.

Directly observed MJP When optimizing Eq. (4.6), there may be many sequences ($\mathcal{O}$'s) available, representing distinct realizations of the process. We use the following algorithm to optimize Eq. (4.6):

1. Initialize the state transition matrix P and rate vector $\boldsymbol{\lambda}$ with uniform values.
2. For every observation sequence $\mathcal{O}$, instantiate the jump points by adding one jump point between every pair of observations, in addition to the start and end points.
3. For each $\mathcal{O}$, use a modified Viterbi algorithm. to find the best state sequence to optimize Eq. (4.6), while keeping the jump points fixed. The modified algorithm includes the dwell time penalty terms, which are dependent upon the assignment of states to the time points.
4. Optimize the dwell times t_k with the state sequences of the trajectories fixed.
5. Optimize P and $\boldsymbol{\lambda}$ with the other variables fixed. The optimal values can be obtained in closed form. For example, if there is only a single observation sequence $\mathcal{O}$ with corresponding inferred

trajectory $\mathcal{S}$, then

$$p_{mj} = \frac{n_{mj}}{\sum_{j=1}^{M} n_{mj}}, \quad m, j \in [M] \tag{4.9}$$

$$\lambda_m = \frac{\xi_\lambda + \sum_k \mathbb{1}[z_k = m]}{\xi_\lambda \mu_\lambda + \sum_k \mathbb{1}[z_k = m] t_k}, \tag{4.10}$$

where n_{mj} denotes to the number of transitions from state m to state j in $\mathcal{S}$.

6. Repeat steps 3-5 until convergence.

Beam search variant We note that the optimization procedure just described is restrictive since the number of jump points is fixed and the jump points are constrained by the observation boundaries. To eliminate this, we also tested a beam search variant of the algorithm to allow for the creation and removal of jump points, but found it did not have much impact in our experiments.

Hidden state MJP The algorithm to optimize the hidden state MJP JUMP-means objective Eq. (4.8) is similar to that for optimizing Eq. (4.6), but with three modifications. First, in place of $\mathcal{O}$, we have the indirect observations of the states $\mathcal{X}$. Second, observation likelihood terms containing ρ are included in the objective minimized by the Viterbi optimization (step 3). Finally, an additional update is performed in step 5 for each of the observation distributions ρ_m:

$$\rho_{mn} = \frac{\sum_\ell \mathbb{1}[z_{\tau_\ell} = m] \mathbb{1}[y_\ell = n]}{\sum_\ell \mathbb{1}[z_{\tau_\ell} = m]} \tag{4.11}$$

for $m \in [M]$ and $n \in [N]$. If each $\rho_m \triangleq (\rho_{m1}, \ldots, \rho_{mN})$ is initialized to be uniform, then the algorithm converges to a poor local minimum, so we add a small amount of random noise to each uniform ρ_m.

4.4 Bayesian nonparametric MJPs

We now consider the Bayesian nonparametric MJP (iMJP) model. The iMJP is based on the hierarchical gamma-exponential process (HΓEP), which we described in Section 2.2.3.

As in the parametric case, we must replace the exponential distribution in Eq. (2.20) with the scaled exponential distribution. After an appropriate scaling of the rest of the hyperparameters, we obtain the hierarchical gamma-gamma process (HΓΓP). The properties of the HΓΓP are given in the Supplementary Material (Appendix A).

Let M denote the number of used states, K_m the number of transitions out of state m, and μ_{ij} the mass on the j-th component of the measure μ_i. For $0 \leq i \leq M, 1 \leq j \leq M$, let $\bar{\pi}_{ij} \triangleq \bar{\mu}_{ij}$ and for $0 \leq i \leq M$, let $\bar{\pi}_{i,M+1} \triangleq 1 - \sum_{j=1}^{M} \bar{\mu}_{ij}$. Let $\bar{t}^*_m \triangleq (t^*_{m1}, \ldots, t^*_{mK_m})$ be the waiting times

following state m and define $t^*_{m\cdot} \triangleq \sum_{j=1}^{K_m} t^*_{mj}$. In order to retain the effects of the hyperparameters in the asymptotics, set $\alpha_0 = \exp(-\xi_1\beta)$ and $\gamma_0 = \kappa_0 = \xi_2$.

The hierarchical gamma-gamma process (HΓΓP) is defined to be:

$$\mu_0 \sim \Gamma\mathrm{P}(\gamma_0, \alpha_0 H_0) \tag{4.12}$$

$$\mu_i \mid \mu_0 \overset{\text{i.i.d.}}{\sim} \Gamma\mathrm{P}(\gamma, \beta\mu_0) \qquad i = 1, 2, \ldots \tag{4.13}$$

$$z_k \mid \{\mu_i\}_{i=0}^{\infty}, \mathcal{U}_{k-1} \sim \bar{\mu}_{z_{k-1}} \tag{4.14}$$

$$t_k \mid \{\mu_i\}_{i=0}^{\infty}, \mathcal{U}_{k-1} \sim \mathrm{Gam}(\beta, \|\mu_{z_{k-1}}\|). \tag{4.15}$$

The measures $\{\mu_i\}_{i=0}^{\infty}$ and H_0 can be integrated out of the HΓΓP generative model in a manner analogous to the way in the the Chinese restaurant franchise in obtained from the hierarchical Dirichlet process (Teh et al., 2006c). However the mass of the measure μ_0 cannot be integrated out. We omit details as they are essentially identical to those in case of the HΓEP (Saeedi and Bouchard-Côté, 2011).

First, we consider the case of integrating out $\{\mu_i\}_{i\geq 0}$. Let M denote the number of used states, K_m the number of transitions out of state m, and r_m the number of states that can be reached from state m in one step. The contribution to the likelihood from the HΓΓP prior is

$$\begin{aligned} p(\mathcal{U}, \kappa_0 \mid \beta, \gamma_0, \gamma, \alpha_0) &= p(\kappa_0 \mid \alpha_0, \gamma_0) p(\mathcal{S} \mid \beta, \alpha_0, \kappa_0) p(\mathcal{T} \mid \beta, \gamma, \kappa_0) \\ &\propto \kappa_0^{\alpha_0 - 1} e^{-\gamma_0\kappa_0} \alpha_0^{M-1} \frac{\Gamma(\alpha_0 + 1)}{\Gamma(\alpha_0 + r_\cdot)} \prod_{m=1}^{M} (\beta\kappa_0)^{r_m - 1} \frac{\Gamma(\beta\kappa_0 + 1)}{\Gamma(\beta\kappa_0 + K_m)} \\ &\quad \times \prod_{m=1}^{M} \frac{\Gamma(\beta(\kappa_0 + K_m))}{\Gamma(\beta)^{K_m}} \frac{(\prod_{j=1}^{K_m} t^*_{mj})^{\beta - 1}}{(\gamma + \sum_{j=1}^{K_m} t^*_{mj})^{\beta(\kappa_0 + K_m)}}, \end{aligned}$$

where $r_\cdot \triangleq \sum_m r_m$. Taking the logarithm, using asymptotic expansions for the Gamma terms, and ignoring $o(\beta)$ terms yields

$$(\alpha_0 - 1)\ln\kappa_0 - \gamma_0\kappa_0 + (M-1)\ln\alpha_0 + \sum_{m=1}^{M} \{(r_m - 1)\ln\kappa_0 + \beta(\kappa_0 + K_m)\ln[\beta(\kappa_0 + K_m)]\}$$

$$\sum_{m=1}^{M} \left\{ -\beta(\kappa_0 + K_m) - K_m[\beta\ln\beta - \beta] + \beta\textstyle\sum_{j=1}^{K_m} \ln t^*_{mj} - \beta(\kappa_0 + K_m)\ln(\gamma + t^*_{m\cdot}) \right\},$$

where $t^*_{m\cdot} \triangleq \sum_{j=1}^{K_m} t^*_{mj}$. In order to retain the effects of the hyperparameters in the asymptotics, set $\alpha_0 = \exp(-\xi_1\beta)$ and $\gamma_0 = \exp(\xi_2\beta)$. Thus, $\kappa_0 \to 0$ as $\beta \to \infty$. We require that $\limsup_{\beta\to\infty} \kappa_0\gamma_0 < \infty$, so without loss of generality we can choose $\kappa_0 = \gamma_0^{-1} = \exp(-\xi_2\beta)$ to

obtain

$$-\beta\left(\xi_1(M-1)+\sum_{m=1}^{M}\left\{\xi_2(r_m-1)-\textstyle\sum_{j=1}^{K_m}\ln t^*_{mj}+K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}\right).$$

Thus, the objective function to minimize is

$$\zeta\sum_{\ell=1}^{L}\mathrm{KL}(y_\ell||\boldsymbol{\rho}_{z_{\tau_\ell}})+\xi_1 M+\sum_{m=1}^{M}\left\{\xi_2(r_m-1)-\textstyle\sum_{j=1}^{K_m}\ln t^*_{mj}-K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}. \tag{4.16}$$

Alternatively, the small variance asymptotics can be derived in the case where $\{\mu_i\}_{i\geq 0}$ is not integrated out. To do so, we first rewrite the HΓΓP generative model in an equivalent form, with H_0 integrated out:

$$\pi_0 \sim \mathrm{GEM}(\alpha_0) \tag{4.17}$$

$$\kappa_0 \sim \mathrm{Gam}(\alpha_0,\gamma_0) \tag{4.18}$$

$$\pi_i \mid \pi_0 \overset{\text{i.i.d.}}{\sim} \mathrm{DP}(\beta\kappa_0,\pi_0), \qquad i=1,2,\ldots \tag{4.19}$$

$$\kappa_i \mid \pi_0 \overset{\text{i.i.d.}}{\sim} \mathrm{Gam}(\beta,\gamma), \qquad i=1,2,\ldots \tag{4.20}$$

$$z_k \mid \{\pi_i\}_{i=1}^{\infty}, \mathcal{U}_{k-1} \sim \pi_{z_{k-1}} \tag{4.21}$$

$$t_k \mid \{\kappa_i\}_{i=1}^{\infty}, \mathcal{U}_{k-1} \sim \mathrm{Gam}(\beta,\kappa_{z_k}). \tag{4.22}$$

For $0\leq i\leq M, 1\leq j\leq M$, let $\bar{\pi}_{i,j}\triangleq\pi_{ij}$ and for $0\leq i\leq M$, let $\bar{\pi}_{i,M+1}\triangleq 1-\sum_{j=1}^{M}\pi_{ij}$. Integrating out $\{\kappa_i\}_{i\geq 1}$, the contribution to the likelihood from the HΓΓP prior is now

$$p(\mathcal{U}_K,\kappa_0,\bar{\pi}\mid\beta,\gamma_0,\gamma,\alpha_0) \tag{4.23}$$

$$=p(\kappa_0\mid\alpha_0,\gamma_0)p(\bar{\pi}_0\mid\alpha_0)p(\bar{\pi}_{1:M}\mid\beta\kappa_0\bar{\pi}_0)p(\mathcal{S}_K\mid\bar{\pi}_{1:M})p(\mathcal{T}_K\mid\beta,\gamma,\kappa_0) \tag{4.24}$$

$$\propto \kappa_0^{\alpha_0-1}e^{-\gamma_0\kappa_0}\prod_{i=1}^{M}\mathrm{Beta}\left(\frac{\bar{\pi}_{0i}}{1-\sum_{j=1}^{i-1}\pi_{0,j}}\middle|1,\alpha_0\right)\mathrm{Dir}(\bar{\pi}_i\mid\beta\kappa_0\bar{\pi}_0)\left(\prod_{k=1}^{K}\bar{\pi}_{z_{k-1},z_k}\right)p(\mathcal{T}_K\mid\beta,\gamma,\kappa_0) \tag{4.25}$$

$$\begin{aligned}\propto\ & \kappa_0^{\alpha_0-1}e^{-\gamma_0\kappa_0}\prod_{i=1}^{M}\left\{\frac{\Gamma(1+\alpha_0)}{\Gamma(\alpha_0)}\left(\frac{1-\sum_{j=1}^{i}\pi_{0,j}}{1-\sum_{j=1}^{i-1}\pi_{0,j}}\right)^{\alpha_0-1}\Gamma(\beta\kappa_0)\prod_{j=1}^{M+1}\frac{\bar{\pi}_{ij}^{\beta\kappa_0\bar{\pi}_{0j}-1}}{\Gamma(\beta\kappa_0\bar{\pi}_{0j})}\right\}\\ &\times\prod_{k=1}^{K}\bar{\pi}_{z_{k-1},z_k}^{\beta\xi}\times\prod_{m=1}^{M}\frac{\Gamma(\beta(\kappa_0+K_m))}{\Gamma(\beta)^{K_m}}\frac{(\prod_{j=1}^{K_m}t^*_{mj})^{\beta-1}}{(\gamma+\sum_{j=1}^{K_m}t^*_{mj})^{\beta(\kappa_0+K_m)}}.\end{aligned} \tag{4.26}$$

We use a slightly different limiting process, with $\gamma_0=\kappa_0=\xi_2$, a positive constant, and scale the

multinomial distributions Eq. (4.21) by $\beta\xi$. Taking the logarithm and and ignoring $o(\beta)$ terms as before yields[4]

$$\begin{aligned}
&\sum_{i=1}^{M}\left\{\ln\alpha_0+\beta\kappa_0\ln\beta\kappa_0-\beta+\sum_{j=1}^{M+1}\left\{-\beta\kappa_0\bar{\pi}_{0,j}\ln(\beta\kappa_0\bar{\pi}_{0,j})+\beta\kappa_0\bar{\pi}_{0,j}+\beta\kappa_0\bar{\pi}_{0,j}\ln\bar{\pi}_{ij}\right\}\right\}\\
&+\sum_{k=1}^{K}\beta\xi\ln\bar{\pi}_{z_{k-1},z_k}+\sum_{m=1}^{M}\left\{\textstyle\sum_{j=1}^{K_m}\beta\ln t^*_{mj}-\beta K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}\\
\sim&\sum_{i=1}^{M}\left\{-\beta\xi_1+\sum_{j=1}^{M+1}\left\{-\beta\kappa_0\bar{\pi}_{0,j}\ln(\bar{\pi}_{0,j})+\beta\kappa_0\bar{\pi}_{0,j}\ln\bar{\pi}_{ij}\right\}\right\}\\
&+\sum_{k=1}^{K}\beta\xi\ln\bar{\pi}_{z_{k-1},z_k}+\sum_{m=1}^{M}\left\{\textstyle\sum_{j=1}^{K_m}\beta\ln t^*_{mj}-\beta K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}\\
\sim&-\beta\left\{\xi_1 M+\xi\sum_{k=1}^{K}\ln\bar{\pi}_{z_{k-1},z_k}+\sum_{m=1}^{M}\left\{\xi_2\mathrm{KL}(\bar{\pi}_0||\bar{\pi}_m)-\textstyle\sum_{j=1}^{K_m}\ln t^*_{mj}-K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}\right\}.
\end{aligned}$$

Thus, the objective function to minimize is

$$\begin{aligned}
&\zeta\sum_{\ell=1}^{L}\ln\rho_{z_{\tau_\ell}y_\ell}+\xi\sum_{k=1}^{K}\ln\bar{\pi}_{z_{k-1},z_k}+\xi_1 M\\
&+\sum_{m=1}^{M}\left\{\xi_2\mathrm{KL}(\bar{\pi}_0||\bar{\pi}_m)-\textstyle\sum_{j=1}^{K_m}\ln t^*_{mj}-K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)\right\}.
\end{aligned}\tag{4.27}$$

Like its parametric counterpart, the Bayesian nonparametric cost function penalizes dwell durations very close to zero via the $\ln t^*_{mj}$ terms. In addition, there are penalties for the number of states and the state transitions. The observation likelihood term in Eq. (4.27) favors the creation of new states to minimize the JUMP-means objective, while the state penalty $\xi_1 M$ and the non-linear penalty term $K_m\ln\left([\gamma+t^*_{m\cdot}]/K_m\right)$ counteracts the formation of a long tail of states with very few data points. The γ hyperparameter introduces an additional, nonlinear cost for each additional state — if a state is occupied for $\Omega(\gamma)$ time, then the γ term for that state does not have much effect on the cost. The KL divergence terms between $\bar{\pi}_0$ and $\bar{\pi}_m$ arise from the hierarchical structure of the prior, biasing the transition probabilities $\bar{\pi}_m$ to be similar to the prior $\bar{\pi}_0$.

4.4.1 Algorithm

For the iMJP case, we have the extra variables M and $\{\bar{\pi}_m\}_{m=0}^{M}$ to optimize. In addition, the number of variables to optimize depends on the number of states in our model. The major change in

[4]This derivation is done by Jonathan Huggins a coauthor in Huggins et al. (2015).

the algorithm from the parametric case is that we must propose and then accept or reject the addition of new states. We propose the following algorithm for optimizing the iMJP:

(1) Initialize $\boldsymbol{\rho}$, $\bar{\pi}_0$ and $\bar{\pi}_1$ with uniform values and set the number of states $M = 1$.

(2) For each observation sequence, apply the Viterbi algorithm and update the times using the new objective function in Eq. (4.27), analogously to steps (3) and (4) in the parametric algorithm.

(3) Perform MAP updates for $\boldsymbol{\rho}$ (as in Eq. (4.11)) and $\bar{\pi}$:

$$\bar{\pi}_{mj} = \frac{\xi n_{mj} + \xi_2 \bar{\pi}_{0j}}{\xi \sum_{j=1}^{M} n_{mj} + \xi_2 \bar{\pi}_{0j}}, \quad m, j \in [M] \tag{4.28}$$

$$\bar{\pi}_{0j} \propto \prod_{m=1}^{M} \bar{\pi}_{mj}^{1/M}, \quad j \in [M]. \tag{4.29}$$

(4) For every state pair $m, m' \in [M]$, form a new state $M + 1$ by considering all transitions from m to m' and reassigning all observations y_ℓ that were assigned to m' to the new state. Update $\bar{\pi}$ and $\boldsymbol{\rho}$ to estimate the overall objective function for every new set of $M + 1$ states formed in this way and accept the state set that minimizes the objective. If no such set exists, do not create a new state and revert back to the old $\bar{\pi}$ and $\boldsymbol{\rho}$.

(5) Repeat steps 2-4 until convergence.

Remark. If instead of multinomial observations we have Gaussian observations, the parameter $\boldsymbol{\rho}_s$ is replaced with the mean parameter μ_s. In this case, we update the mean for each state using the data points assigned to the state, similar to the procedure for k-means clustering (see, *e.g.,* Jiang et al. 2012; Roychowdhury et al. 2013b).

4.5 Experiments

In this section we provide a quantitative analysis of the JUMP-means algorithm and compare its performance on synthetic and real datasets with standard inference methods for MJPs. For evaluation, we consider multiple sequences of discretely observed data and randomly hold out a subset of the data. We report reconstruction error for performance comparison. We apply our algorithms to 2 synthetic and 3 real datasets. To demonstrate the wider applicability of our algorithms, in addition to a user behavior dataset, we perform evaluations on a disease progression and also a vital signs monitoring dataset.

4.5.1 Parametric models

For the parametric models, we compare JUMP-means to maximum likelihood estimation of the MJP parameters learned by EM (Asger and Ledet, 2005), the MCMC method proposed by Rao and

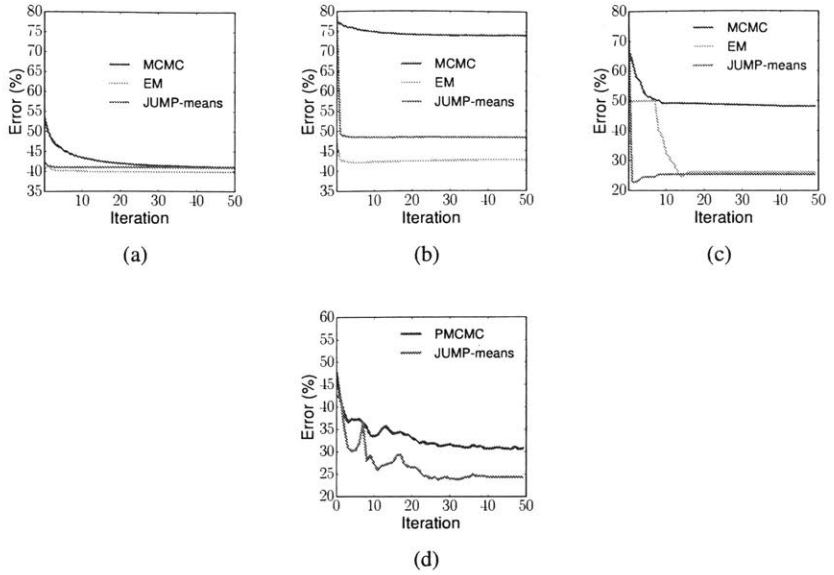


Figure 4-2: **Mean error vs iterations for 4 datasets.** **(a)** Synthetic 1; **(b)** Synthetic 2; **(c)** MS; and **(d)** MIMIC datasets. In each case the JUMP-means algorithms have better or comparable performance to other standard methods of inference in MJPs. Mean error vs CPU runtime plots can be found in the Supplementary Material.

Teh (2013) and a simple baseline where we ignore the sequential structure of the data. We run three sets of experiments (2 synthetic, 1 real) for our evaluation.

Synthetic 1: directly observed states For evaluating the model on a directly observed process, we generate 100 different datasets randomly from various MJPs with 10 states. To generate each dataset, we first generate the rows of the transition probability matrix and transition rates independently from $\mathsf{Dir}(1)$ and $\mathsf{Gam}(1,1)$, respectively. Next, given the rates and transition probabilities for each dataset, we sample 500 sequences of length 20. We hold out 30% of the observations at random for testing reconstruction error.

We run JUMP-means by initializing the algorithm with a uniform transition matrix P and set the rate vector $\boldsymbol{\lambda}$ to be all ones. We run 300 iterations of the algorithm described in Section 4.3.3; each iteration is one scan through all the sequences. We set the hyperparameters ξ, ξ_λ, and μ_λ equal to 1, 1, and .5, respectively. For MCMC, we initialize the jump points using the time points of the observations. We place independent $\mathsf{Dir}(1)$ priors on P and independent $\mathsf{Gam}(1,1)$ priors on $\boldsymbol{\lambda}$. We initialize EM with a uniform P and an all-ones $\boldsymbol{\lambda}$. We run both MCMC and EM for 300 iterations, then reconstruct observations using the Bayes estimator approximated from the 300 posterior samples. For our baseline we use the most common observation in the dataset as an estimate of the missing observations.

	Data Set			Mean Error (%)				
	# Points	Held Out	# States	BL	EM	MCMC	SVA	PMCMC
Synthetic 1 (P-DO)	10,000	30 %	10	69.7	**40.2**	**41.9**	**41.2**	-
Synthetic 2 (P-H)	10,000	30 %	5	51.8	**42.9**	74.6	46.5	-
Users	1,000	10 %	20	85.1	80.9	74.6	**62.1**	-
MS (P-DO)	390	50 %	3	51.2	**26.2**	48.1	**25.4**	-
MIMIC (NP-H)	2,208	25 %	-	42.3	25.7*	-	**24.3**	30.9

Table 4.1: **Statistics and mean observation reconstruction error for the various models on different datasets. Key:** BL = Baseline; P = parametric; SVA = JUMP-means; NP = nonparametric; DO = directly observed; H = hidden; MS = multiple sclerosis data set; MIMIC = blood pressure data set. *Best result obtained by running EM with various number of hidden states (up to 12).

Table 4.1 gives the mean reconstruction error across sequences for the various methods. Note that JUMP-means performs better than MCMC, and is almost on par with EM. Fig. 4-2(a) shows the average error across all the datasets for each method versus number of iterations. In terms of CPU time, each iteration of JUMP-means (Java), EM (Java), and MCMC (Python) takes 0.3, 1.61 and 42 seconds, respectively. We also ran experiments with the beam search variant described in Section 4.3.3; however, we did not obtain any significant improvement in results.

Synthetic 2: hidden states For the hidden state case, we generate 100 different datasets for MJPs with 5 hidden and 5 observed states, with varying parameters as above. In each dataset there are 500 sequences of length 20. In addition to parameters in the directly observed case, we generate observation likelihood terms for each state from $\mathrm{Dir}(1)$.

We initialize the transition probabilities and the rate vectors for JUMP-means, MCMC and EM in a fashion similar to the directly observed case. For the observation likelihood $\boldsymbol{\rho}$, we use $\mathrm{Dir}(1)$ as a prior for MCMC, uniform distributions for EM initialization and a uniform probability matrix with a small amount of random noise for JUMP-means initialization. We set $\xi, \xi_\lambda, \mu_\lambda$ as before and ξ to 1.

We run each algorithm for 300 iterations. For JUMP-means, we use the hidden state MJP algorithm described in Section 4.3.3. Table 4.1 and Fig. 4-2(b) again demonstrate that JUMP-means outperforms MCMC by a large margin and performs comparably to EM. The poor performance of MCMC is due to slow mixing over the parameters and state trajectories. The slow mixing is a result of the coupling between the latent states and the observations, which is induced by the observation likelihood.

Analyzing user behavior traces Identifying the tasks that a user has performed can be done via inferring the most-likely latent trajectory given the observations from that user. We study the performance of our model in a dataset collected from users of a photo editing application. After pre-processing the dataset by only keeping the top 20 used tools and truncating the sequences to 20 observations for each user, we applied the algorithm to 250 users. For evaluation, we randomly hold

out 10% of the data points and reconstruct those observations. Initialization and hyperparameters are the same as the previous experiment.

Similar to the synthetic experiments, we compare our method with EM and MCMC and report the results in Table 4.1. The results show that our method significantly outperforms both MCMC and EM. The poor performance of the EM is most likely due to the large state space.

Disease progression in Multiple Sclerosis (MS) Estimating disease progression and change points in patients with Multiple Sclerosis (MS) is an active research area (see, *e.g.,* Mandel 2010). We can cast the progression of the disease in a single patient as an MJP, with different states representing the various stages of the disease. Obtaining the most-likely trajectory for this MJP can aid in understanding the disease progression and enable better care.

For our experiments, we use a real-world dataset collected from a phase III clinical trial of a drug for MS. This dataset tracks the progression of the disease for 72 different patients over three years. We randomly hold out 50% of the observations and evaluate on the observation reconstruction task. The observations are values of a disability measure known as EDSS, recorded at different time points. Initialization and hyperparameters are the same as Synthetic 1.

Table 4.1 shows that JUMP-means significantly outperforms MCMC, achieving almost a 50% relative reduction in reconstruction error. JUMP-means again achieves comparable results with EM. Fig. 4-3 (top panel) provides an example of the latent trajectories from JUMP-means and maximum likelihood estimate for a single patient. The MLE trajectory includes two infinitesimal dwell times, which do not reflect realistic behavior of the system (since we do not expect a patient to be in a disease state for an infinitesimal amount of time). On the other hand, the trajectory produced by JUMP-means takes into account the dwell times of the various stages of the disease and provides a more reasonable picture of its progression.

4.5.2 Nonparametric model

Vital signs monitoring (MIMIC) We now consider a version of the problem of understanding physiological signals discussed in the introduction. We use data from the MIMIC database (Goldberger et al., 2000; Moody and Mark, 1996), which contains recordings of several vital parameters of ICU patients. Specifically, we consider blood pressure readings of 69 ICU patients collected over a 24-hour period and sub-sample observation sequences of length 32 for each patient, keeping the start and end times fixed.[5] For testing, we randomly hold out ~25% of the observations.

To initialize JUMP-means, we choose uniform matrices for $\rho, \bar{\pi}_0$ and $\bar{\pi}_1$ and set $M = 1$. The hyperparameters γ and ξ_1 are set to 5, while ζ, ξ, and ξ_2 are set to 0.005. Using a Gaussian likelihood model for the observations, we run our model for 50 iterations. We compare with particle

[5]We use a small dataset for testing since PMCMC cannot easily scale to larger datasets.

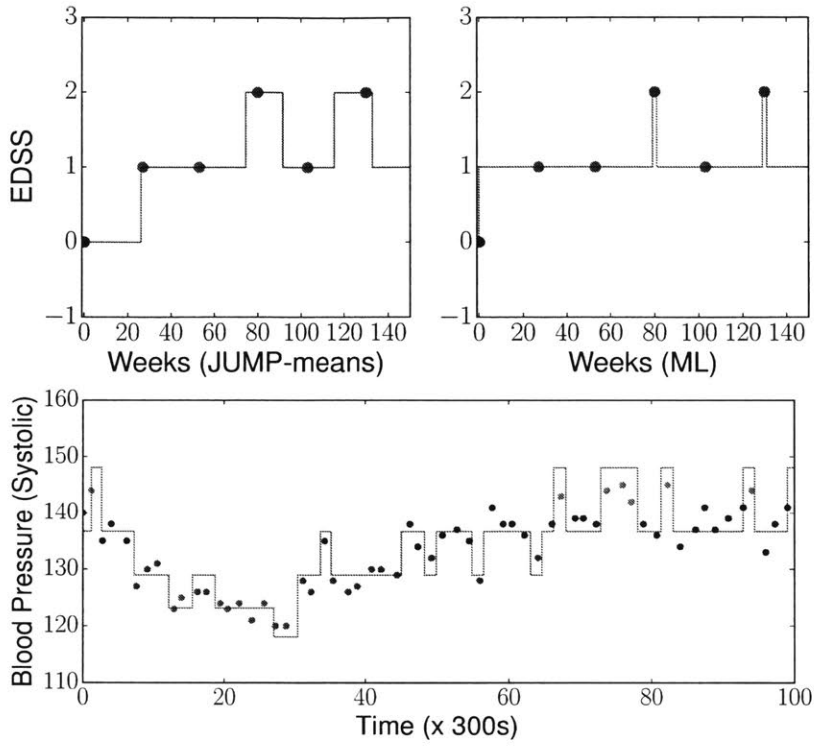


Figure 4-3: **Latent trajectories inferred by JUMP-means in MS and MIMIC datasets. Top:** Latent trajectories inferred by JUMP-means and ML estimate for a patient in the MS dataset. **Bottom:** Latent trajectory inferred by JUMP-means for a patient in MIMIC dataset.

MCMC (PMCMC) (Andrieu et al., 2010b) and EM. PMCMC is a state-of-the-art inference method for iMJPs (Saeedi and Bouchard-Côté, 2011), which we run for 300 iterations with 100 particles. For PMCMC, we first categorize the readings into the standard four categories for blood pressure provided by NIH[6]. We run EM with a number of hidden states from 1 to 12 and report the best performance among all the results. For initializing the EM, we use the same setting as the Synthetic 2 case.

For evaluation, we consider the time point of a test observation and categorize the mean of the latent state at this time point (using the same categories obtained above) to compare against the actual category. Table 4.1 shows that JUMP-means significantly outperforms PMCMC and obtains a 21% relative reduction in average error rate. Fig. 4-2(c) plots the error against iterations of both algorithms. In terms of CPU time, each iteration of JUMP-means (Java) and PMCMC (Java) takes 0.17 and 1.95 seconds, respectively. Compared to EM's error rate of 25.7%, JUMP-means reaches

[6] http://www.nhlbi.nih.gov/health/health-topics/topics/hbp

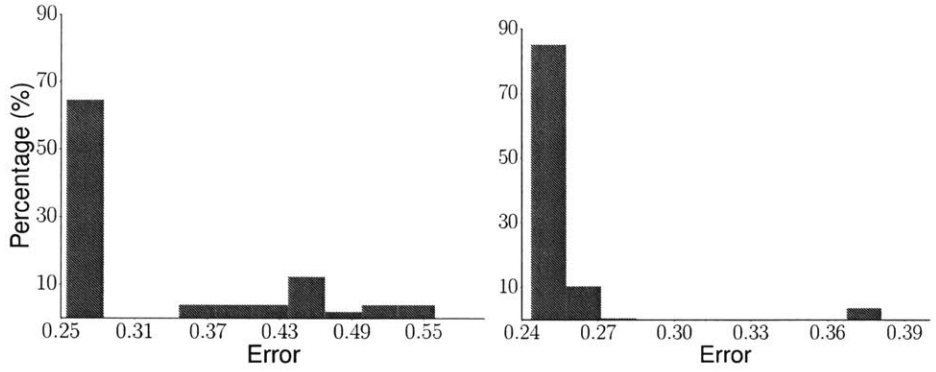


Figure 4-4: **Histograms of error reconstruction for runs with different hyperparameter settings** (a) MS (P-DO, 48 settings), and (b) MIMIC (NPB-H, 1125 settings) datasets.

a rate of 24.3% without the need to separately train for different number of states. The second-best result for the EM had an error of 45%, which shows the importance of model selection when using EM.

Fig. 4-3 (bottom) provides an example of the latent trajectory inferred by JUMP-means. The observations are uniquely colored by the latent state they are assigned. We note that the model captures different levels of blood pressure readings and provides a non-degenerate latent trajectory.

Hyperparameters A well-known problem when applying SVA methods is that there are a number of hyperparameters to tune. In our objective functions, some of these hyperparameters (γ, μ_λ, and ξ_λ) have natural interpretations so prior knowledge and common sense can be used to set them, but others do not. Fig. 4-4 shows histograms over the errors we obtain for runs of JUMP-means on the MS and MIMIC datasets with different settings. We can see that a significant fraction of the runs converge to the minimum error, while some settings — in particular when the hyperparameters were of different orders of magnitude — led to larger errors. Hence, the sensitivity study indicates the robustness of JUMP-means to the choice of hyperparameters.

Scaling Fig. 4-5 shows the total runtime and reconstruction error of the non-parametric JUMP-means algorithm on increasingly large amounts of synthetic data. The algorithm is able to handle up to a million data points with the runtime scaling linearly with data size. Furthermore, the error rate decreases significantly as the amount of data increases.

For these experiments we generated 4 datasets consisting of 10^2 to 10^5 sequences. All datasets are sampled from a single hidden state MJP with 5 hidden states and 5 possible observations. For the 20 observations in each sequence a Gaussian likelihood is used. Finally, for the held out results, we categorized the observations in 5 bins, removed 30% of the data points and predicted their category.

Time-accuracy plots for the experiments In Fig. 4-6, we compare the time-accuracy across

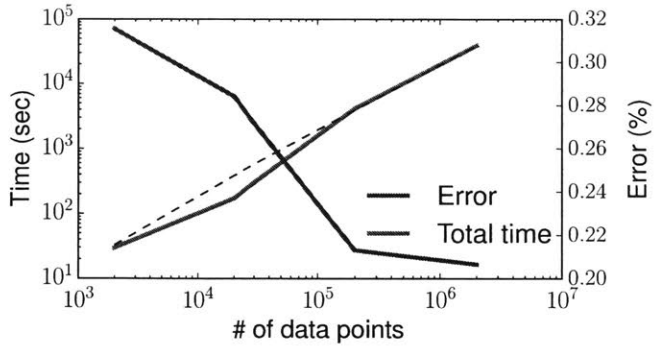


Figure 4-5: **JUMP-means scaling linearly with data size.** Runtime and error of nonparametric JUMP-means algorithm with increasing synthetic data size. The runtime scales linearly with data size (dashed black line).

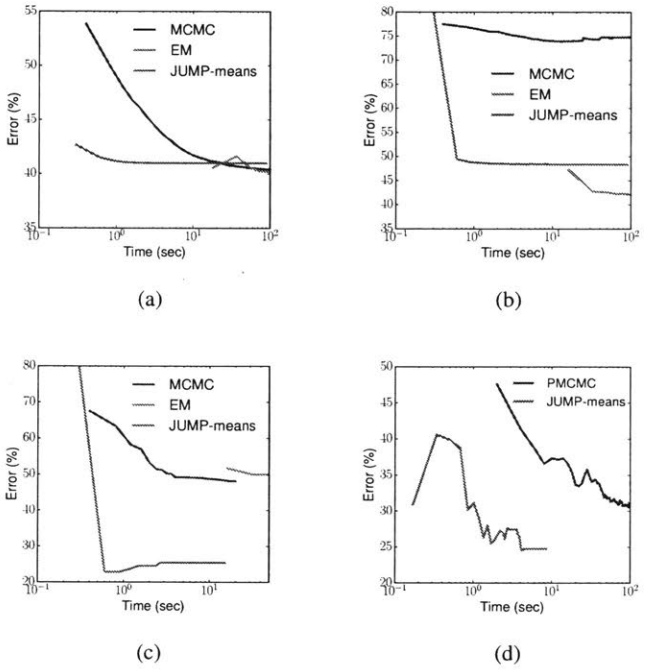


Figure 4-6: **Mean error vs CPU runtime for different datasets.** **(a)** Synthetic 1; **(b)** Synthetic 2; **(c)** MS; and **(d)** MIMIC datasets. In each case the JUMP-means algorithms have better or comparable performance to other standard methods of inference in MJPs.

different methods for different datasets. EM, PMCMC, and JUMP-means are implemented in Java and MCMC is implemented in Python. To plot the MCMC results, we give a speed boost of 100x in the results to compensate for Python's slow interpreter. From our experience with scientific comput-

ing applications, we believe this is a generous adjustment. Also we note that the EM implementation used in our experiments is not the most optimized in terms of time per iteration. However, our goal is to show that JUMP-means can achieve comparable performance with a reasonable implementation of MCMC and EM.

4.6 Conclusion

We have presented JUMP-means, a new approach to inference in MJPs using small-variance asymptotics. We derived novel objective functions for parametric and Bayesian nonparametric models and proposed efficient algorithms to optimize them. Our experiments demonstrate that JUMP-means can be used to obtain high-quality non-degenerate estimates of the latent trajectories in user trace segmentation applications and also other real datasets. JUMP-means offers attractive speed-accuracy tradeoffs for both parametric and nonparametric problems, and achieved state-of-the-art reconstruction accuracy on nonparametric problems.

CHAPTER 5

Topic modeling applied to user behavior traces with spherical HDP [1]

5.1 Introduction

In Chapters 3 and 4, we assumed the user trace data has a sequential structure and built our latent variable models based on that assumption. Assuming a dependency structure over time allowed us to learn dynamics within each task (Chapter 3) and also between the tasks (Chapter 4). In this chapter, we relax that assumption and instead assume each user trace can be represented as a bag of actions. With this setting we will not be able to learn the dynamics of the tasks but we can still infer the tasks that each user is interested in. We apply a novel variant of topic models to the user traces and infer the tasks (*i.e.,* topic) that are common in the whole dataset.

Prior work on topic modeling has mostly involved the use of categorical likelihoods (Blei et al., 2003; Blei and Lafferty, 2006; Rosen-Zvi et al., 2004). Applications of topic models in the textual domain treat words as discrete observations, ignoring the semantics of the language. Recent developments in distributional representations of words (Mikolov et al., 2013; Pennington et al., 2014) have succeeded in capturing certain semantic regularities, but have not been explored extensively in the context of topic modeling.

Distributional representations have also been applied to user data in order to learn the relationship between the user tasks, system commands and natural language descriptions. Adar et al. (2014) introduced a framework in which they learned high dimensional vector space representation of system commands by mining a large corpus of web documents about a software application. These vector representations can capture semantic similarities between commands via directional similarity metrics. For instance similar commands such as "eraser tool" and "background eraser tool" that

[1] This chapter is based on the work by Batmanghelich et al. (2016) (http://aclweb.org/anthology/P/P16/P16-2087.pdf)

may often be used together will have vector representations with higher directional similarity compared to commands like "eraser tool" and "lasso tool". In this chapter, we propose a probabilistic topic model with a novel observational distribution that integrates well with directional similarity metrics. Our novel topic model which can incorporate the semantic similarities between the actions, can also be applied to natural language text documents for which we have word vector representations such as *word2vec* (Mikolov et al., 2013). In the remaining of this chapter we use word vectors and action vectors interchangeably.

One way to employ semantic similarity is to use the Euclidean distance between word vectors, which reduces to a Gaussian observational distribution for topic modeling (Das et al., 2015). The *cosine distance* between word embeddings is another popular choice and has been shown to be a good measure of semantic relatedness (Mikolov et al., 2013; Pennington et al., 2014). The von Mises-Fisher (vMF) distribution is well-suited to model such directional data (Dhillon and Sra, 2003; Banerjee et al., 2005) but has not been previously applied to topic models.

In this chapter, we use vMF as the observational distribution. Each action can be viewed as a point on a unit sphere with topics being canonical directions. More specifically, we use a Hierarchical Dirichlet Process (HDP) (Teh et al., 2006a), a Bayesian nonparametric model we described in Section 2.2.2, to automatically infer the number of topics. We implement an efficient inference scheme based on Stochastic Variational Inference (SVI) (Hoffman et al., 2013).

We perform experiments on a user behavior trace dataset from a photo editing application and also two different English text corpora: 20 NEWSGROUPS and NIPS and compare against two baselines - HDP and Gaussian LDA. Our model, spherical HDP (sHDP), outperforms all three systems on the measure of *topic coherence*. For instance, sHDP obtains gains over Gaussian LDA of 97.5% on the NIPS dataset and 65.5% on the 20 NEWSGROUPS dataset. Qualitative inspection reveals consistent topics produced by sHDP. We also empirically demonstrate that employing SVI leads to efficient topic inference.

5.2 Related Work

Topic modeling and word embeddings Das et al. (2015) proposed a topic model which uses a Gaussian distribution over word embeddings. By performing inference over the vector representations of the words, their model is encouraged to group words that are semantically similar, leading to more coherent topics. In contrast, we propose to utilize von Mises-Fisher (vMF) distributions which rely on the cosine similarity between the word vectors instead of euclidean distance.

vMF in topic models The vMF distribution has been used to model directional data by placing points on a unit sphere (Dhillon and Sra, 2003). Reisinger et al. (2010) propose an admixture model

that uses vMF to model documents represented as vector of normalized word frequencies. This does not account for word level semantic similarities. Unlike their method, we use vMF over word embeddings. In addition, our model is nonparametric.

Nonparametric topic models HDP and its variants have been successfully applied to topic modeling (Paisley et al., 2015; Blei, 2012; He et al., 2013); however, all these models assume a categorical likelihood in which the words are encoded as one-hot representation.

Vector representations in user modeling Adar et al. (2014) introduced a framework for learning vector representations of system commands in software applications using a corpus of online documents. Their framework is capable of learning the semantic regularities between commands; furthermore, it can learn the relationship between the commands, tasks and natural language descriptions. For instance, it can learn a mapping between a natural language query and its most likely corresponding system command. In this chapter, we use their pre-trained vector representations learned over a large corpus of online documents about Adobe photoshop. Yang et al. (2017) proposed a similar model but instead of mapping commands to vectors, they proposed vector representations for the user behavior traces directly. This enables them to make personalized recommendations based on a user behavior trace. Our model is related to theirs as it also proposes a representation for each user behavior trace; however, instead of directly learning a vector from a trace of actions, we first map the actions into vectors using the pre-trained vectors and then apply a topic model to the transformed dataset.

5.3 Model

In this section, we describe the generative process for user traces. Rather than one-hot representation of actions, we employ normalized action embeddings (Mikolov et al., 2013; Adar et al., 2014) to capture semantic meanings of associated actions. Action n from user d is represented by a normalized M-dimensional vector y_{dn} and the similarity between actions is quantified by the cosine of angle between the corresponding action vectors. Note that our model can also be applied to natural language text datasets where y_{dn} denotes word n from document d.

5.3.1 Generative model

Our model is based on the Hierarchical Dirichlet Process (HDP) (see Chapter 2 for a review of the HDP). The model assumes a collection of "topics" that are shared across documents in the corpus. The topics are represented by the topic centers $\mu_k \in \mathbb{R}^M$. Since action vectors are normalized, the μ_k can be viewed as a direction on unit sphere.

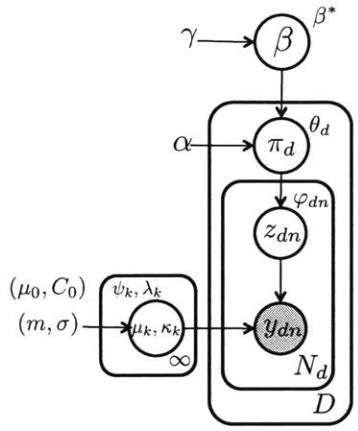


Figure 5.3.1: **Graphical representation of our spherical HDP (sHDP) model**. The symbol next to each random variable denotes the parameter of its variational distribution. We assume D users in the dataset, each user has N_d actions and there are countably infinite topics represented by (μ_k, κ_k).

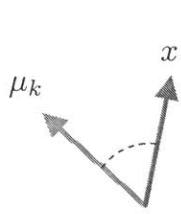
Von Mises–Fisher (vMF) distribution Von Mises–Fisher (vMF) is a distribution that is commonly used to model directional data. The likelihood of the topic k for action y_{dn} is:

$$f(y_{dn}; \mu_k, \kappa_k) = \exp\left(\kappa_k \mu_k^T y_{dn}\right) C_M(\kappa_k)$$

where κ_k is the concentration of the topic k, the $C_M(\kappa_k) := \kappa_k^{M/2-1} / \left((2\pi)^{M/2} I_{M/2-1}(\kappa_k)\right)$ is the normalization constant, and $I_\nu(\cdot)$ is the modified Bessel function of the first kind at order ν. Interestingly, the log-likelihood of the vMF is proportional to $\mu_k^T y_{dn}$ (up to a constant), which is equal to the cosine distance between two vectors. This distance metric is also used by Mikolov et al. (2013) to measure semantic proximity.

For inference we can use a conjugate prior for the mean parameter μ_k; however, the concentration parameter does not have a conjugate prior and we need to resort to approximate inference. The conjugate prior for a vMF distribution with fixed concentration parameter is a vMF distribution. We denote the parameters of this prior distribution by μ_0 and κ_0. Given N data points $y_{1:N}$, we have the following posterior distribution for the mean parameter:

$$\begin{aligned} p(\mu | y_{1:N}; \kappa, \mu_0, \kappa_0) &\propto \text{vMF}(\mu; \mu_0, \kappa_0) \prod_{i=1}^{N} \text{vMF}(y_i; \mu, \kappa) \\ &= C(\kappa_0) C(\kappa)^N \exp\left(\mu^T \left(\kappa_0 \mu_0 + \kappa \sum_{n=1}^{N} y_i\right)\right). \end{aligned}$$

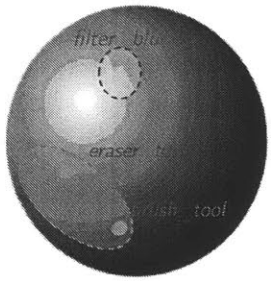


$$p(x|\mu_k) \propto \exp(x^T \mu_k)$$

$$p(x|\mu_k) = C_M(1) \exp(x^T \mu_k)$$

Normalization Constant

(a)



$$p(x|\mu_k, \kappa) = C_M(\kappa) \exp(\kappa\, x^T \mu_k)$$

(b)

Figure 5.3.2: **Von Mises–Fisher (vMF) distribution (a)** Given the mean vector representation of the topic, μ_k, the likelihood of a vector x in the vMF distribution is proportional to the exponential of the similarity to the center of the topic. We need to normalize this value to get a valid probability density value. **(b)** vMF distribution has an extra parameter, κ, which controls the concentration of the topic The bigger κ results in more focused topic. In the above figure the green ellipse illustrates a smaller concentrations parameter.

Hence, the posterior is a vMF distribution with mean parameter $\nu_N / \|\nu_N\|_2$ and concentration parameter $\|\nu_N\|_2$ where $\nu_N = \kappa_0 \mu_0 + \kappa \sum_{n=1}^{N} y_i$. This property of the vMF distribution helps us in developing a variational scheme for inference. For the concentration parameter, we utilize an importance sampling approach (details below).

Hierarchical Dirichlet process When sampling a new document, a subset of topics determine the distribution over words. We let z_{dn} denote the topic selected for the word n of document d. Hence, z_{dn} is drawn from a categorical distribution: $z_{dn} \sim \text{Mult}(\pi_d)$, where π_d is the proportion of topics for document d. We draw π_d from a Dirichlet Process which enables us to estimate the the number of topics from the data. The generative process for the generation of new document is as follows:

$$\beta \sim \text{GEM}(\gamma) \qquad \pi_d \sim \text{DP}(\alpha, \beta)$$

$$\kappa_k \sim \text{log-Normal}(m, \sigma^2) \qquad \mu_k \sim \text{vMF}(\mu_0, C_0)$$

$$z_{dn} \sim \text{Mult}(\pi_d) \qquad y_{dn} \sim \text{vMF}(\mu_k, \kappa_k)$$

where $\text{GEM}(\gamma)$ is the stick-breaking distribution with concentration parameter γ, $\text{DP}(\alpha, \beta)$ is a Dirichlet process with concentration parameter α and stick proportions β (Teh et al., 2012). We use log-normal and vMF as hyper-prior distributions for the concentrations (κ_k) and centers of the

topics (μ_k) respectively. Fig. 5.3.1 provides a graphical illustration of the model.

5.3.2 Stochastic variational inference

In the rest of the chapter, we use bold symbols to denote the variables of the same kind (*e.g.*, $\mathbf{y}_d = \{y_{dn}\}_n$, $\mathbf{z} := \{z_{dn}\}_{d,n}$). We employ stochastic variational mean-field inference (SVI) (Hoffman et al., 2013) to estimate the posterior distributions of the latent variables. SVI enables us to sequentially process batches of documents which makes it appropriate in large-scale settings.

To approximate the posterior distribution of the latent variables, the mean-field approach finds the optimal parameters of the fully factorizable q (*i.e.*, $q(\mathbf{z}, \beta, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\kappa}) := q(\mathbf{z})q(\beta)q(\boldsymbol{\pi})q(\boldsymbol{\mu})q(\boldsymbol{\kappa})$) by maximizing the Evidence Lower Bound (ELBO),

$$\mathcal{L}(q) = \mathbb{E}_q \left[\log p(\boldsymbol{Y}, \mathbf{z}, \beta, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\kappa})\right] - \mathbb{E}_q \left[\log q\right]$$

where $\mathbb{E}_q[\cdot]$ is expectation with respect to q, $p(\boldsymbol{Y}, \mathbf{z}, \beta, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\kappa})$ is the joint likelihood of the model specified by the HDP model. For the variational distribution in the HDP model, we use the direct assignment truncation scheme we described in Section 2.2.4.

The variational distributions for $z, \boldsymbol{\pi}, \boldsymbol{\mu}$ have the following parametric forms,

$$\begin{aligned} q(\mathbf{z}) &= \text{Mult}(\mathbf{z}|\boldsymbol{\varphi}) \\ q(\boldsymbol{\pi}) &= \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\theta}) \\ q(\boldsymbol{\mu}) &= \text{vMF}(\boldsymbol{\mu}|\boldsymbol{\psi}, \boldsymbol{\lambda}), \end{aligned}$$

where Dir denotes the Dirichlet distribution and $\boldsymbol{\varphi}, \boldsymbol{\theta}, \boldsymbol{\psi}$ and $\boldsymbol{\lambda}$ are the parameters we need to optimize the ELBO.

Update equations for the parameters of the vMF variational distribution ($\boldsymbol{\psi}, \boldsymbol{\lambda}$) For the parameters of the $q(\boldsymbol{\mu})$ since the vMF distributions are conjugate with respect to each other, we have closed form update equations. That is, for a batch size of one (*i.e.*, processing one user/document at time) we have the following update equations for $\boldsymbol{\psi}$ and $\boldsymbol{\lambda}$:

$$\begin{aligned} t &\leftarrow (1-\rho)t + \rho s(\boldsymbol{y}_d, \boldsymbol{\varphi}_{dk}) \\ \psi &\leftarrow t/\|t\|_2, \quad \lambda \leftarrow \|t\|_2, \end{aligned}$$

where, ρ is the step size, t is the natural parameter for vMF and $s(\boldsymbol{y}_d, \boldsymbol{\varphi}_{dk})$ is a function computing the expected sufficient statistics of vMF distribution of the topic k. For $s(\boldsymbol{y}_d, \boldsymbol{\varphi}_{dk})$ we have:

$$s(\boldsymbol{y}_d, \boldsymbol{\varphi}_{dk}) = C_0\mu_0 + D\mathbb{E}_q[\kappa_k] \sum_{w=1}^{W} \mathbb{E}_q[z_{dwk}]y_{dw}$$

The prior distribution κ does not follow a conjugate distribution; hence, its posterior does not have a closed-form. Since κ is only one dimensional variable, we use importance sampling to approximate its posterior. Following Gopal and Yang (2014), we use the Jensen inequality and the expectation of posterior parameters to derive the approximate conditional distribution of κ_k:

$$\begin{aligned}
p(\kappa_k|\boldsymbol{y}_d, m, \sigma^2, \mu_0, C_0, \alpha, \beta, \gamma) &\propto p(\kappa_k, \boldsymbol{y}_d|m, \sigma^2, \mu_0, C_0, \alpha, \gamma) \\
&\approx \mathbb{E}_q[p(\kappa_k, \boldsymbol{y}_d, \boldsymbol{z}_d, \beta, \boldsymbol{\pi}_d, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \alpha, \gamma)] \\
&\geq \exp\left(\mathbb{E}_q[\log p(\kappa_k, \boldsymbol{y}_d, \boldsymbol{z}_d, \beta, \boldsymbol{\pi}_d, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \alpha, \gamma)]\right) \\
&\propto \exp\left(\sum_{w=1}^{W} \mathbb{E}_q[z_{dwk}] \log C_M(\kappa_k) + \kappa_k \sum_{w=1}^{W} \mathbb{E}_q[z_{dwk}] y_{dw}^T \mathbb{E}_q[\mu_k]\right) \\
&\quad \times \text{log-Normal}(\kappa_k|m, \sigma^2).
\end{aligned}$$

We grid the one dimensional space of κ and compute the importance weights of different κ values. We use these importance weights for approximating the expected κ and consequently updating the parameters ψ and λ.

Update equations for the parameters of the HDP variational distribution (β, φ, θ) Similar to Bryant and Sudderth (2012), we view β as a parameter; hence, $q(\beta) = \delta_{\beta^*}(\beta)$. We use numerical gradient ascent to optimize for β^*. See Section 2.2.4 for details of the direct assignment truncation scheme that we are using here.

For a batch size of one, the update equations for the parameters are:

$$\begin{aligned}
\varphi_{dwk} &\propto \exp\{\mathbb{E}_q[\log \text{vMF}(y_{dw}|\psi_k, \lambda_k)] + \mathbb{E}_q[\log \pi_{dk}]\} \\
\theta_{dk} &\leftarrow (1-\rho)\theta_{dk} + \rho(\alpha\beta_k + D\sum_{w=1}^{W} \omega_{wj}\varphi_{dwk})
\end{aligned}$$

where D, ω_{wj}, W are the total number of users, number of actions w in user trace j, the total number of actions in the dictionary (of actions), respectively. We can obtain $\mathbb{E}_q[\log \text{vMF}(y_{dw}|\psi_k, \lambda_k)]$ and

		Gaussian LDA		
select_refine_edge	brush_tool	window_paths	select_transform_selection	image_reveal_all
paint_bucket_tool	nudge	view_snap_to_document	pencil_tool	layer_arrange_send_to_back
edit_transform_again	eraser_tool	window_paragraph_styles	edit_preferences_file	view_show_smart_guides
layer_layer_style_create	layer_layer_style_copy_layer	layer_merge_visible	rectangle_tool	view_lock_guides
crop_tool	file_open	brush_tool	elliptical_marquee_tool	nudge
0.203	0.177	0.232	0.190	0.208
		Spherical HDP		
brush_tool	file_export_data_sets_as_files	window_extensions_connections	select_deselect	crop_tool
eraser_tool	window_paragraph_styles	window_workspace_default_workspace	select_refine_edge	healing_brush_tool
lasso_tool	edit_preferences_file_handling	window_workspace_save_workspace	select_inverse	clone_stamp_tool
magic_wand_tool	file_scripts_script_events_manager	file_revert	edit_paste	magnetic_lasso_tool
dodge_tool	layer_hide_layers	edit_purge_histories	select_load_selection	patch_tool
0.257	0.242	0.250	0.268	0.245

Table 5.1: **Examples of top words for the most coherent topics on the user behavior trace dataset.** Examples are generated by Gaussian LDA (k=20) and Spherical HDP. The last row for each model is the topic coherence computed using the method introduced by Fang et al. (2016).

$\mathbb{E}_q[\log \pi_{dk}]$ from the following equations:

$$\mathbb{E}_q[\log \text{vMF}(y_{dw}|\psi_k, \lambda_k)] = \mathbb{E}_q[\log C_M(\kappa_k)] + \mathbb{E}_q[\kappa_k] y_{dw}^T \mathbb{E}_q[\mu_k],$$
$$\mathbb{E}_q[\log \pi_{dk}]\} = \Psi(\theta_{dk}) - \Psi(\sum_{k'} \theta_{dk'}),$$

where Ψ is the first derivative of the log Gamma function.

5.4 Experiments

Setup We perform experiments on three different datasets. One user behavior trace dataset and 2 text corpora. The user behavior trace dataset consists of 7914 users and 275510 total actions. We use the 4000 dimensional action embeddings from Adar et al. (2014) and reduce the dimension of the vectors to 50 using PCA. We post-process the vectors to have unit ℓ^2-norm.

The 2 text corpora are 11266 documents from 20 NEWSGROUPS[2] and 1566 documents from the NIPS corpus[3]. We utilize 50-dimensional word embeddings trained on text from Wikipedia using *word2vec*[4]. Similar to action vectors, we post-process the vectors to have unit ℓ^2-norm.

We evaluate our model using the measure of topic coherence (Newman et al., 2010), which has been shown to effectively correlate with human judgement (Lau et al., 2014). For the text corpora, we compute the Pointwise Mutual Information (PMI) using a reference corpus of 300k documents from Wikipedia. The PMI is calculated using co-occurence statistics over pairs of words (u_i, u_j) in 20-word sliding windows:

$$\text{PMI}(u_i, u_j) = \log \frac{p(u_i, u_j)}{p(u_i) \cdot p(u_j)}$$

[2] http://qwone.com/~jason/20Newsgroups/

[3] http://www.cs.nyu.edu/~roweis/data.html

[4] https://code.google.com/p/word2vec/

Gaussian LDA							
vector	shows	network	hidden	performance	net	figure	size
image	feature	learning	term	work	references	shown	average
gaussian	show	model	rule	press	introduction	neurons	present
equation	motion	neural	word	tion	statistical	point	family
generalization	action	input	means	ing	related	large	versus
images	spike	data	words	eq	comparison	neuron	spread
gradient	series	function	approximate	performed	source	small	median
theory	final	time	derived	em	statistics	fig	physiology
dimensional	robot	set	describe	vol	free	cells	children
1.16	0.4	0.35	0.29	0.25	0.25	0.21	0.2
Spherical HDP							
neural	function	analysis	press	pattern	problem	noise	algorithm
layer	linear	theory	cambridge	fig	process	gradient	error
neurons	functions	computational	journal	temporal	method	propagation	parameters
neuron	vector	statistical	vol	shape	optimal	signals	computation
activation	random	field	eds	smooth	solution	frequency	algorithms
brain	probability	simulations	trans	surface	complexity	feedback	compute
cells	parameter	simulation	springer	horizontal	estimation	electrical	binary
cell	dimensional	nonlinear	volume	vertical	prediction	filter	mapping
synaptic	equation	dynamics	review	posterior	solve	detection	optimization
1.87	1.73	1.51	1.44	1.41	1.19	1.12	1.03

Table 5.2: **Examples of top words for the most coherent topics on the NIPS dataset.** Examples are generated by Gaussian LDA (k=40) and Spherical HDP. The last row for each model is the topic coherence (PMI) computed using Wikipedia documents as reference.

For the user behavior dataset, we use the topic coherence calculation method proposed by Fang et al. (2016). Their method does not require a large corpus and can provide the topic coherence using the action/word vectors directly. We compare our model with two baselines: HDP and the Gaussian LDA model . We ran G-LDA with various number of topics (k).

Results Table 5.3 details the topic coherence averaged over all topics produced by each model. We observe that our sHDP model outperforms G-LDA by 0.1 points on the user behavior dataset, 0.08 points on 20 NEWSGROUPS, and by 0.17 points on the NIPS dataset. We can also see that the individual topics inferred by sHDP make sense qualitatively and have higher coherence scores than G-LDA (Tables 5.1 and 5.2). This supports our hypothesis that using the vMF likelihood helps in producing more coherent topics. sHDP produces 16 topics for the 20 NEWSGROUPS and 92 topics on the NIPS dataset.

Fig. 5.4.3 shows a plot of normalized log-likelihood against the runtime of sHDP and G-LDA.[5] We calculate the normalized value of log-likelihood by subtracting the minimum value from it and dividing it by the difference of maximum and minimum values. We can see that sHDP converges faster than G-LDA, requiring only around five iterations while G-LDA takes longer to converge.

[5]Our sHDP implementation is in Python and the G-LDA code is in Java.

Model	Topic Coherence		
	USERS	20 NEWS	NIPS
HDP	0.201	0.037	0.270
G-LDA (k=20)	0.203	-0.017	0.215
G-LDA (k=40)	0.195	0.052	0.248
G-LDA (k=60)	0.178	0.082	0.137
G-LDA (k=100)	0.217	-0.032	0.267
sHDP	**0.300**	**0.162**	**0.442**

Table 5.3: **Average topic coherence for various baselines** (HDP, Gaussian LDA (G-LDA)) and sHDP. k=number of topics. Best scores are shown in bold.

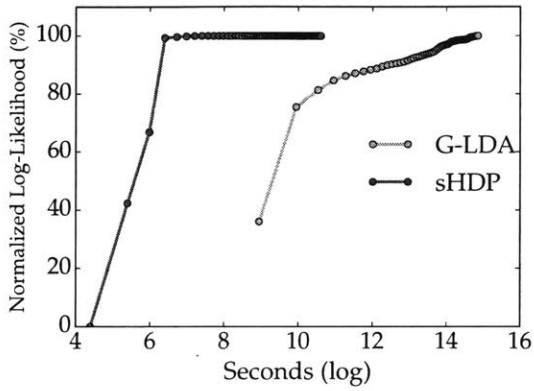


Figure 5.4.3: **Normalized log-likelihood (in percentage) over a training set of size 1566 documents from the NIPS corpus.** Since the log-likelihood values are not comparable for the Gaussian LDA and the sHDP, we normalize them to demonstrate the convergence speed of the two inference schemes for these models.

5.5 Conclusion

Classical topic models do not account for semantic regularities in language. Recently, distributional representations of words have emerged that exhibit semantic consistency over directional metrics like cosine similarity. Neither categorical nor Gaussian observational distributions used in existing topic models are appropriate to leverage such correlations. In this chapter, we demonstrate the use of the von Mises-Fisher distribution to model actions/words as points over a unit sphere. We use HDP as the base topic model and propose an efficient algorithm based on Stochastic Variational Inference. Our model naturally exploits the semantic structures of action embeddings and word embeddings while flexibly inferring the number of topics. We show that our method outperforms two competitive approaches in terms of topic coherence on three different datasets. One limitation of sHDP is that the inferred topics will be unimodal due to the unimodality of the vMF distribution.

Having a more flexible observation model will help in solving this limitation; one candidate of such flexible model can be a deep generative model with an HDP latent structure.

CHAPTER 6

Multimodal prediction and personalization in software applications with deep generative models[1]

6.1 Introduction

In the last three chapters our goal was to develop models to deal with log files and user behavior traces. We proposed models and algorithms that can infer tasks performed by the users from their log files. In the current chapter, instead of inferring the tasks, we focus on prediction and personalization.

Many office workers spend most of their working days using pro-oriented software applications. These applications are often powerful, but complicated. This complexity may overwhelm and confuse novice users, and even expert users may find some tasks time-consuming and repetitive. We want to use machine learning and statistical modeling to help users manage this complexity.

Fortunately, modern software applications collect large amounts of data from users with the aim of providing them with better guidance and more personalized experiences. A photo-editing application, for example, could use data about how users edit images to learn what kinds of adjustments are appropriate for what images, and could learn to tailor its suggestions to the aesthetic preferences of individual users. Such suggestions can help both experts and novices: experts can use them as a starting point, speeding up tedious parts of the editing process, and novices can quickly get results they could not have otherwise achieved.

Several models have been proposed for predicting and personalizing user interaction in different software applications. These existing models are limited in that they only propose a single prediction or are not readily personalized. Multimodal predictions[2] are important in cases where, given an

[1]This chapter is based on the work by Saeedi et al. (2017a) (https://arxiv.org/pdf/1704.04997.pdf).

[2]We mean "multimodal" in the statistical sense (i.e., coming from a distribution with multiple maxima), rather than in the human-computer-interaction sense (*i.e.,* having multiple modes of input or output). This term has also been used in

input from the user, there could be multiple possible suggestions from the application. For instance, in photo editing/enhancement, a user might want to apply different kinds of edits to the same photo depending on the effect he or she wants to achieve. A model should therefore be able to recommend multiple enhancements that cover a diverse range of styles.

In this chapter, we introduce a framework for multimodal prediction and personalization in software applications. We focus on photo-enhancement applications, though our framework is also applicable to other domains where multimodal prediction and personalization is valuable. Fig. 6.2.1 demonstrates our high-level goals: we want to learn to propose diverse, high-quality edits, and we want to be able to personalize those proposals based on users' historical behavior.

Our modeling and inference approach is based on the variational autoencoder (VAE) (Kingma and Welling, 2013). We propose an extension of the VAE which uses a hierarchical structure to learn styles across many diverse users. We further extend our model to provide personalized results, learning each user's personal style from their historical behavior. We introduce a novel encoder architecture that, for each user, analyzes each edit independently, and then combines the results in a symmetric, exchangeable way that extends to any number of user edits.

We apply our framework to three different datasets (collected from novice, semi-expert, and expert users) of image features and user edits from a photo-enhancement application and compare its performance qualitatively and quantitatively to various baselines. We demonstrate that our model outperforms other approaches.

6.2 Related work

In this section we provide an overview of the available models for predicting photo edits and summarize their pros and cons.

6.2.1 Related work on the prediction of photo edits

There are two main categories of models, parametric and nonparametric, that have been used for prediction of photo edits:

Parametric methods These methods approximate a parametric function by minimizing a squared (or a similar) loss. The loss is typically squared L_2 distance in Lab color space, which more closely approximates human perception than RGB space (Sharma and Bala, 2002). This loss is reasonable if the goal is to learn from a set of consistent, relatively conservative edits. But when applied to a dataset of more diverse edits, a model that minimizes squared error will tend to predict the *average*

other related work on multimodal predictions (*e.g.*, Bishop 1994; Tang and Salakhutdinov 2013; Dauphin and Grangier 2015).

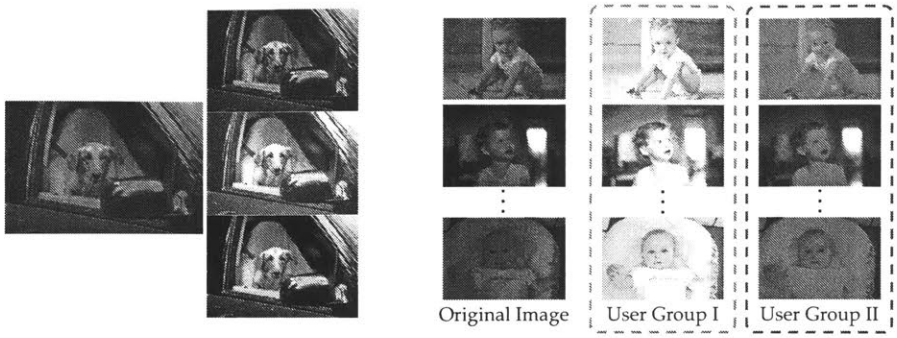


Figure 6.2.1: **The main goals of our proposed models.** (a) **Multimodal photo edits**: For a given photo, there may be multiple valid aesthetic choices that are quite different from one another. (b) **User categorization**: A synthetic example where different user clusters tend to prefer different slider values. Group 1 users prefer to increase the exposure and temperature for the baby images; group 2 users reduce clarity and saturation for similar images.

edit. At best, this will lead to conservative predictions; in the worst case, the average of several good edits may produce a bad result.

Bychkovsky et al. (2011) collect a dataset of 5000 photos enhanced by 5 different experts; they identify a set of features and learn to predict the user adjustments after training on the collected dataset. They apply a number of regression techniques such as LASSO and show their proposed adjustments can match the adjustments of one of the 5 experts. Their method only proposes a single adjustment and the personalization scheme that they suggest requires the user to edit a set of selected training photos. Yan et al. (2016) use a deep neural network to learn a mapping from an input photo to an enhanced one following a particular style; their results show that the proposed model is able to capture the nonlinear and complex nature of this mapping. More recently, Gharbi et al. (2017) propose a network architecture that is faster in processing every image compared to the model by Yan et al. (2016). Both of these models only propose a single style of adjustment.

Nonparametric methods The few available nonparametric methods are typically able to propose multiple edits or some uncertainty over the range of adjustments. Lee et al. (2015) propose a method that can generate a diverse set of edits for an input photograph. The authors have a curated set of exemplar images in various styles. They use an example-based style-transfer algorithm to transfer the style from an exemplar image to an input photograph. To choose the right exemplar image, they do a semantic similarity search using features that they have learned via a convolutional neural network (CNN). Although their approach can recommend multiple edits to a photo, their edits are destructive; that is, the recommended edits are directly applied to the photo and the user is

not able to further customize those edits. Koyama et al. (2016) introduce a model for personalizing photo edits only based on the history of edits by a single user. The authors use a self-reinforcement procedure in which after every edit by a user they: 1) update the distance metric between the user's past photos, 2) update a feature vector representation of the user's photos, and 3) update an enhancement preference model based on the feature vectors and the user's enhancement history. This model requires data collection from a single user and does not benefit from other users' information.

6.2.2 Related multimodal prediction models

Traditional neural networks using mean squared error (MSE) loss cannot naturally handle multimodal prediction problems, since MSE is minimized by predicting the average response. Neal (1992) introduces stochastic latent variables to the network and proposes training Sigmoid Belief Networks (SBN) with only binary stochastic variables. However, this model is difficult to train, and it can only make piecewise-constant predictions and is therefore not a natural fit to continuous-response prediction problems.

Bishop (1994) proposes mixture density networks (MDN), which are more suitable for continuous data. Instead of using stochastic units, the model directly outputs the parameters of a Gaussian mixture model. The complexity of MDNs' predictive distributions is limited by the number of mixture components. If the optimal predictive distribution cannot be well approximated by a relatively small number of Gaussians, then an MDN may not be an ideal choice.

Tang and Salakhutdinov (2013) add deterministic hidden variables to SBNs in order to model continuous distributions. The authors showed improvements over the SBN; nevertheless, training the stochastic units remained a challenge due to the difficulty of doing approximate inference on a large number of discrete variables. Dauphin and Grangier (2015) propose a new class of stochastic networks called linearizing belief networks (LBN). LBN combines deterministic units with stochastic binary units multiplicatively. The model uses deterministic linear units which act as multiplicative skip connections and allow the gradient to flow without diffusion. The empirical results show that this model can outperform standard SBNs.

6.3 Models

Given the limitations of the available methods for predicting photo edits (described in Section 6.2.1), our goal is to propose a framework in which we can: 1) recommend a set of diverse, parametric edits based on a labeled dataset of photos and their enhancements, 2) categorize the users based on their style and type of edits they apply, and finally 3) personalize the enhancements based on the user category. We focus on the photo-editing application in this chapter, but the proposed framework is applicable to other domains where users must make a selection from a large, richly parameterized

design space where there is no single right answer (for example, many audio processing algorithms have large numbers of user-tunable parameters).

Our framework is based on VAEs, we described in Section 2.1.3, and follows a mixture-of-experts design (Murphy, 2012, Section 11.2.4). We first introduce a conditional VAE that can generate diverse set of enhancements to a given photo. Next, we extend the model to categorize the users based on their adjustment style. Our model can provide interpretable clusters of users with similar style. Furthermore, the model can provide personalized suggestions by first estimating a user's category and then suggesting likely enhancements conditioned on that category. We slightly modify the notation from Section 2.1.3: we denote N i.i.d. observations by $y = \{y_n\}_{n=1}^N$ and the latent variables by z_n.

6.3.1 Multimodal prediction with conditional Gaussian mixture VAE (CGM-VAE)

Given a photo, we are interested in predicting a set of edits. Each photo is represented by a feature vector x_n and its corresponding edits y_n are represented by a vector of slider values (*e.g.*, contrast, exposure, saturation, etc.). We assume that there are L clusters of possible edits for each image. To generate the sliders y_n for a given image x_n, we first sample a cluster assignment s_n and a set of latent features z_n from its corresponding mixture component $\mathcal{N}(\mu_{s_n}, \Sigma_{s_n})$. Next, conditioned on the image and z_n, we sample the slider values. The overall generative process for the slider values $\{y_n\}_{n=1}^N$ conditioned on the input images $\{x_n\}_{n=1}^N$ is

$$\begin{aligned} s_n|\pi &\overset{\text{iid}}{\sim} \pi, \qquad z_n|s_n,\{\mu_\ell,\Sigma_\ell\}_{\ell=1}^L \sim \mathcal{N}(\mu_{s_n},\Sigma_{s_n}), \\ y_n|x_n,z_n,\theta &\sim \mathcal{N}(\mu(z_n,x_n;\theta),\Sigma(z_n,x_n;\theta)), \end{aligned} \tag{6.1}$$

where $\mu(z_n, x_n; \theta)$ and $\Sigma(z_n, x_n; \theta)$ are flexible parametric functions, such as MLPs, of the input image features x_n concatenated with the latent features z_n. Summing over all possible values for the latent variables s_n and z_n, the marginal likelihood $p(y_n|x_n) = \sum_{s_n} \int_{z_n} p(y_n, s_n, z_n|x_n) dz_n$ yields complex, multimodal densities for the image edits y_n.

The posterior $p(s, z|x, y)$ is intractable. We approximate it with variational recognition models as

$$p_\theta(s,z|x,y) \approx q_{\phi_s}(s|x,y) q_{\phi_z}(z|x,y,s). \tag{6.2}$$

Note that this variational distribution does not model s and z as independent. For $q_{\phi_s}(s|x, y)$, we use an MLP with a final softmax layer, and for $q_{\phi_z}(z|x, y, s)$, we use a Gaussian whose mean and covariance are the output of an MLP that takes s, x, and y as input. Fig. 6.3.2 (parts a and b) outlines the graphical model structures of the CGM-VAE and its variational distributions q_ϕ.

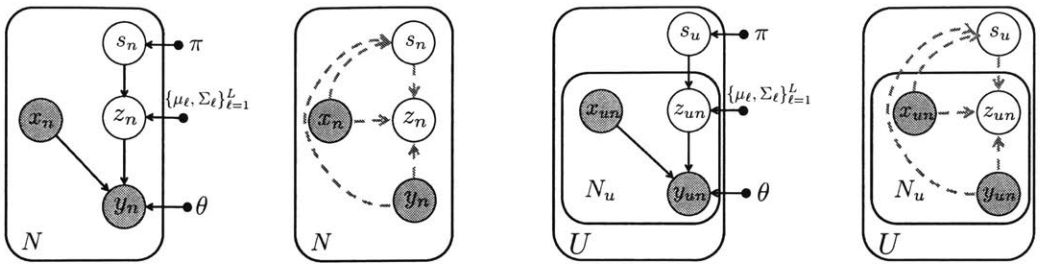


Figure 6.3.2: **Graphical models for CGM-VAE and P-VAE** (a) The graphical model for CGM-VAE introduced in Section 6.3.1 (b) The dependency structure for the variational approximations $q_{\phi_s}(s|x, y)$ and $q_{\phi_z}(z|x, y, s)$ in CGM-VAE (c) The P-VAE model introduced in Section 6.3.2) for categorization and personalization. There are U users and each user u has N_u photos. (d) The dependency structure in the variational distributions for the P-VAE model. Note that the recognition network for s_u depends on all the images and their corresponding slider values of user u.

Given this generative model and variational family, to perform inference we maximize a variational lower bound on $\log p_\theta(y|x)$, writing the objective as

$$\mathcal{L}(\theta, \phi) \triangleq \mathbb{E}_{q_\phi(s,z|x,y)}[\log p_\theta(y|z, x)] - \mathrm{KL}(q_\phi(s, z|x, y)||p_\theta(s, z)).$$

By marginalizing over the latent cluster assignments s, the CGM-VAE objective can be optimized using stochastic gradient methods and the reparameterization trick. Marginalizing out the discrete latent variables is not computationally intensive since s and y are conditionally independent given z, $p_\theta(s, z)$ is cheap to compute relative to $p_\theta(y|x, z)$, and we use a relatively small number of clusters. However, with a very large discrete latent space, one could use alternate approaches such as the Gumbel-Max trick (Maddison et al., 2016) or REBAR (Tucker et al., 2017).

6.3.2 Categorization and personalization

In order to categorize the users based on their adjustment style, we extend the basic CGM-VAE model to a hierarchical model that clusters users based on the edits they make. We call this new model personalized VAE or *P-VAE* for short. While the model in the previous section considered each image-edit pair x_n, y_n in isolation, we now organize the data according to U distinct users, using x_{un} to denote the nth image of user u and y_{un} to denote the corresponding slider values (see Fig. 6.3.2(c)). N_u denotes the number of photos edited by user u. As before, we assume a GMM with L components $\{\mu_\ell, \Sigma_\ell\}_{\ell=1}^L$ and mixing weights π, but here these clusters will model differences between users.

For each user u we sample a cluster index s_u to indicate the user's category, then for each photo $n \in \{1, \ldots, N_u\}$ we sample the latent attribute vector z_{un} from the corresponding mixture

component:

$$s_u|\pi \overset{\text{iid}}{\sim} \pi, \quad z_{un}|s_u, \{(\mu_\ell, \Sigma_\ell)\}_{\ell=1}^{L} \overset{\text{iid}}{\sim} \mathcal{N}(\mu_{s_u}, \Sigma_{s_u}).$$

Finally, we use the latent features z_{un} to generate the vector of suggested slider values y_{un}. As before, we use a multivariate normal distribution with mean and variance generated from an MLP parameterized by θ:

$$y_{un}|x_{un}, z_{un}, \theta \overset{\text{iid}}{\sim} \mathcal{N}(\mu(z_{un},x_{un};\theta), \Sigma(z_{un}, x_{un};\theta)).$$

For inference in the P-VAE model, our goal is to maximize the following variational lower bound (VLB) for a dataset of U users:

$$\begin{aligned}\mathcal{L}(\theta,\phi) \triangleq \frac{1}{U}\sum_u \Big[\sum_{n=1}^{N_u} \mathbb{E}_{q_\phi(z,s|x,y)}[\log p_\theta(y_{un}|x_{un}, z_{un})] \\ - \mathrm{KL}(q_\phi(z_{un}|x_{un}, y_{un}, s_u)||p_\theta(z_{un}|s_u))\Big] \\ - \mathrm{KL}(q_\phi(s_u|\{x_{un}, y_{un}\}_{n=1}^{N_u})||p(s_u)).\end{aligned}$$

In the following we define the variational factors and the recognition networks that we use.

Variational factors For the local variables z and s, we restrict $q(z|s)$ to be normal and we have $q(s)$ in the categorical form. As in the CGM-VAE, we marginalize over cluster assignments at the user level. Fig. 6.3.2 (parts c and d) outlines the graphical model structures of the P-VAE and its variational distributions q_ϕ.

For the variational factor of the latent mixture component index s_u, we write:

$$q(s_u|\{x_{un}, y_{un}\}_{n=1}^{N_u};\phi) \propto \exp\left\{\langle \log\pi + \sum_{n=1}^{N_u} \log r(y_{un}, x_{un};\phi),\ t_s(s_u)\rangle\right\}, \tag{6.3}$$

where $t_s(s_u)$ denotes the one-hot vector encoding of s_u and $r(y_{un}, x_{un};\phi)$ is the recognition network that belongs to some parametrized class of functions. That is, for each user image x_{un} and corresponding set of slider values y_{un}, the recognition network produces a potential over the user's latent mixture component s_u. These image-by-image guesses are then combined with each other and with the prior to produce the inferred variational factor on s_u.

This recognition network architecture is both natural and convenient. It is natural because a powerful enough r can set $r_k(y_{un}, x_{un};\phi) \propto p_\theta(y_{un}|x_{un}, s_u = k)$, in which case $q_\phi(s_u|\{x_{un}, y_{un}\}_{n=1}^{N_u}) \equiv p(s_u|\{x_{un}, y_{un}\}_{n=1}^{N_u})$ and there is no approximation error. It is convenient because it analyzes image-edit pairs independently, and these evidence potentials are combined in a symmetric, ex-

changeable way that extends to any number of user images N_u.

6.4 Experiments

We evaluate our models and several strong baselines on three datasets. We focus on the photo editing software Adobe Lightroom. The datasets that we use cover three different types of users that can be roughly described as 1) casual users who do not use the application regularly, 2) frequent users who have more familiarity with the application and use it more frequently 3) experts who have more experience in editing photos than the other two groups. We randomly split all three datasets into 10% test, 10% validation, and 80% train set. For more details on the datasets, baselines and hyperparameter settings, see the supplementary materials.

Datasets The casual users dataset consists of 345000 images along with the slider values that a user has applied to the image in Lightroom. There are 3200 users in this dataset. Due to privacy concerns, we only have access to the extracted features from a CNN applied to the images. Hence, each image in the dataset is represented by a 1024-dimensional vector. For the possible edits to the image, we only focus on 11 basic sliders in Lightroom. Many common editing tasks boil down to adjusting these sliders. The 11 basic sliders have different ranges of values, so we standardize them to all have a range between -1 and 1 when training the model.

Dataset	**Casual**		**Frequent**		**Expert**		
Eval. Metric	LL	JSD	LL	JSD	LL	JSD	LAB
MLP	-15.71 ± 0.21	0.26 ± 0.04	-2.72 ± 0.31	0.11 ± 0.02	-4.28 ± 0.12	0.22 ± 0.06	7.81 ± 0.26
LBN	-7.12 ± 0.15	0.14 ± 0.02	-3.7 ± 0.43	0.13 ± 0.02	-4.89 ± 0.24	0.17 ± 0.04	7.44 ± 0.29
MDN	-14.53 ± 0.25	0.31 ± 0.06	-1.67 ± 0.47	0.24 ± 0.08	-4.91 ± 0.07	0.28 ± 0.11	8.41 ± 0.27
CGM-VAE	$\mathbf{-6.39 \pm 0.11}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{-1.42 \pm 0.18}$	$\mathbf{0.08 \pm 0.02}$	$\mathbf{-2.6 \pm 0.15}$	$\mathbf{0.12 \pm 0.05}$	$\mathbf{6.72 \pm 0.27}$

Table 6.1: **Quantitative results:** *LL*: Predictive log-likelihood for our model CGM-VAE and the three baselines. The predictive log-likelihood is calculated over the test sets from all three datasets. *JSD*: Jensen-Shannon divergence between normalized histograms of the true sliders and our model predictions over the test sets (lower is better). See Fig. 6.4.3 for an example of these histograms. *LAB*: LAB error between the images retouched by the experts and the images retouched by the model predictions. For each image we generate 3 proposals and compare that with the images generated by the top 3 active experts in the experts dataset.

The frequent users dataset contains 45000 images (in the form of CNN features) and their corresponding slider values. There are 230 users in this dataset. These users generally apply more changes to their photos compared to the users in the casual group.

Finally, the expert users dataset (Adobe-MIT5k, collected by Bychkovsky et al. 2011) contains 5000 images and edits applied to these images by 5 different experts, for a total of 25000 edits.

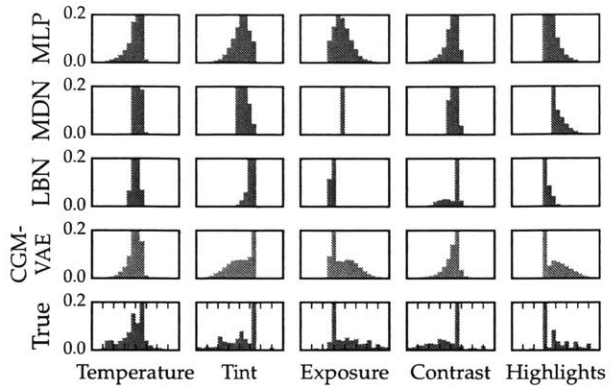


Figure 6.4.3: **Marginal statistics for the prediction of the sliders in the casual users dataset (test set).** Due to space limitation, We only display the top 5 mostly used sliders in the dataset. LBN has limited success compared to CGM-VAE. MLP mostly concentrates around the mean edit. The quantitative comparison between different methods in terms of the distance between normalized histograms is provided in Table 6.1.

We augment this dataset by creating new images after applying random edits to the original images. To generate a random edit from a slider, we add uniform noise from a range of $\pm 10\%$ of the total range of that slider. Given the augmented set of images, we extract the "FC7" features of a VGG-16 (Simonyan and Zisserman, 2014) pretrained network and use the 4096-dimensional feature vector as a representation of each image in the dataset. After augmenting the dataset, we have 15000 images and 75000 edits in total. Similar to other datasets, we only focus on the basic sliders in Adobe Lightroom.

Baselines We compare our model for multimodal prediction with several models: a multilayer perceptron (MLP), mixture density network (MDN), and linearizing belief network (LBN). The MLP is trained to predict the mean and variance of a multivariate Gaussian distribution; this model will demonstrate the limitations of even a strong model that makes unimodal predictions. The MDN and LBN, which are specifically designed for multimodal prediction, are other baselines for predicting multimodal densities. Table 6.1 summarizes our quantitative results.

We use three different evaluation metrics to compare the models. The first metric is the predictive log-likelihood computed over a held-out test set of different datasets. Another metric is the Jensen-Shannon divergence (JSD) between normalized histograms of marginal statistics of the true sliders and the predicted sliders. Fig. 6.4.3 shows some histograms of these marginal statistics for the casual users.

Finally, we use the mean squared error in the CIE-LAB color space between the expert-retouched image and the model-proposed image. We use the CIE-LAB color space as it is more perpetually

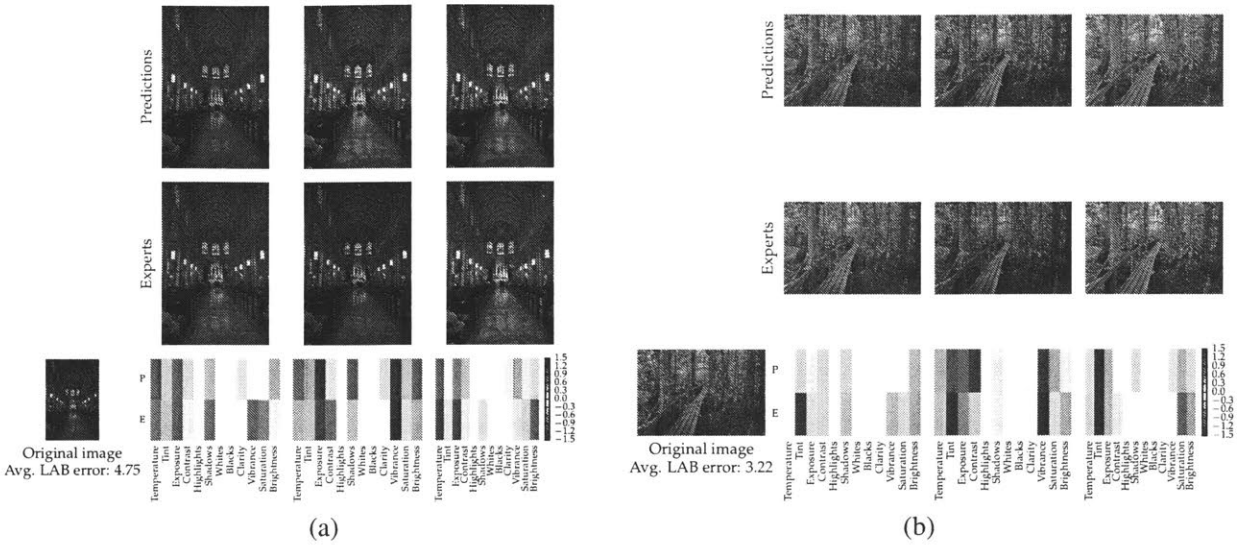


Figure 6.4.4: **Multimodal photo edits.** Sample slider predictions from the CGM-VAE model (denoted by P in the figure) compared to the edits of 3 most active experts in the expert users dataset (denoted by E). The images are selected from the test subset of the dataset; the 3 samples are selected from a set of 10 proposals from the CGM-VAE model such that they align with the experts. To show the difference between the model and experts, we apply their sliders to the original image. For more examples, refer to Appendix B.

linear compared to RGB. We only calculate this error for the experts dataset (test set) since that is the only dataset with available retouched images. To compute this metric, we first apply the predicted sliders from the models to the original image and then convert the generated RGB image to a LAB image. For reference the difference between white and black in CIE-LAB is 100 and photos with no adjustments result in an error of 10.2 . Table 6.1, shows that our model outperforms the baselines across all these metrics.

Hyperparameters For the CGM-VAE model, we choose the dimension of the latent variable from $\{2, 20\}$ and the number of mixture components from the set $\{1, 3, 5, 10\}$. Note that by setting the number of mixture components to 1, CGM-VAE will reduce to a conditional VAE. For the remaining hyperparameters see the supplementary materials. We choose the best hyperparameter setting based on the VLB of the held-out dataset. In all three datasets, 3 mixture components provide the best VLB for the validation datasets.

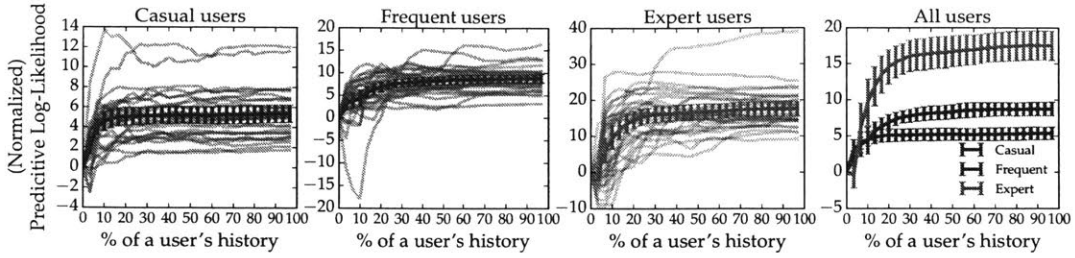


Figure 6.4.5: **Predictive log-likelihood for users in the test set of different datasets.** For each user in the test set, we compute the predictive log-likelihood of 20 images, given 0 to 30 images and their corresponding sliders from the same user. 30 sample trajectories and the overall average $\pm$ s.e. is shown for casual, frequent and expert users. The figure shows that knowing more about the user (up to around 10 images) can increase the predictive log-likelihood. The log-likelihood is normalized by subtracting off the predictive log-likelihood computed given zero images. Note the different y-axis in the plots. The rightmost plot is provided for comparing the average predictive log-likelihood across datasets.

Tasks In addition to computing the predictive log-likelihood and JSD over the held-out test sets for all three datasets, we consider the following two tasks:

1. Multimodal prediction: We predict different edits applied to the same image by the users in the experts dataset. Our goal is to show that CGM-VAE is able to capture different styles from the experts.

2. Categorizing the users and adapting the predictions based on users' categories: We show that the P-VAE model, by clustering the users, makes better predictions for each user. We also illustrate how inferred user clusters differ in terms of edits they apply to similar images.

6.4.1 Multimodal predictions

To show that the model is capable of multimodal predictions, we propose different edits for a given image in the test subset of the experts dataset. To generate these edits, we sample from different cluster components of our CGM-VAE model trained on the experts dataset. For each image we generate 20 different samples and align these samples to the experts' sliders. From the 5 experts in the dataset, 3 propose a more diverse set of edits compared to the others; hence, we only align our results to those three to show that the model can reasonably capture a diverse set of styles.

For each image in the test set, we compare the predictions of MLP, LBN, MDN and the CGM-VAE with the edits from the 3 experts. In MLP (and also MDN), we draw 20 samples from the Gaussian (mixture) distribution with parameters generated from the MLP (MDN). For the LBN, since the network has stochastic units, we directly sample 20 times from the network. We align

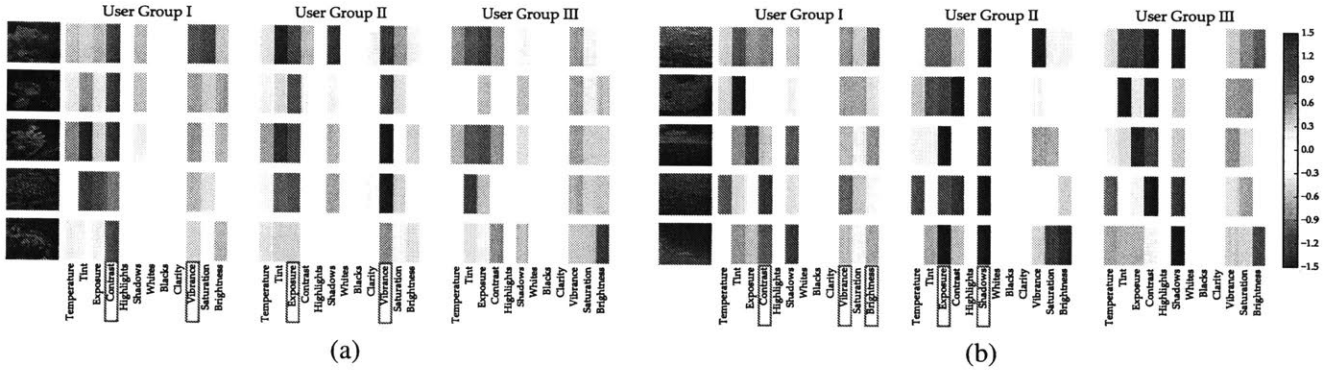


Figure 6.4.6: **User categorization.** Two examples of sample edits for three different user groups which the P-VAE model has identified (in the experts dataset). (a) For similar flower photos, users in group I prefer to use low contrast and vibrance, whereas group II users tend to increase the exposure and vibrance from their default values. There is also group III users which do not show any specific preference for similar flower photos. (b) The same user groups for another set of similar photos with dominant blue colors. For more examples, see the supplementary materials.

these samples to the experts' edits and find the LAB error between the expert-retouched image and the model-proposed image.

To report the results, we average across the 3 experts and across all the test images. The LAB error in Table 6.1 indicates that CGM-VAE model outperforms other baselines in terms of predicting expert edits. Some sample edit proposals and their corresponding LAB errors are provided in Fig. 6.4.4. This figure shows that the CGM-VAE model can propose a diverse set of edits that is reasonably close to those of experts. For further examples see the supplementary material.

6.4.2 Categorization and personalization

Next, we demonstrate how the P-VAE model can leverage the knowledge from a user's previous edits and propose better future edits. For the users in the test sets of all three datasets, we use between 0 and 30 image-slider pairs to estimate the posterior of each user's cluster membership. We then evaluate the predictive log-likelihood for 20 other slider values conditioned on the images and the inferred cluster memberships.

Fig. 6.4.5 depicts how adding more image-slider combinations can generally improve the predictive log-likelihood. The log-likelihood is normalized by subtracting off the predictive log-likelihood computed given zero images. The effect of adding more images is shown for 30 different sampled users; the overall average for the test dataset is also shown in the figure. To compare how various datasets benefit from this model, the average values from the 3 datasets are overlaid. According to this figure, the frequent users benefit more than the casual users and the expert users benefit the

most.[3]

To illustrate how the trained P-VAE model proposes edits for different user groups, we use a set of similar images in the experts dataset and show the predicted slider values for those images. Fig. 6.4.6 shows how the inferred user groups edit a group of similar images (*i.e.,* flowers). This figure provides further evidence that the model is able to propose a diverse set of edits across different groups; moreover, it shows each user group may have a preference over which slider to use. For more examples see the supplementary material.

6.5 Conclusion

We proposed a framework for multimodal prediction of photo edits and extend the model to make personalized suggestions based on each user's previous edits. Our framework outperforms several strong baselines and demonstrates the benefit of having interpretable latent structure in VAEs. Although we only applied our framework to the data from photo editing applications, it can be applied to other domains where multimodal prediction, categorization and personalization are essential. Our proposed models could be extended further by assuming more complicated graphical model structure such as admixture models instead of the Gaussian mixture model that we used. Furthermore, the categories learned by our model can be utilized to gain insights about the types of the users in the dataset.

[3]To apply the P-VAE model to the experts dataset, we split the image-slider combinations from each of the 5 experts into groups of 50 image-sliders and pretend that each group belongs to a different user. This way we can have more users to train the P-VAE model. However, this means the same expert may have some image-sliders in both train and test datasets. The significant advantage gained in the experts dataset might be due in part to this way of splitting the experts. Note that there are still no images shared across train and test sets.

CHAPTER 7

Improving variational inference for latent variable models: discrete particle variational inference[1]

7.1 Introduction

In Chapters 3 to 6, we proposed latent variable models with discrete latent structure. Probabilistic models defined over large collections of discrete random variables have arisen in multiple fields in addition to user behavior modeling. Examples include hidden Markov models for sequential data; Bayesian networks; mixture models for tabular and relational data; and discrete Markov random field models, which have become popular in fields ranging from computer vision to information extraction. These models are typically specified in terms of a probability distribution $P(x)$ defined over a collection of variables $x = \{x_n\}$ occupying points in an underlying discrete space $\mathcal{X}$. One key approximate inference problem that arises in many applications is identifying high-probability configurations of the discrete variables.

As mentioned in Chapter 2, approximate inference algorithms can be grouped into two main categories: Monte Carlo methods and variational methods.The flexibility and simplicity of Monte Carlo methods have made them the workhorse of statistical computation (Robert and Casella, 2004). There are two basic approaches to Monte Carlo inference for discrete probabilistic models. The first, Markov chain Monte Carlo methods, work by running a Markov chain with transition operator T whose equilibrium distribution asymptotically approaches $P(x)$—that is, $T^k P_0(x) \approx P(x)$ for any initial distribution on states $P_0(x)$. The second, sequential Monte Carlo methods, build up a sample from a distribution that approximates $P(x)$ by sampling from a sequence of more tractable distributions, typically defined over subspaces of $\mathcal{X}$. However, their accuracy is difficult to measure, and the amount of computation required for satisfactory accuracy can be prohibitive in practice.

[1]This chapter is based on the work by Saeedi et al. (2017b) (http://jmlr.org/papers/v18/15-615.html).

In contrast to Monte Carlo methods, variational methods tend to converge quickly, guarantee an improved bound after each iteration, and supply an easily monitored objective function (unlike Monte Carlo methods). However, for complex discrete models, the bias induced by variational approximations can sometimes lead to poor predictive performance. For example, consider a discrete probabilistic model where two binary variables x_1 and x_2 are constrained to take the same value, i.e. $P(x_1, x_2) = 0$ if $x_1 \neq x_2$. All the probability mass is on the states $x_1 = x_2 = 0$ and $x_1 = x_2 = 1$. In "mean-field" variational inference, this distribution might be approximated as $Q_{\theta_1}(x_1)Q_{\theta_2}(x_2)$, with θ_1 and θ_2 representing coin weights that used to model x_1 and x_2 as independent Bernoulli distributions. This family of variational approximations to $P(x_1, x_2)$ cannot capture the true distribution, and in fact cannot qualitatively capture the simple constraint that $x_1 = x_2$. These limitations prompted the development of more sophisticated approximations (*e.g.,* Bouchard-Côté and Jordan, 2009; Jaakkola and Jordan, 1998), but these incur additional computational cost and can be difficult or impossible to apply to a given problem.

This chapter introduces a new approximate inference method, called *discrete particle variational inference* (DPVI), that aims to combine key strengths of both Monte Carlo and variational inference. The key insight in DPVI is to use a weighted collection of samples — the kind of "particle approximation" output by Monte Carlo methods — as the approximating family for variational inference. Suppose we got to pick where to place the particles in the hypothesis space; where would we put them? Intuitively, we would want to distribute them in such a way that they cover high probability regions of the target distribution, but without the particles all devolving onto the mode of the distribution. This problem can be formulated precisely within the framework of variational inference. We derive a coordinate ascent update for particle approximations that iteratively minimizes the Kullback-Leibler (KL) divergence between the particle approximation and the target distribution.

In DPVI, the location of the particles become the parameters of the approximating family. This simple choice has appealing consequences. Like Monte Carlo, DPVI can handle problems where the posterior has multiple modes, and yields exact results in a well-defined limit (as the number of particles goes to infinity). Like standard mean-field variational methods, DPVI is based on optimizing a lower bound on the partition function; when this quantity is not of intrinsic interest, it facilitates convergence assessment and debugging. Like both Monte Carlo and combinatorial search, DPVI can take advantage of factorization, sequential structure, and custom search operators.

The rest of the chapter is organized as follows. After introducing our general framework, we describe how it can be applied to filtering and smoothing problems. We then show experimentally that variational particle approximations can overcome a number of problems that are challenging for conventional Monte Carlo methods. In particular, our approach is able to produce a diverse, high probability set of particles in situations where Monte Carlo and mean-field variational methods

sometimes degenerate.

7.2 Background

Our goal is to approximate a probability distribution $P(x)$ over discrete latent variables $x = \{x_1, \ldots, x_N\}, x_n \in \{1, \ldots, M_n\}$, where the target distribution is known only up to a normalizing constant Z: $P(x) = f(x)/Z$.

We assume that $P(x)$ is a Markov network defined on a graph G, so that $f(x)$ factorizes according to:

$$f(x) = \prod_c f_c(x_c), \tag{7.1}$$

where $c \subseteq \{1, \ldots, N\}$ indexes the maximal cliques of G.

Recall from Chapter 2 that a general way to approximate $P(x)$ is with a weighted collection of K particles, $\{x^1, \ldots, x^K\}$. For discrete latent variables, we have:

$$P(x) \approx Q(x) = \sum_{k=1}^{K} w^k \delta[x, x^k], \tag{7.2}$$

where $x^k = \{x_1^k, \ldots, x_N^k\}, x_n^k \in \{1, \ldots, M_n\}$ and $\delta[\cdot, \cdot] = 1$ if its arguments are equal and 0 otherwise. x^ks are stochastically generated particles from a proposal distribution, $x^k \sim \phi(\cdot)$, and their weights are computed according to $w^k \propto f(x^k)/\phi(x^k)$.

In an SMC framework, where we sequentially sample the latent variables at each time point using a proposal distribution $\phi(x_n|x_{n-1})$, the procedure can produce conditionally low probability particles; therefore, most algorithms include a resampling step which replicates high probability particles and kills off low probability particles. The downside of resampling is that it can produce degeneracy: the particles become concentrated on a small number of hypotheses, and consequently the effective number of particles is low.

We next consider particle approximations from the perspective of variational inference. We then turn to the application of particle approximations to inference in stochastic dynamical systems.

7.3 Variational particle approximations

Variational inference can be connected to Monte Carlo methods by viewing the particles as a set of variational parameters parameterizing Q. For the particle approximation defined in Eq. (7.2), the

negative variational free energy takes the following form:

$$\mathcal{L}[Q] = \sum_{k=1}^{K} w^k \log \frac{f(x^k)}{w^k V^k}, \tag{7.3}$$

where $V^k = \sum_{j=1}^{K} \delta[x^j, x^k]$ is the number of times an identical replica of x^k appears in the particle set. We wish to find the set of K particles and their associated weights that maximize $\mathcal{L}[Q]$, subject to the constraint that $\sum_{k=1}^{K} w^k = 1$. This constraint can be implemented by defining a new functional with Lagrange multiplier λ:

$$\tilde{\mathcal{L}}[Q] = \mathcal{L}[Q] + \lambda \left(\sum_{k=1}^{K} w^k - 1 \right). \tag{7.4}$$

Taking the functional derivative of the Lagrangian with respect to w^k and equating to zero, we obtain:

$$\begin{aligned} \frac{\partial \tilde{\mathcal{L}}[Q]}{\partial w^k} &= \log f(x^k) - \log w^k - \log V^k + \lambda - 1 = 0 \\ &\Longrightarrow w^k = Z_Q^{-1} f(x^k)/V^k, \end{aligned} \tag{7.5}$$

where

$$Z_Q = \exp(\lambda - 1)^{-1} = \sum_{k=1}^{K} \frac{f(x^k)}{V^k}. \tag{7.6}$$

We can plug the above result back into the definition of $\mathcal{L}[Q]$:

$$\begin{aligned} \mathcal{L}[Q] &= Z_Q^{-1} \sum_{k=1}^{K} \frac{f(x^k)}{V^k} \log \frac{f(x^k) V^k}{Z_Q^{-1} f(x^k) V^k} \\ &= Z_Q^{-1} \sum_{k=1}^{K} \frac{f(x^k)}{V^k} \log Z_Q \\ &= \log Z_Q \end{aligned} \tag{7.7}$$

Thus, $\mathcal{L}[Q]$ is maximized by choosing the K values of x with the highest score. The following theorem shows that allowing $V^k > 1$ (i.e., having replica particles) can never improve the bound.

Theorem 1. Let Q and Q' denote two particle approximations, where Q consists of unique particles ($V_Q^k = 1$ for all k) and Q' is identical to Q except that particle $x_{Q'}^j$ is replicated $V_{Q'}^j$ times (displacing $V_{Q'}^j$ other particles with cumulative score F). For any choice of particles, $\mathcal{L}[Q] \geq \mathcal{L}[Q']$.

Proof. We first apply Jensen's inequality to obtain an upper bound on $\mathcal{L}[Q']$:

$$\mathcal{L}[Q'] \leq \log \sum_{k=1}^{K} w_{Q'}^k Z_{Q'} = \log \sum_{k=1}^{K} \frac{f(x_{Q'}^k)}{V_{Q'}^k}. \tag{7.8}$$

Since $\mathcal{L}[Q] = \log Z_Q$, we wish to show that $Z_Q \geq \sum_{k=1}^{K} \frac{f(x_{Q'}^k)}{V_{Q'}^k}$. All the particles in Q and Q' are identical except for the $V_{Q'}^j$ particles in Q that were displaced by replicas of $x_{Q'}^j$ in Q'; thus we only need to establish that $f(x_{Q'}^j) + F \geq \frac{V_{Q'}^j f(x_{Q'}^j)}{V_{Q'}^j} = f(x_{Q'}^j)$, where the left hand side of the inequality is the change in negative variational free energy after the replication of particles. Since the score $f(x)$ can never be negative, the cumulative score F can also never be negative ($F \geq 0$) and the inequality holds for any choice of particles. □

The variational bound can be optimized by coordinate ascent, as specified in Algorithm 1, which we refer to as *discrete particle variational inference* (DPVI). This algorithm takes advantage of the fact that when optimizing the bound with respect to a single variable, only potentials local to that variable need to be computed. In particular, let $\tilde{x}^k$ be a replica of x^k with a single-variable modification, $\tilde{x}_n^k = m$. We can compute the unnormalized probability of this particle efficiently using the following equation:

$$f(\tilde{x}^k) = f(x^k) \frac{\mathcal{F}_n(\tilde{x}^k)}{\mathcal{F}_n(x^k)} \tag{7.9}$$

where $\mathcal{F}_n(x) = \prod_{c:n \in c} f_c(x_c)$. The variational bound for the modified particle can then be computed using Eq. (7.7). Particles can be initialized arbitrarily. When repeatedly iterated, DPVI will converge to a local maximum of the negative variational free energy.[2] Note that in principle more sophisticated methods can be used to find the top K modes (*e.g.,* Flerova et al., 2012; Yanover and Weiss, 2003); however, we have found that this coordinate ascent algorithm is fast, easy to implement, and very effective in practice (as our experiments below demonstrate).

An important aspect of this framework is that it maintains one of the same asymptotic guarantees as importance sampling: Q converges to P as $K \to \infty$, since in this limit DPVI is equivalent to exact inference. Thus, DPVI combines advantages of variational methods (monotonically decreasing KL divergence between Q and P) with the asymptotic correctness of Monte Carlo methods. It is important to note that asymptotic correctness might be useless in practice unless something is known about the convergence rate. This issue is not unique to DPVI; it also applies to Monte

[2]Naturally, initialization affects performance, since the objective function has local optima. For example, if the posterior is multimodal and none of the particles are initialized near the dominant mode, then the particle approximation will likely miss a significant portion of the probability mass. Studying the effects of initialization is an important practical challenge for the application of DPVI. In our experiments, we report averages across multiple random initializations.

Algorithm 1 Discrete particle variational inference

/* N is the number of latent variables */
/* x^k is the set of all latent variables for the kth particle: $x^k = \{x_1^k, \ldots, x_N^k\}$ */
/* M_n is the support of latent variable x_n */
Input: initial particle approximation Q with K particles, tolerance ϵ
while $|\mathcal{L}[Q] - \mathcal{L}[Q']| > \epsilon$ **do**
  for $n = 1$ to N **do**
    $\mathcal{X} = \emptyset$
    for $k = 1$ to K **do**
      Copy particle k: $\tilde{x}^k \leftarrow x^k$
      for $m = 1$ to M_n **do**
        Modify particle: $\tilde{x}_n^k \leftarrow m$
        Score $\tilde{x}^k$ using Eq. (7.9)
        $\mathcal{X} \leftarrow \mathcal{X} \cup (\tilde{x}^k, f(\tilde{x}^k))$
      end for
    end for
    Select K unique particles from $\mathcal{X}$ with the largest scores
    Construct new particle approximation $Q'(x) = \sum_{k=1}^{K} w^k \delta[x, x^k]$
    Compute variational bound $\mathcal{L}[Q']$ using Eq. (7.7)
  end for
end while Return particle approximation Q'

Carlo and variational methods. For certain Markov chain Monte Carlo (MCMC) samplers, it can be shown that the chain converges to the posterior at a geometric rate (Mengersen et al., 1996; Meyn and Tweedie, 1993). A small amount of work has investigated convergence properties of variational methods for specific models (Hall et al., 2011; Wang et al., 2006), but in general the issue of convergence rate for variational methods is an open question.

The asymptotic complexity of DPVI in the sequential setting is $O(SNK)$ where S is the maximum support size of the latent variables. For the iterative update of the particles the complexity is $O(TCSK)$, where T is the maximum number of iterations until convergence and C is the maximum clique size. In our experiments, we empirically observed that we only need a small number of iterations and particles in order to outperform our baselines.

7.4 Filtering and smoothing in hidden Markov models

We now describe how variational particle approximations can be applied to filtering and smoothing in hidden Markov models (HMMs). Consider a hidden Markov model with observations $y =$

$\{y_1, \ldots, y_N\}$ generated by the following stochastic process:

$$P(y, x, \theta) = P(\theta) \prod_n P(y_n|x_n, \theta) P(x_n|x_{n-1}, \theta), \tag{7.10}$$

where θ is a set of transition and emission parameters. We are particularly interested in *marginalized* HMMs where the parameters are integrated out: $P(y, x) = \int_\theta P(y, x, \theta) d\theta$. This induces dependencies between observation n and all previous observations, making inference challenging.

Filtering is the problem of computing the posterior over the latent variables at time n given the history $y_{1:n}$. To construct the variational particle approximation of the filtering distribution, we need to compute the product of potentials for variable n:

$$\mathcal{F}_n(x) = P(y_n|x_{1:n}, y_{1:n-1}) P(x_n|x_{1:n-1}). \tag{7.11}$$

Recall from the previous section that $\mathcal{F}_n(x)$ is the joint probability of the maximal cliques to which x_n belongs. We can then apply the coordinate ascent update described in the previous section. This update is simplified in the filtering context due to the underlying Markov structure. Specifically, Eq. (7.9) is given by:

$$f(\tilde{x}^k) = f(x^k) P(y_n|x_n^k = m, x_{1:n-1}^k, y_{1:n-1}) P(x_n^k = m|x_{1:n-1}^k). \tag{7.12}$$

At each time step, the algorithm selects the K continuations (new variable assignments of the current particle set) that maximize the negative variational free energy.

Smoothing is the problem of computing the posterior over the latent variables at time n given data from both the past and the future, $y_{1:N}$. The product of potentials is given by:

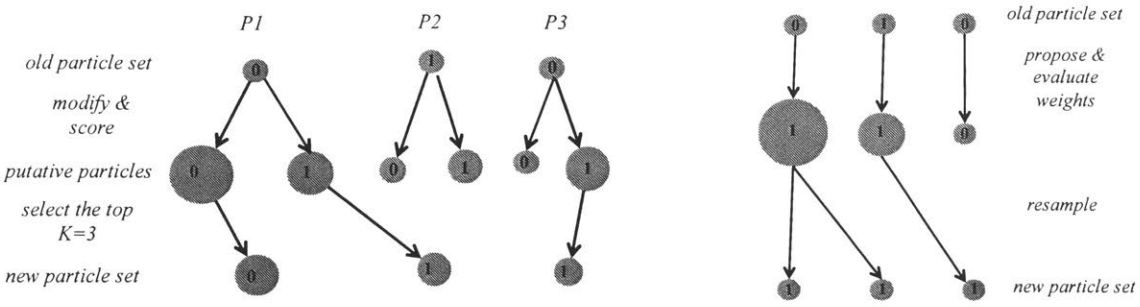
$$\mathcal{F}_n(x) = P(y_n|x_{1:n}, y_{-n}) P(x_n|x_{-n}), \tag{7.13}$$

where x_{-n} refers to all the latent variables except x_n (and likewise for y_{-n}). This potential can be plugged into the updates described in the previous section.

To understand DPVI applied to filtering problems, it is helpful to contemplate three possible fates for a particle at time n (illustrated in Fig. 7.4.1):

- **Selection**: A single continuation of particle k has non-zero weight. This can be seen as a deterministic version of particle filtering, where the sampling operation is replaced with a max operation.

- **Splitting**: Multiple continuations of particle k have non-zero weight. In this case, the particle is split into multiple particles at the next iteration.



(A) DPVI (B) Particle Filtering

Figure 7.4.1: **Schematic of DPVI versus particle filtering for filtering problems**. Illustration of different filtering scenarios over 2 time steps in a binary state space with $K = 3$ particles. The number in each circle indicates the binary value of the corresponding variable. Arrows indicate the evolution of the particles. (A) DPVI: The size of the putative particles represents the score of the particle. The K continuations with highest score are selected for propagation to the next time step. The size of the new particle set corresponds to the normalized score. Particle $P1$ is split, $P2$ is deleted and one putative particle from $P3$ is selected. (B) Particle filtering: The size of the node represents the weight of the particle for the resampling step.

- **Deletion**: No continuations of particle k have non-zero weight. In this case, the particle is deleted from the particle set.

Similar to particle filtering with resampling, DPVI deletes and propagates particles based on their probability. However, as we show later, DPVI is able to escape some of the problems associated with resampling.

7.5 Related work

DPVI is related to several other ideas in the statistics literature:

- DPVI is a special case of a *mixture mean-field variational approximation* (Jaakkola and Jordan, 1998; Lawrence, 2000):

$$Q(x) = \sum_{k=1}^{K} Q(k) \prod_{n=1}^{N} Q(x_n|k). \tag{7.14}$$

  In DPVI, $Q(k) = w^k$ and $Q(x_n|k) = \delta[x_n, x_n^k]$. From the simple restriction that the component distributions must be delta functions, we derived a new algorithm that is simpler and more efficient than mixture mean-field (which requires separate updates for the mixture weights), while sacrificing some of its expressivity. Another distinct advantage of DPVI is

that the variational updates do not require the additional lower bound used in general mixture mean-field, due to the intractability of the mean-field updates.

- When $K = 1$, DPVI is equivalent to *iterated conditional modes* (ICM; Besag, 1986), which iteratively maximizes each latent variable conditional on the rest of the variables. This algorithm is simple to implement, efficient (relative to variational and Monte Carlo algorithms), and has been successfully applied to computer vision tasks such as reconstruction and segmentation of Markov random fields. However, the algorithm is susceptible to local optima without the aid of relaxation techniques like simulated annealing (Greig et al., 1989).

- DPVI is conceptually similar to nonparametric variational inference (Gershman et al., 2012), which approximates the posterior over a continuous state space using a set of particles convolved with a Gaussian kernel (see Miller et al., 2016, for more sophisticated extensions of this idea). This approach was shown to be effective for probabilistic models that lack the conjugate-exponential structure required for exact mean-field inference. Because nonparametric variational inference approximates continuous densities, it is inapplicable to the discrete problems considered here.

- Frank et al. (2009) used particle approximations within a variational message passing algorithm. The resulting approximation is "local" in the sense that the particles are used to approximate messages passed between nodes in a factor graph, in contrast to the "global" approximation produced by DPVI, which attempts to capture the distribution over the entire set of variables. It is an interesting question for future research to understand what classes of probabilistic models are better approximated using local vs. global approaches.

- Ionides (2008) described a truncated version of importance sampling in which weights falling below some threshold are set to the threshold value. Ionides (2008) showed that truncation reduced sensitivity to the proposal distribution and derived optimal truncations as a function of the number of samples. This is similar (though not equivalent) to the DPVI setting where latent variables are sampled exhaustively and without replacement.

- Schniter et al. (2008) described an approximate inference algorithm, *fast Bayesian matching pursuit*, which can be viewed as a special case of DPVI applied to Gaussian mixture models.

- In Jones et al. (2005), a *shotgun stochastic search* algorithm is suggested, which proposes local changes to the latent variables with probability proportional to the unnormalized posterior. While this method of evaluating local changes is similar to our coordinate algorithm for the DPVI objective, it is important to note that DPVI is not a stochastic search algorithm (unless $K = 1$): It maintains a collection of particles in order to approximate the posterior distribution, which is important for applications that require a representation of uncertainty.

- Finally, DPVI is closely related to the problem of finding the K most probable latent variable assignments (Flerova et al., 2012; Yanover and Weiss, 2003). We view this problem through the lens of particle approximations, connecting it to both Monte Carlo and variational methods. The techniques developed for finding the K-best assignments could be fruitfully applied to optimizing the DPVI objective function.

Our experimental strategy is to compare DPVI with popular algorithms that have similar computational complexity, which is why we focus on particle filtering, Gibbs sampling and mean-field approximations. Although mixture mean-field will by definition lead to a better approximation, its computational complexity is considerably greater, which may explain why it is has never achieved the popularity of standard mean-field approaches. Some of the approaches listed above are not applicable to the discrete setting (nonparametric variational inference and fast Bayesian matching pursuit), and some are point estimators instead of true probabilistic approximations (iterated conditional modes and shotgun stochastic search).

7.6 Experiments

In this section, we compare the performance of DPVI to several widely used approximate inference algorithms, including particle filtering, Gibbs sampling and variational methods. We first present a didactic example to illustrate how DPVI can sometimes succeed where particle filtering fails. We then apply DPVI to four popular but intractable probabilistic models: the Dirichlet process mixture model (DPMM; Antoniak, 1974; Escobar and West, 1995), the infinite HMM (iHMM; Beal et al., 2002; Teh et al., 2006b), the infinite relational model (IRM; Kemp et al., 2006) and the Ising model. The results for the DPMM and the Ising model are provided in Appendix C.

7.6.1 Didactic example: binary HMM

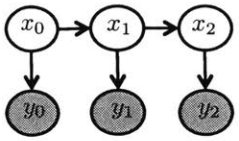
As a didactic example, we use a simple HMM with binary hidden states (x) and observations (y):

$$\begin{aligned} P(x_{n+1} = 0|x_n = 0) &= \alpha_0 \\ P(x_{n+1} = 1|x_n = 1) &= \alpha_1 \\ P(y_n = 0|x_n = 0) &= \beta_0 \\ P(y_n = 1|x_n = 1) &= \beta_1, \end{aligned} \tag{7.15}$$

with α_0, α_1, β_0, and β_1 all less than 0.5. Constraining the parameters to be less than 0.5 makes some sequences more likely; approximating the posterior using a particle filter (with resampling) may result in capturing only these sequences. We will use this model to illustrate how DPVI differs

(A)

DPVI	Monte Carlo	Mean-Field
$Q(x) = \sum_{k=1}^{K} w^k \prod_{n=1}^{N} \delta[x_n, x_n^k]$	$Q(x) = \frac{1}{K} \sum_{k=1}^{K} \prod_{n=1}^{N} \delta[\hat{x}_n, \hat{x}_n^k]$	$Q(x) = \prod_{n=1}^{N} Q(x_n)$



(B)

DPVI		Monte Carlo	Mean-Field
State Seq. $[x_0, x_1, x_2]$	Weight	State Seq. $[x_0, x_1, x_2]$	Mean-Field Parameters
[0,0,0]	0.15	[1,0,1]	$Q(x_0 = 0) = 0.4$
[0,1,1]	0.03	[1,1,1]	$Q(x_1 = 0) = 0.63$
[1,1,1]	0.32	[1,1,1]	$Q(x_2 = 0) = 0.15$
[1,0,0]	0.15	[1,1,1]	
[1,0,1]	0.35	[1,0,1]	
$Pr(x_0 = 1, x_1 = 0, x_2 = 1\|y) \approx 0.35$		$Pr(x_0 = 1, x_1 = 0, x_2 = 1\|y) \approx 0.4$	$Pr(x_0 = 1, x_1 = 0, x_2 = 1\|y) \approx 0.32$

Figure 7.6.2: **Comparison of approximate inference schemes.** (*A*) Approximating families for DPVI, Monte Carlo and mean-field. (*B*) Approximating the posterior probability of a sequence $(x_0 = 1, x_1 = 0, x_2 = 1)$ for the above 3 schemes on a binary HMM. Given the weights for different sequences in DPVI the posterior probability is the weight corresponding to that sequence. For Monte Carlo approximation, the posterior can be approximated from the normalized counts of sampled $(x_0 = 1, x_1 = 0, x_2 = 1)$ sequences. Finally, for the mean-field approximation, we have $Pr(x_0 = 1, x_1 = 0, x_2 = 1|y) = Q(x_0 = 1)Q(x_1 = 0)Q(x_2 = 1)$.

from particle filtering. Fig. 7.6.2 compares several inference schemes for this model.

For illustration, we use the following parameters: $\alpha_0 = 0.2$, $\alpha_1 = 0.1$, $\beta_0 = 0.3$, and $\beta_1 = 0.2$. Suppose you observe a sequence generated from this model. For a sufficiently long sequence, a particle filter with resampling will eventually delete most conditionally unlikely particles, due to the fact that there is some probability on each step that any given unlikely particle will be deleted. The particle filter will thus suffer from degeneracy for long sequences. On the other hand, without resampling the approximation will degrade over time because conditionally unlikely particles are never replaced by better particles. For this reason, it is sometimes suggested that resampling only be performed when the effective sample size (ESS) falls below some threshold.

The ESS is calculated as $\text{ESS} = \frac{1}{\sum_{k=1}^{K}(w^k)^2}$. A low ESS means that most of the weight is being placed on a small number of particles, and hence the approximation may be degenerate (although in some cases this may mean that the target distribution is peaky). We evaluated particle filtering with multinomial resampling on synthetic data generated from the HMM described above. Approx-

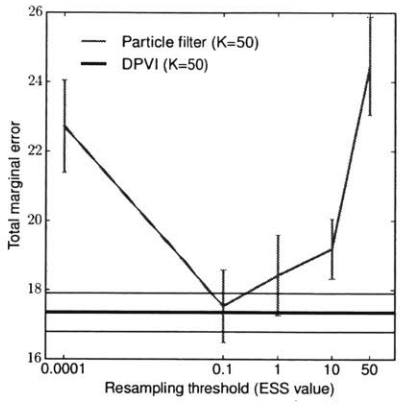


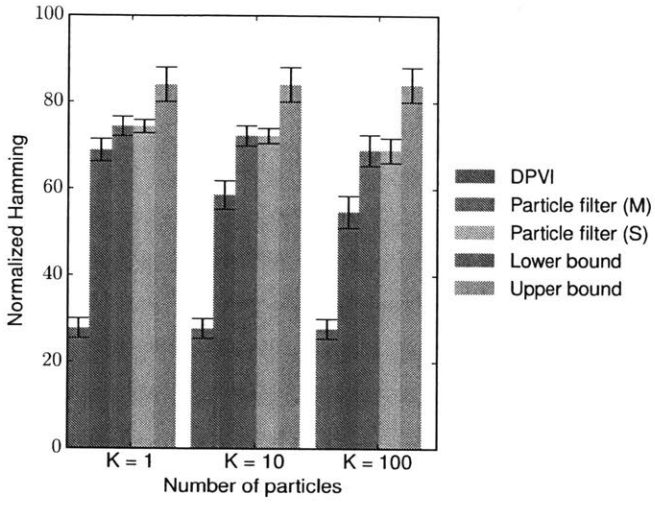
Figure 7.6.3: **HMM with binary hidden states and observations.** Total marginal error computed for a sequence of length 200. For particle filtering the total error for every ESS value is averaged over 5 sequences generated from the HMM; in addition, for each sequence we reran the particle filter 5 times (thus 25 runs total). Note the logarithmic scale of the x-axis. Error bars and the thin black lines correspond to standard error of the mean.

imation accuracy was measured by using the forward-backward algorithm to compute the hidden state posterior marginals exactly and then comparing these marginals to the particle approximation. Fig. 7.6.3 shows performance as a function of ESS threshold, demonstrating that there is a fairly narrow range of thresholds for which performance is good. Thus in practice, successful applications of particle filtering may require computationally expensive tuning of this threshold.

In contrast, DPVI achieves performance comparable to the optimal particle filter, but without a tunable threshold. This occurs because DPVI uses an implicit threshold that is automatically tuned to the problem. Instead of resampling particles, DPVI deletes or propagates particles deterministically based on their relative contribution to the variational bound. We can always incrementally add particles, unlike tuning the threshold. So although K can be viewed as a tuning parameter, we can adapt it with relatively little expense, monotonically increasing the approximation quality in a way that can be easily quantified.

7.6.2 Infinite HMM

We utilize DPVI for inference in the iHMM model introduced in Section 2.2.2. We show the performance of the model on a synthetic and two real datasets.



(A)

Method	Runtime (sec)
DPVI ($K = 1$)	1.28
DPVI ($K = 10$)	3.56
DPVI ($K = 100$)	204.42
PF ($K = 1$)	1.14
PF ($K = 10$)	1.92
PF ($K = 100$)	31.99

(B)

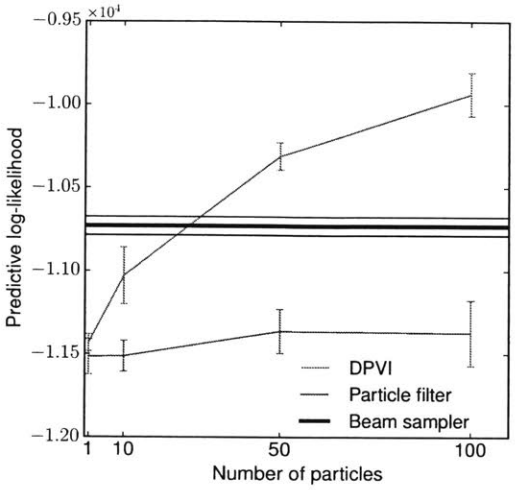
Figure 7.6.4: **Infinite HMM on the synthetic dataset** (*A*). Results on 500 synthetic data points generated from an HMM with 10 hidden states. Error is the Hamming distance between the true hidden sequence and the sampled sequence, averaged over 50 datasets. M: multinomial resampling; S: stratified resampling. Lower bound is the expected Hamming distance between data-generating distribution and ground truth. Upper bound is the expected Hamming distance between uniform distribution and ground truth. (*B*) Run time comparison for the synthetic iHMM dataset. We denote particle filtering method by PF.

Synthetic data

We generated 50 sequences with length 500 from 50 different HMMs, each with 10 hidden and 5 observed states. For the rows of the transition and initial probability matrices of the HMMs we used a symmetric Dirichlet prior with concentration parameter 0.1; for the emission probability matrix, we used a symmetric Dirichlet prior with concentration parameter 10.

Fig. 7.6.4A illustrates the performance of DPVI and particle filtering (with multinomial and stratified resampling) for varying numbers of particles ($K = 1, 10, 100$). Performance error was quantified by computing the Hamming distance between the true hidden sequence and the sampled sequence. The Munkres algorithm was used to maximize the overlap between the two sequences. The results show that DPVI outperforms particle filtering in all three cases.

When the data consist of long sequences, resampling at every step will produce degeneracy in particle filtering; this tends to result in a smaller number of clusters relative to DPVI. The superior accuracy of DPVI suggests that a larger number of clusters is necessary to capture the latent structure of the data. Not surprisingly, this leads to longer run times (Fig. 7.6.4B), but it is important to note that particle filtering and DPVI have comparable per-cluster time complexity.



(A)

Method	Runtime (sec)
DPVI ($K = 1$)	4.73
DPVI ($K = 10$)	41.62
DPVI ($K = 100$)	1685
PF ($K = 1$)	1.64
PF ($K = 10$)	28.08
PF ($K = 100$)	211.66
Beam sampler	260

(B)

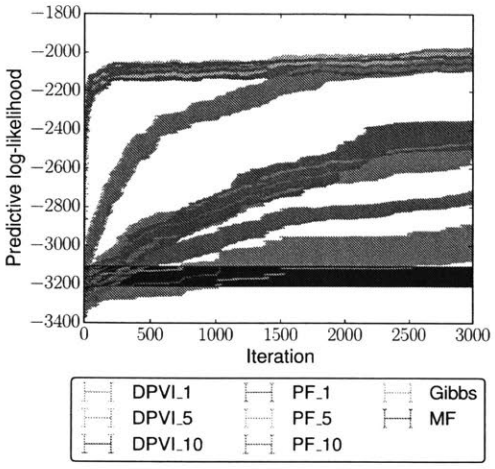
Figure 7.6.5: **Infinite HMM for the text analysis task.** (*A*) Predictive log-likelihood for the Alice in Wonderland dataset. (*B*) Results using the "Alice in Wonderland" dataset. The maximum number of inferred clusters is $\hat{C} = 147$ for DPVI ($K = 100$) and $\hat{C} = 21$ for particle filter ($K = 100$). As expected, the per cluster runtime of the two methods are comparable.

Text analysis

We next analyzed a real-world dataset, text taken from the beginning of "Alice in Wonderland", with 31 observation symbols (letters). We used the first 1000 characters for training, and the subsequent 4000 characters for test. Performance was measured by calculating the predictive log-likelihood. We fixed the hyperparameters α and γ to 1 for both DPVI and the particle filtering.

We ran one pass of DPVI (filtering) and particle filtering over the training sequence. We then sampled 50 datasets from the distribution over the sequences. We truncated the number of states and used the learned transition and emission matrices to compute the predictive log-likelihood of the test sequence. To handle the unobserved emissions in the test sequence we used "add-δ" smoothing with $\delta = 1$. Finally, we averaged over all the 50 datasets.

We also compared DPVI to the beam sampler (Van Gael et al., 2008), a combination of dynamic programming and slice sampling, which was previously applied to this dataset. For the beam sampler, we followed the setting of Van Gael et al. (2008). We run the sampler for 10000 iterations and collect a sample of hidden state sequence every 200 iterations. Fig. 7.6.4B shows the predictive log-likelihood for varying numbers of particles. Even with a small number of particles, DPVI can outperform both particle filtering and the beam sampler.



(A)

Method	Predictive LL
DPVI ($K = 1$)	-2983.24 ± 93.77
DPVI ($K = 5$)	$\mathbf{-2018.02 \pm 9.23}$
DPVI ($K = 10$)	$\mathbf{-2058.75 \pm 33.62}$
PF ($K = 1$)	-2743.60 ± 28.49
PF ($K = 5$)	-2503.95 ± 40.46
PF ($K = 10$)	-2424.01 ± 72.72
Mean field	-3158.69 ± 52.67
Gibbs	$\mathbf{-2030.50 \pm 56.55}$

(B)

Figure 7.6.6: **Infinite HMM results for the user behavior analysis task.** *(A)* Predictive log-likelihood vs iteration for the user behavior dataset. The error bars correspond to the standard error and are computed over 20 runs. In every iteration of DPVI and particle filtering, we do a forward filtering-backward smoothing pass over all the sequences of the dataset. *(B)* Predictive log-likelihood after 3000 iterations of DPVI, Gibbs, particle filter and mean field for the user behavior dataset (with 1000 data points as held-out). The best performance is achieved by DPVI with 5-10 particles and also the Gibbs sampler. Run time for each epoch is as follows: 0.726 sec for mean field, 0.593 sec for Gibbs, 4.52 sec for particle filter with 10 particles, and 4.74 sec for DPVI with 10 particles.

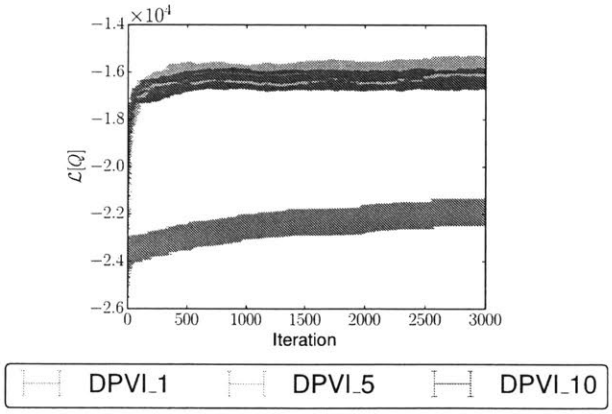


Figure 7.6.7: **Infinite HMM results for the user behavior analysis task.** Lower bound vs iteration for the user behavior dataset. The error bars correspond to the standard error and are computed over 20 runs. In every iteration of DPVI, we do a forward filtering-backward smoothing pass over all the sequences of the dataset.

User behavior analysis

We analyzed a dataset of user behavior in a photo editing software application. The dataset contains sequences of edits applied by users to different photos. An iHMM can be utilized for better understanding the high-level tasks of the users. That is, a sequence of multiple edits may be required to perform a high-level task such as cropping or masking a photo. Our dataset contains 30000 edits from which we use 1000 data points as a held-out set. There are 23 possible observations (edits) in the dataset. We compare DPVI with 1, 5 and 10 particles with particle filtering, Gibbs sampling and mean field inference schemes. For all the inference schemes, we set the hyperparameters α and γ to 1. In every iteration of DPVI and particle filtering, we do a forward filtering-backward smoothing pass over all the sequences of the dataset. Our results, shown in Fig. 7.6.6(B), demonstrate that DPVI with 5 and 10 particles can converge in fewer iterations compared to other reasonable baselines. We illustrate the lower bound (Eq. (7.7)) convergence in Fig. 7.6.7. The results are computed over 20 runs and for 1, 5 and 10 particles.

It is important to note here that we do not expect DPVI to outperform Gibbs sampling in all scenarios; when computation time is not strongly limited, we expect DPVI and Gibbs to perform similarly. This point applies here as well as to the experiments reported in the sections below. We see DPVI as a useful alternative to Gibbs when the computational budget is low and the required fidelity of the approximation can be satisfied by capturing a few of the posterior modes.

7.6.3 Infinite relational model (IRM)

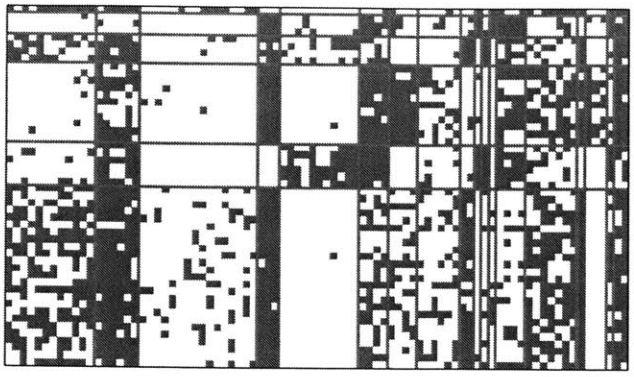
The IRM (Kemp et al., 2006) is a nonparametric model of relational systems. The model simultaneously discovers the clusters of entities and the relationships between the clusters. A key assumption of the model is that each entity belongs to exactly one cluster.

Given a relation R involving J types of entities, the goal is to infer a vector of cluster assignments x^j for all the entities of each type $j = 1, \ldots, J$.[3] Assuming the cluster assignments for each type are independent, the joint density of the relation and the cluster assignment vectors can be written as:

$$P(R, x^1, \ldots, x^J) = P(R|x^1, \ldots, x^J) \prod_{j=1}^{J} P(x^j). \tag{7.16}$$

The cluster assignment vectors are drawn from a CRP(α) prior. Given the cluster assignment vectors, the relations are drawn from a Bernoulli distribution with a parameter η that depends on the clusters involved in that relation. For instance, in a single two-place relation, $\eta(a, b)$ is the probability of having a link between any given pair (i, j) where i is in cluster a and j is in cluster b.

[3]The IRM model can be defined for multiple relations but for simplicity we only describe the single relation case.



Sample animal clusters:
A1: Hippopotamus, Elephant, Rhinoceros
A2: Seal, Walrus, Dolphins, Blue Whale, Killer Whale, Humpback Whale
A3: Beaver, Otter, Polar Bear

Sample feature clusters:
F1: Hooves, Long neck, Horns
F2: Inactive, Slow, Bulbous Body, Tough Skin
F3: Lives in Fields, Lives in Plains, Grazer
F4: Walks, Quadrupedal, Ground
F5: Fast, Agility, Active, Tail

Figure 7.6.8: **Co-clustering of animals and features with an IRM.** Co-clustering of animals (rows) and features (columns) after 50 iterations of DPVI with 10 particles in the infinite relational model.

More formally, let us define an M dimensional relation $R : T^{d_1} \times \ldots T^{d_M} \mapsto \{0, 1\}$, over J different types. Each relational value is generated according to:

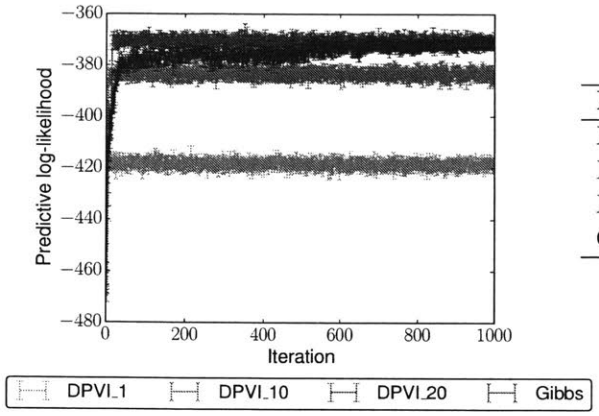
$$R(i_1, \ldots, i_M)|x^1, \ldots, x^J \sim \text{Bern}(\eta(x_{i_1}^{d_1}, \ldots, x_{i_M}^{d_M})), \tag{7.17}$$

where d_m denotes the label of the type (i.e., $d_m \in \{1, \cdots, J\}$) and i_m is the entity occupying position m in the relation. Each entry of parameter matrix η is drawn from a Beta(β, β) distribution. By using a conjugate Beta-Bernoulli model, we can analytically marginalize the parameters η (see Kemp et al., 2006), allowing us to directly compute the likelihood of the relational matrix given the cluster assignments, $P(R|x^1, \ldots, x^J)$.

We compared the performance of DPVI with Gibbs sampling, using predictive log-likelihood on held-out data as a performance metric. The "animals" dataset analyzed by Kemp et al. (2006), was used for this task. This dataset (Osherson et al., 1991) is a two type dataset $R : T_1 \times T_2 \to \{0, 1\}$ with animals and features as it types; it contains 50 animals and 85 features.

We removed 20% of the relations from the dataset and computed the predictive log-likelihood for the held-out data. We ran DPVI with 1, 10 and 20 particles for 1000 iterations. Given the weights of the particles, we computed the weighted log-likelihood. We also ran 20 independent runs of the Gibbs sampler and DPVI for 1000 iterations and computed the average predictive log-likelihood. Every iteration scans all the data points in all the types sequentially. We set the hyperparameters α and β to 1. Fig. 7.6.8 illustrates the co-clustering discovered by DPVI for the dataset, demonstrating intuitively reasonable animal and feature clusters.

The results after 1000 iterations are presented in Fig. 7.6.9B. The best performance is achieved by DPVI with 20 particles. Fig. 7.6.9A shows the predictive log-likelihood for every iteration of



(A)

Method	Predictive LL
DPVI ($K = 1$)	-418.49 ± 4.24
DPVI ($K = 10$)	-382.46 ± 7.34
DPVI ($K = 20$)	$\mathbf{-370.67 \pm 6.23}$
Gibbs (avg. of 20 runs)	$\mathbf{-371.74 \pm 5.07}$

(B)

Figure 7.6.9: **Infinite relational model results for the animals dataset** *(A)* Predictive log-likelihood vs iteration for the animals dataset. The error bars correspond to the standard error and for both methods are computed over 20 runs. *(B)* Predictive log-likelihood after 1000 iterations of DPVI and Gibbs (with 50 burnin iterations) for the animals dataset (with 20 % held-out). DPVI with 20 particles performs in par with the Gibbs sampler.

DPVI and Gibbs sampling. DPVI with 10 and 20 particles converge in 11 and 18 iterations, respectively. In terms of computation time per iteration of DPVI versus Gibbs, the only difference for DPVI with one particle and Gibbs is the sorting cost. Hence, for the multiple particle versus multiple runs of Gibbs sampling, the only additional cost is the sorting cost for multiple particles (e.g. 10 or 20). However, this insignificant additional cost is compensated for by a faster convergence rate in our experiments.

7.7 Conclusions

This chapter introduced a particle-based variational method that applies to a broad class of inference problems in discrete models. We described a practical algorithm for optimizing the particle approximation, and showed empirically that it can outperform widely-used Monte Carlo and variational algorithms. The key to the success of this approach is an intentional selection of particles: rather than generating them randomly (as in Monte Carlo algorithms), we deterministically choose a set of unique particles that optimizes the KL divergence between the approximation and the target distribution.

This approach leads to an interesting view on the problem of resampling in sequential Monte Carlo. Resampling is necessary to remove conditionally unlikely particles, but the resulting loss of particle diversity can lead to degeneracy. As we showed in our experiments, tuning an ESS threshold for resampling can improve performance, but requires finding a relatively narrow sweet spot for the

threshold. DPVI achieves comparable performance to the best particle filter by using a deterministic strategy for deleting and replacing particles and does not require tuning thresholds. Each particle is guaranteed to be unique and have high probability among all states discovered so far.

DPVI also suggests new hybrids of ideas from Monte Carlo and variational inference. Consider models where all particle extensions cannot be enumerated. In this setting, one could randomly choose particle extensions. To use these particles in a Monte Carlo scheme we may need to know the output probability density of the particle extension mechanism. However, if we use the results in DPVI, we just need to be able to score the results under the joint probability distribution. No proposal distribution is needed. This could make it possible to use proposal mechanisms that can be seen to work well empirically but that are difficult to analyze *a priori*.

Although our empirical results are promising, much more empirical and theoretical work is needed to understand the fundamental tradeoffs between variational and Monte Carlo inference. However, the results for DPVI on several problems are promising, and the approach to defining variational approximations may be more broadly applicable. We hope this work encourages others to develop different hybrids of Monte Carlo and variational inference that overcome the limitations of each approach when used in isolation.

CHAPTER 8

Conclusion

In this final chapter of the thesis, we summarize the main contributions of the thesis and also suggest some future research directions. We proposed models for analyzing user behavior in software applications. We focused on three tasks which are common in user modeling: 1) discovering common usage patterns across different users, 2) identifying different user groups based on their usage patterns, and 3) personalization and user-dependent recommendation. We cover an overview of our models for these tasks and also various alternative approaches in Chapter 1. A more detailed description of these models is provided in Chapters 3 to 6. Specifically, we covered latent variable models on task 1 in Chapters 3 to 5 and the models on the remaining tasks in Chapter 6. For posterior inference in these models, we rely heavily on variational inference as closed-form conditional posterior distributions are not available for the models. In Chapter 7, we tackle a major limitation with variational inference in discrete latent variable models. In particular, for complex discrete latent variable models, the bias induced by variational approximations can sometimes lead to poor predictive performance. We proposed an inference framework which combines the advantages of both Monte Carlo methods and variational inference. In what follows, we summarize our main contributions and suggest some future research directions.

Chapter 3: Segmenting user behavior traces with an efficient Bayesian nonparametric approach

Main contributions In Chapter 3, we introduce siHMM a Bayesian nonparametric with efficient stochastic variational inference. The model is an extension of widely used iHMM and in contrast to iHMM, it can capture two levels of dynamics implicitly. Compared to hierarchical HMM (HHMM) models which are used for modeling multiple levels of dynamics, it has a simpler and more efficient inference scheme. We use the model to segment user behavior traces and identify the points where

the user switches from one task to another task. We show that the model empirically outperforms iHMM and performs on par with the more complex HHMM models. We also demonstrate that the model is not only limited to user modeling; it can be applied to other domains such as activity recognition and segmenting animal behavior.

Possible research directions Potentially, one can use the state-indepnedent transition vector learned from an siHMM for summarizing a sequence. This summary vector can be utilized for user profiling and identifying user groups which is essentially second task from the three tasks that we focus on for user modeling. Another possible research direction is extending siHMM to more than two levels of dynamics which is a challenging problem. This requires keeping track of the states within each segment. Consequently, we will lose the simple and efficient inference scheme that we have for two-level dynamics.

Chapter 4: Markov jump processes for modeling user behavior traces

Main contributions In Chapter 4 we apply a continuous-time model, MJP, to the task of segmenting user behavior traces where we may have observations at unequal time intervals. MJPs are more realistic for this task compared to their discrete-time counterparts since they allow for transitions between states at any time point between observations. However, this more realistic model comes with a price of more complicated and less scalable inference. Available inference algorithms based on particle MCMC scale poorly and mix slowly. Optimization-based methods such as EM are inapplicable if the state size is countably infinite (*e.g.*, Bayesian nonparametric MJPs). Finally, finding the most probable latent trajectory using maximum likelihood approaches will result in degenerate solutions. We introduce JUMP-means algorithm, a small variance asymptotic approach to estimating the most probable trajectories in MJPs which does not suffer from the aforementioned issues.

Possible research directions MJPs are not directly applicable to dataset with multiple levels of dynamics. We argued in Chapter 3 that this may be a reasonable assumption for user traces where there could be some structure within each segment (user task). Hence, a hierarchical extension of MJPs are more suitable for the user behavior segmentation. However, inference in hierarchical MJPs is even more challenging compared to HHMMs since we also need to keep track of the dwell times in different states at multiple levels of the model. A possible direction for extending MJPs to two level hierarchies could be an approach similar to siHMM where we implicitly model the hierarchical structure.

Chapter 5: Topic modeling applied to user behavior traces with spherical HDP

Main contributions Up to this chapter, we assumed a sequential structure for the user trace data. In Chapter 5, we relax this assumption and treat each user trace as a *bag of actions*. We can apply topic models for inferring the topics (*i.e.,* user tasks in our application) to the bag of actions representation; however, standard topic models ignore the semantic regularities in the action space. Semantic regularities in this space can encode the similarities between different actions. Distributional representation of words (*e.g., word2vec*) are able to capture these regularities in the natural language but they have not been explored extensively in the context of topic modeling. For the user action space, Adar et al. (2014) introduce *CommandSpace*, the distributional representation in the action space that has been trained on a large dataset of user actions in Adobe photoshop. In this chapter, we introduce sHDP, an HDP model with vMF observations which can incorporate the semantic structure of the actions. Our model, performs significantly better than the standard LDA and also Gaussian LDA (Das et al., 2015) baselines in terms of topic coherence for a user trace data and also two English text corpora.

Possible research directions Our model uses the pre-trained action and word embeddings and does not allow for joint learning of the embeddings and the topics. Joint training of these components may result in improved performance; that is a hypothesis worth exploring. However, inference in such models is challenging as we have a more complex observation model. Frameworks such as structured VAEs (Johnson et al., 2016), which are capable of combining graphical models and neural network observation models, may be applicable to this problem.

Chapter 6: Multimodal prediction and personalization in software applications with deep generative models

Main contributions We focus on the tasks of identifying different user groups and personalization in Chapter 6. Given a dataset of image and edit pairs for users, our goal is to develop a model for predicting the edit for a new image. Available prediction models are limited in that they only propose a single prediction or are not readily personalized. Multimodal predictions are important in cases where, given an input from the user, there could be multiple possible suggestions from the application. In photo editing/enhancement, a user might want to apply different kinds of edits to the same photo depending on the effect he or she wants to achieve. A model should therefore be able to recommend multiple enhancements for different user groups and styles. In this chapter, we introduce CGM-VAE, a framework for multimodal prediction that can be extended for personalization tasks. CGM-VAE can identify diverse set of user groups and propose edits from different styles. The personalized variant of the model can recommend edits based on the history of user's edits. We

demonstrate on three datasets that our model can outperform several reasonable baselines in terms of the predictive log-likelihood and also error in the LAB space.

Possible research directions CGM-VAE model assumes each user belongs to a single mixture component. A more realistic assumption is that each image-edit pair for each user can belong to a different mixture component. This is reminiscent of admixture models and requires a more complicated inference scheme. Although in theory structured VAE can work, the flexibility of the model at both the latent structure and the observation model may make the inference even more challenging.

Chapter 7: Improving variational inference for latent variable models: discrete particle variational inference

Main contributions Inference in our proposed latent variable models in Chapters 3 to 6 is based on variational inference which in practice has been shown to be more scalable compared to Monte Carlo inference approaches. However, without flexible variational distributions, we may have poor posterior approximations. For instance, a poor variational approximation may cover only a single mode in a multimodal posterior distribution. In Chapter 7, we introduce a new approximate inference method, called DPVI, that aims to combine key strengths of both Monte Carlo and variational inference. In DPVI, similar to "particle approximation" output by Monte Carlo methods, we use a weighted collection of samples as the approximating family for variational inference. We distribute the samples in such a way that they cover high probability regions of the target distribution, but without the samples all devolving onto the mode of the distribution. We compare the performance of this new inference method with several baselines on sequential and non-sequential latent variable models and show it has superior time/accuracy trade-offs compared to these alternatives.

Possible research directions The major limitation of DPVI is that it is restricted to discrete latent variable models. Extending it to models with continuous latent variables can be done by combining mean-field (for continuous variables) and DPVI. However, this approach is not a principled way to make the DPVI work on models with mixed variables; proposing an algorithm that can cover both types of variables in a unified way can be a challenging task. Extending DPVI type methods for mixed latent variable models allows us to apply them to models such as sHDP or siHMM.

APPENDIX A

Bayesian Nonparametric MJPs for SVA

We show that the ΓΓP retains the key properties of the ΓEP (Saeedi and Bouchard-Côté, 2011): conjugacy and exchangeability. Let $T_i \triangleq \sum_{j=1}^{k} \mathbb{1}[z_{j-1} = i]t_j$ and $F_i \triangleq \sum_{j=1}^{k} \mathbb{1}[z_{j-1} = i]\delta_{z_j}$ be the sufficient statistics of the observations.

Proposition A.0.1. *The* ΓΓP *is a conjugate family:* $\mu_i \,|\, \mathcal{U}_k \sim GamP(\beta\mu_i', \gamma_i')$, *where* $\mu_i' = \mu_0 + F_i$ *and* $\gamma_i' = \gamma + T_i$.

Proof sketch. The proof is analogous to that for Proposition 2 in (Saeedi and Bouchard-Côté, 2011). The key additional insight is that $X \sim \mathsf{Gam}(\beta a, b)$ and $Y \mid X \sim \mathsf{Gam}(\beta, X)$ are conjugate: $X \mid Y \sim \mathsf{Gam}(\beta(a+1), b+Y)$. □

In order to give the joint distribution of the times $\mathcal{T} \triangleq \mathcal{T}_K \triangleq (t_1, \ldots, t_K)$, we first derive the predictive distribution for the ΓΓP, $(z_{k+1}, t_{k+1}) \,|\, \mathcal{U}_k$. We make use of the following family of densities.

Definition A.0.2 (Shaped Translated Pareto)**.** Let $\beta > 0, \alpha > 0, \gamma > 0$. A random variable S is *shaped translated Pareto*, denoted $S \sim \mathsf{STP}(\beta, \alpha, \gamma)$, if it has density

$$f(t) = \frac{\gamma^{\alpha\beta}}{B(\beta, \alpha\beta)} \frac{t^{\beta-1}}{(t+\gamma)^{(1+\alpha)\beta}},$$

where $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ is the beta function.

Proposition A.0.3. *The predictive distribution of the* ΓΓP *is*

$$(z_{k+1}, t_{k+1}) \,|\, \mathcal{U}_k \sim \bar{\mu}_{z_k}' \times \mathsf{STP}(\beta, \|\bar{\mu}_{z_k}'\|, \gamma_{z_k}'). \tag{A.1}$$

Proof. By Proposition A.0.1, it suffices to show that if $\mu \sim \Gamma\mathrm{P}(\beta\mu_0, \gamma)$, $s \mid \mu \sim \bar{\mu}$, and $t \mid \mu \sim \mathsf{Gam}(\beta, \|\mu\|)$, then $(s,t) \sim \bar{\mu} \times \mathsf{STP}(\beta, \kappa_0, \gamma)$, where $\kappa_0 \triangleq \|\mu_0\|$. Letting $x = \|\mu\|$, the distribution of t is

$$p(t) = \int_0^\infty p(t \mid x)p(x)\mathrm{d}x = \int_0^\infty \frac{x^\beta t^{\beta-1}e^{-xt}}{\Gamma(\beta)} \frac{\gamma^{\beta\kappa_0}x^{\beta\kappa_0-1}e^{-\gamma x}}{\Gamma(\beta\kappa_0)}\mathrm{d}x$$
$$= \frac{\gamma^{\beta\kappa_0}t^{\beta-1}}{\Gamma(\beta)\Gamma(\beta\kappa_0)} \int_0^\infty x^{\beta(1+\kappa_0)-1}e^{-(\gamma+t)x}\mathrm{d}x = \frac{\gamma^{\beta\kappa_0}t^{\beta-1}}{\Gamma(\beta)\Gamma(\beta\kappa_0)} \frac{\Gamma(\beta(1+\kappa_0))}{(\gamma+t)^{\beta(1+\kappa_0)}}.$$

□

We can now show that the process is exchangeable by exhibiting the joint distribution of waiting times:

Proposition A.0.4. *Let $\bar{t}^*_m = (t^*_{m1}, \ldots, t^*_{mK_m})$ be the waiting times following state m. Then $\bar{t}^*_m$ is an exchangeable sequence with joint distribution*

$$p(\bar{t}^*_m) = \frac{\Gamma(\beta(\kappa_0 + K_m))}{\Gamma(\beta)^{K_m}} \frac{(\prod_{j=1}^{K_m} t^*_{mj})^{\beta-1}}{(\gamma + \sum_{j=1}^{K_m} \tau_{mj})^{\beta(\kappa_0+K_m)}} \tag{A.2}$$

Proof sketch. Take the product of the predictive distributions of $\tau_{m1}, \ldots, \tau_{mK_m}$. □

APPENDIX B

Sample results for multimodal prediction in software applications with deep generative models

B.1 Details of experiments

Hyperparameter settings For training all the models, we use two possible learning rates 0.001 and 0.0001. For the MDN, MLP and CGM-VAE, we use 4 hidden layers with the same number of hidden nodes (500 or 1000) in all the layers. For LBN, we use two deterministic hidden layers with linear activation function same as Dauphin and Grangier (2015) and 500 or 1000 nodes; we also use two stochastic layers with the same number of nodes with sigmoid activation functions following Dauphin and Grangier (2015). We try two possible minibatch sizes of 100 and 200. For the models which need the number of mixture components (*i.e.,* CGM-VAE and MDN), we select this number from the set $\{1, 3, 5, 10\}$. Finally, for the CGM-VAE model, we choose the dimension of the latent variable from $\{2, 20\}$. We choose the best hyperparameter setting based on the variational lower bound of the held-out dataset.

Baselines We choose a set of reasonable baselines that can cover related models in both domains of multimodal prediction and automatic photo enhancement. As mentioned in Section 2, literature on the automatic photo enhancement can be divided into two main categories of models: 1) parametric methods which typically minimize an MSE loss: MLP and MDN baselines capture these methods, and 2) nonparametric methods that are not reasonable baselines for us since their proposed edits are destructive (*e.g.,* Lee et al., 2015) or do not benefit from other users' information (*e.g.,* Koyama et al., 2016). We also add LBN as a strong baseline since it has been shown that it can outperform the MDN and other standard multimodal prediction baselines (Dauphin and Grangier, 2015) .

Splitting the datasets We split each dataset into random subsets of train, validation and test in a way that for all three datasets (*i.e.,* casual, frequent and expert users) we have subsets with reasonable sizes. Larger training sets may result in non-representative validation and test sets for our small dataset (*i.e.,* expert users), and larger test and validation sets may result in non-representative training set for the same dataset. The ratio that we used is just one way for having a reasonable size subsets; however, we showed for these random subsets and across all three datasets and for three different evaluation metrics our approach outperforms all the other baselines significantly.

B.2 Sample edits from CGM-VAE and sample user categorization from P-VAE

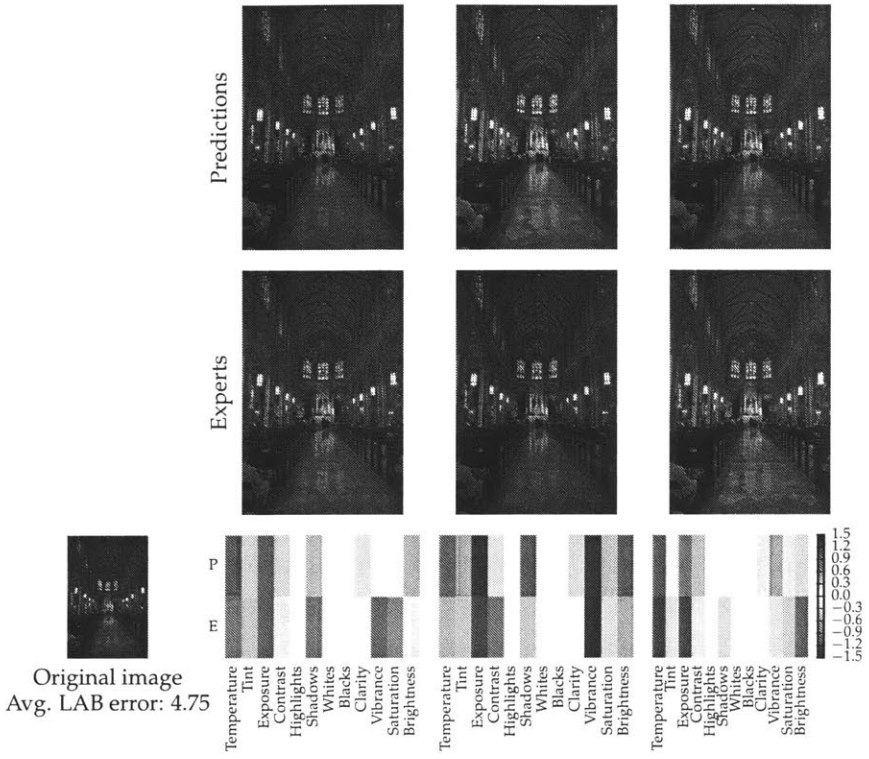


Figure B.2.1: Image 4876 from Adobe-MIT5k dataset

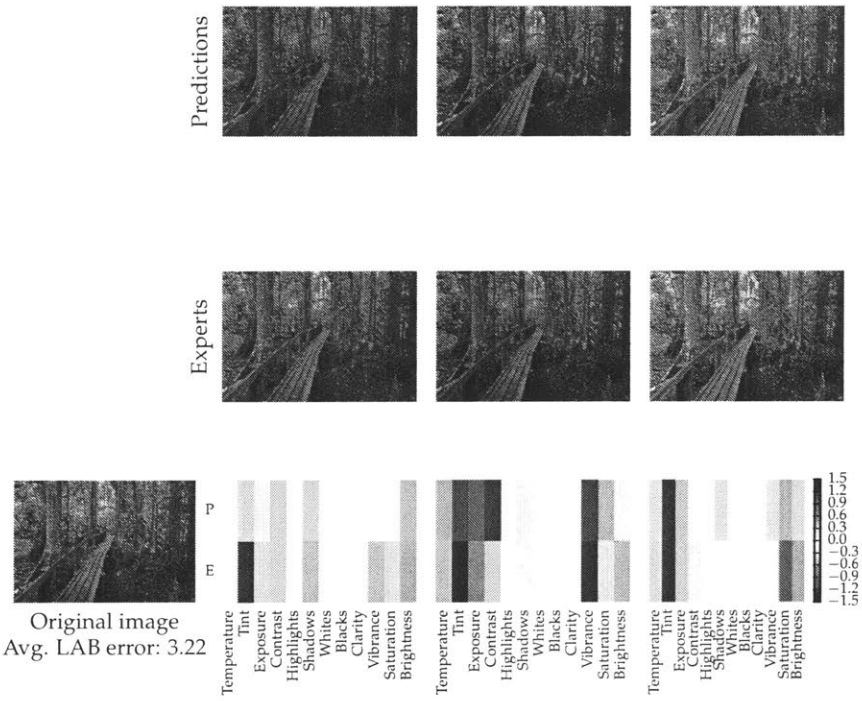


Figure B.2.2: Image 4855 from Adobe-MIT5k dataset

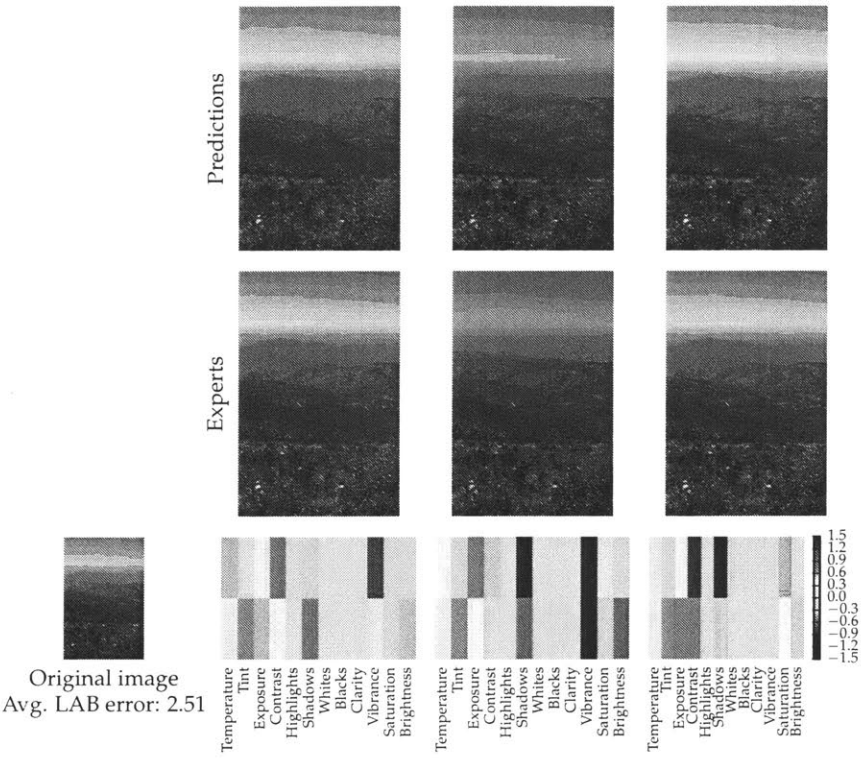


Figure B.2.3: Image 4889 from Adobe-MIT5k dataset

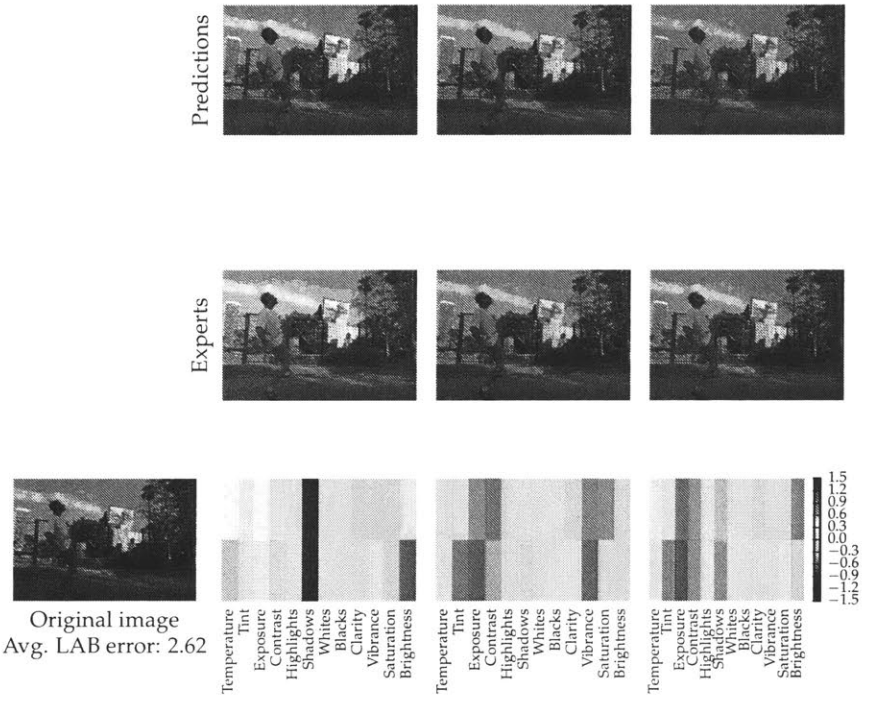


Figure B.2.4: Image 4910 from Adobe-MIT5k dataset

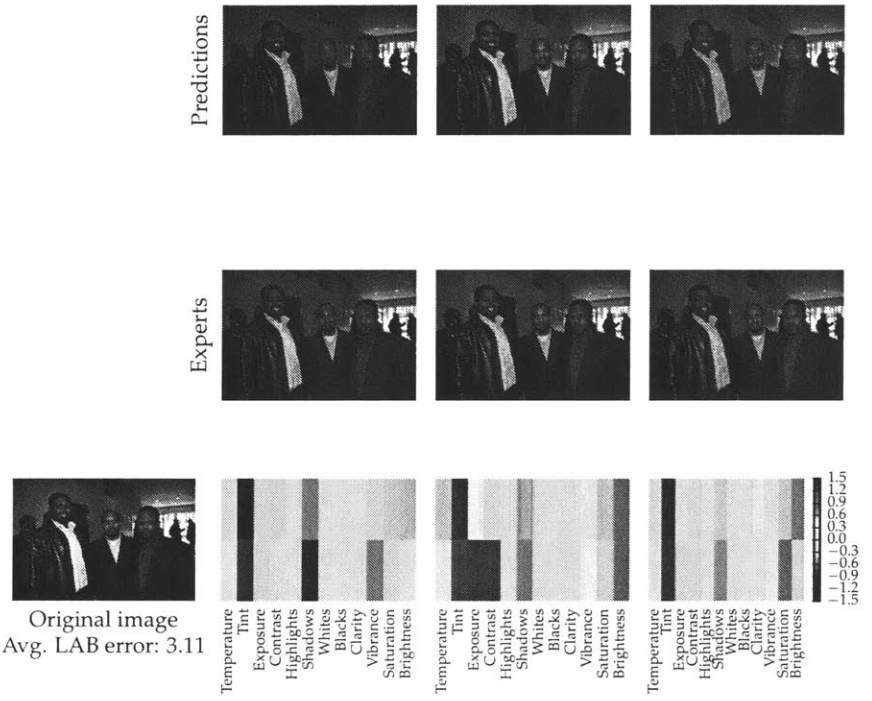


Figure B.2.5: Image 4902 from Adobe-MIT5k dataset

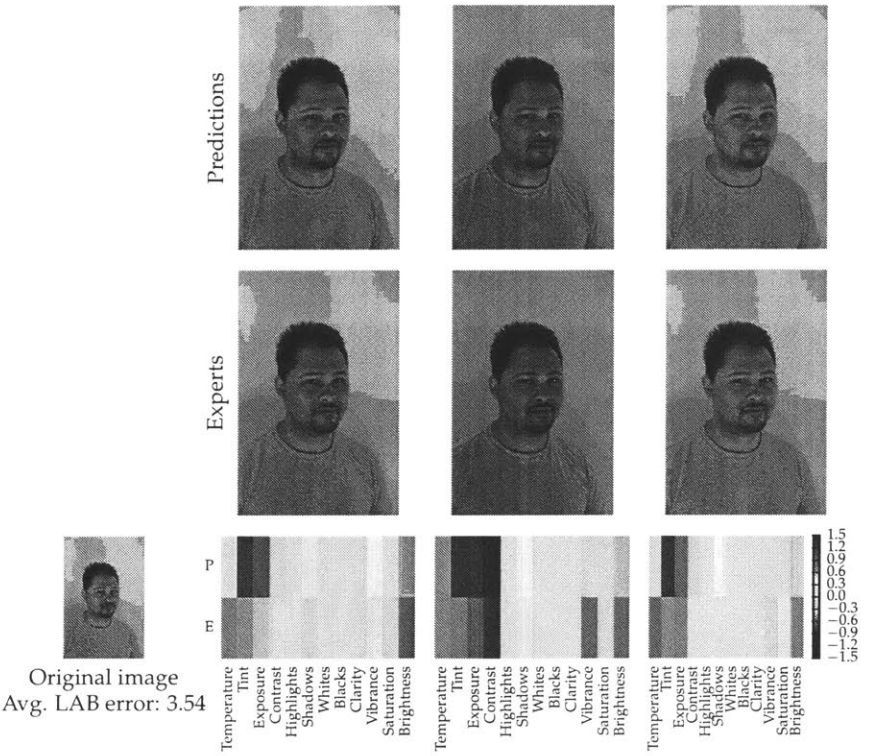


Figure B.2.6: Image 4873 from Adobe-MIT5k dataset

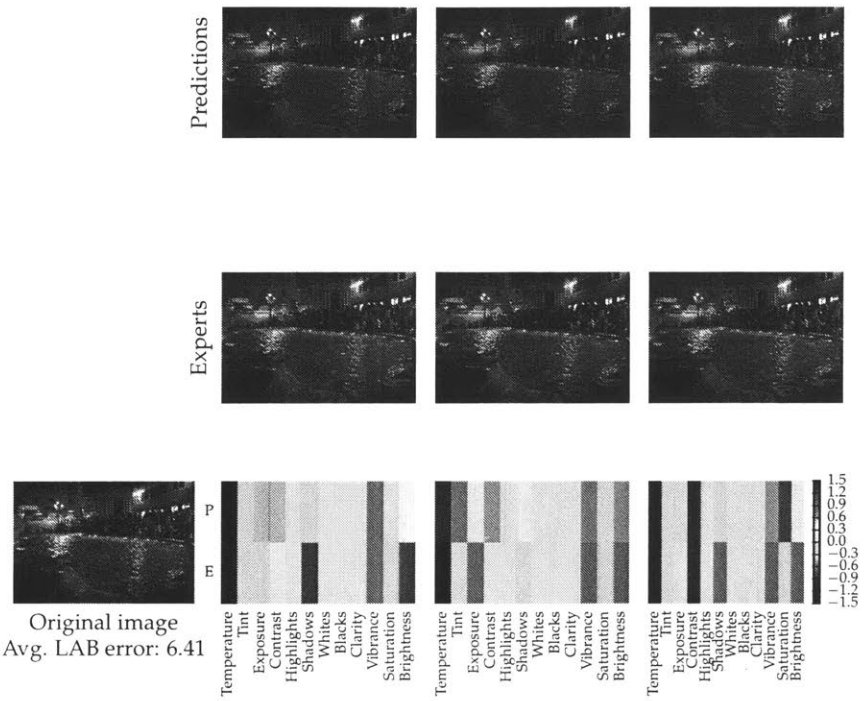


Figure B.2.7: Image 4882 from Adobe-MIT5k dataset

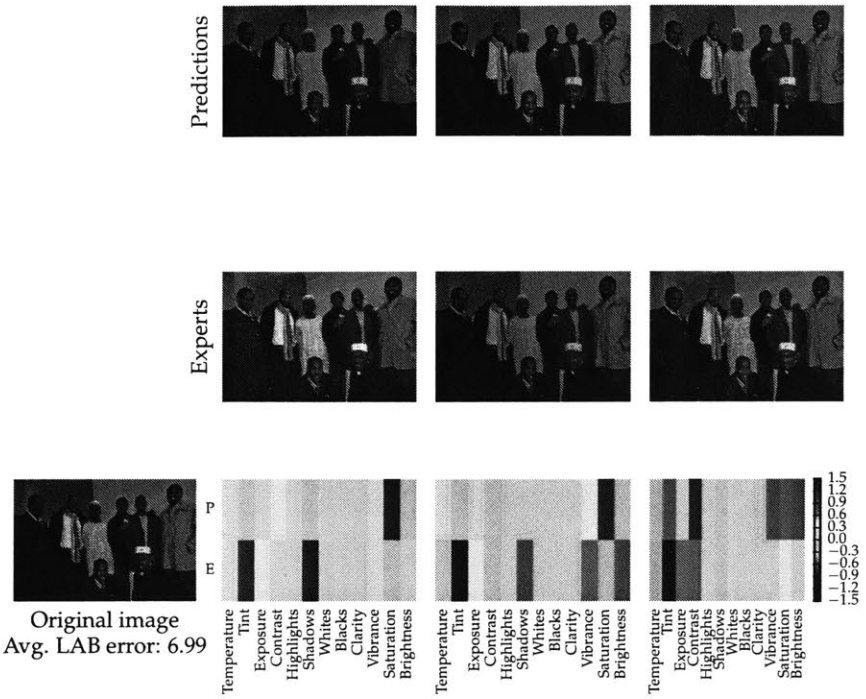


Figure B.2.8: Image 4872 from Adobe-MIT5k dataset

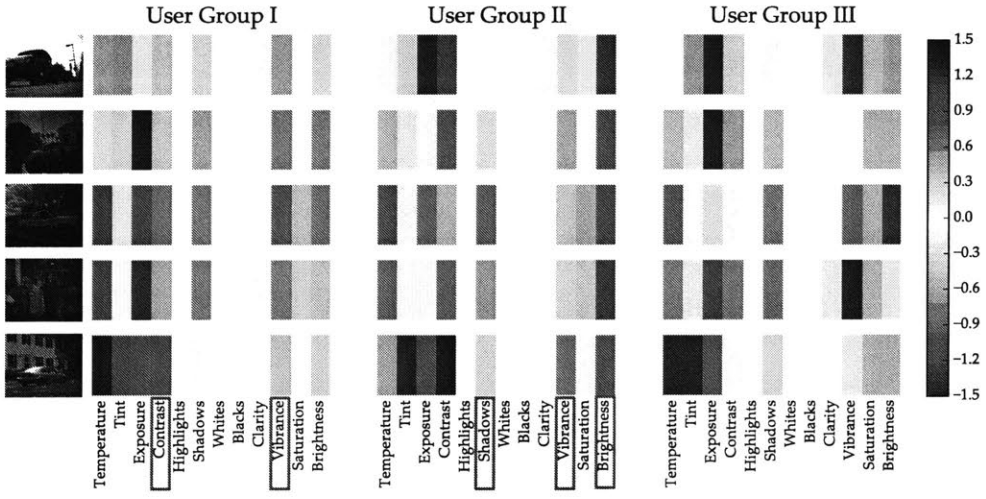


Figure B.2.9: A sample user categorization from the P-VAE model (car image).

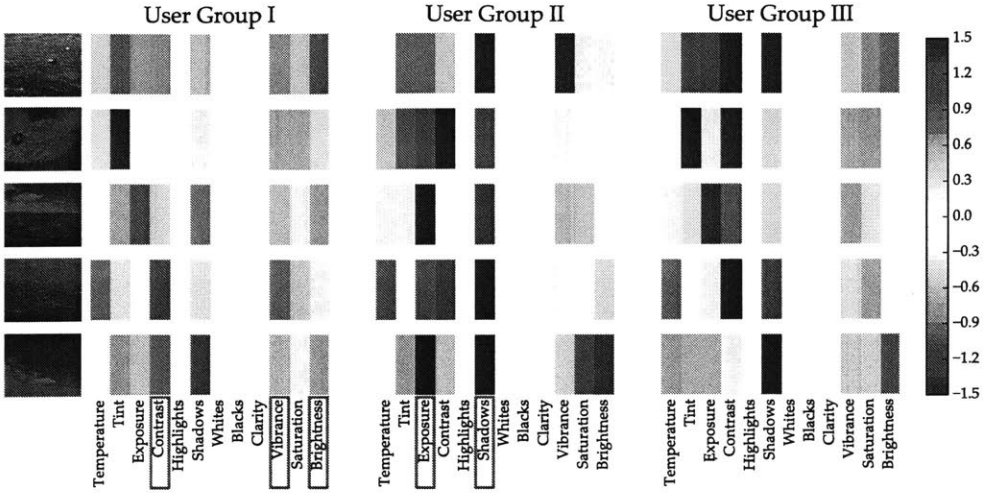


Figure B.2.10: A sample user categorization from the P-VAE model (similar photos with dominant blue colors).

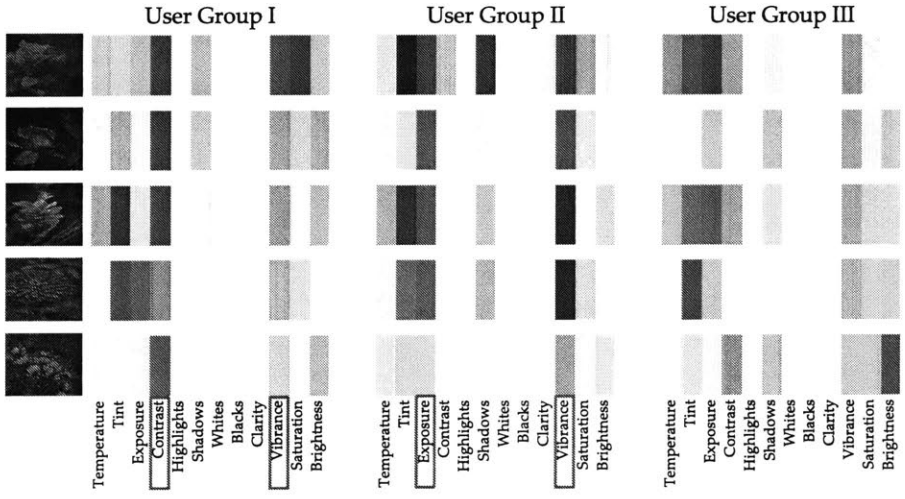


Figure B.2.11: A sample user categorization from the P-VAE model (flowers).

APPENDIX C

More experiments on discrete particle variational inference (DPVI)

C.0.1 Dirichlet process mixture model

In this section, we demonstrate the performance of DPVI on DPMM introduced in Section 2.2.1[1].

Synthetic data

We first demonstrate our approach on synthetic datasets drawn from various mixtures of bivariate Gaussians (see Table C.1). The model parameters for each simulated dataset were chosen to create a spectrum of increasingly overlapping clusters. In particular, we constructed models out of the following building blocks:

$$\mu_1 = \begin{pmatrix} 0.0, & 0.0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 0.5, & 0.5 \end{pmatrix}$$
$$\Sigma_1 = \begin{pmatrix} 0.25, & 0.0 \\ 0.0, & 0.25 \end{pmatrix}, \qquad \Sigma_2 = \begin{pmatrix} 0.5, & 0.0 \\ 0.0, & 0.5 \end{pmatrix}.$$

For the DPMM, we used a Normal likelihood with a Normal-Inverse-Gamma prior on the component parameters:

$$y_{nd}|x_n = k \sim \mathcal{N}(m_{kd}, \sigma_{kd}^2), \qquad m_{kd} \sim \mathcal{N}(0, \sigma_{kd}^2/\tau), \qquad \sigma_{kd}^2 \sim \text{IG}(a, b), \tag{C.1}$$

where $d \in \{1, 2\}$ indexes observation dimensions and $\text{IG}(a, b)$ denotes the Inverse Gamma distribution with shape a and scale b. We used the following hyperparameter values: $\tau = 25, a = 1, b = 1, \alpha = 0.5$.

[1]These experiments are done by Tejas Kulkarni a coauthor in Saeedi et al. (2017b).

Dataset	PF ($K = 20$)	MF	DPVI ($K = 1$)	DPVI ($K = 20$)
D1: $[\mu_1, 4\mu_2, 8\mu_2], \Sigma_1$	0.97±0.03	0.80±0.10	0.93±0.05	0.99±0.02
D2: $[\mu_1, 4\mu_2, 8\mu_2], \Sigma_2$	0.89±0.05	0.63±0.02	0.86±0.07	0.90±0.03
D3: $[\mu_1, 2\mu_2, 4\mu_2], \Sigma_1$	0.58±0.12	0.53±0.05	0.51±0.03	0.74±0.16
D4: $[\mu_1, 2\mu_2, 4\mu_2], \Sigma_2$	0.50±0.06	0.29±0.05	0.46±0.05	0.55±0.07
D5: $[\mu_1, \mu_2, 2\mu_2], \Sigma_1$	0.05±0.05	0.28±0.12	0.014±0.02	0.14±0.10
D6: $[\mu_1, \mu_2, 2\mu_2], \Sigma_2$	0.15±0.08	0.12±0.03	0.11±0.06	0.19±0.07

Table C.1: **Clustering accuracy (V-Measure) for DPMM.** Each dataset consisted of 200 points drawn from a mixture of 3 Gaussians. For each dataset, we repeated the experiment 150 times by iterating through random seeds, reporting mean and standard error. The left column shows the ground truth mean for each cluster and the covariance matrix (shared across clusters). PF denotes particle filtering and MF denotes mean-field.

Number of particles	**DPVI**	**Particle Filtering**
$K = 10$	15.20s	14.71s
$K = 50$	153.75s	184.17s
$K = 100$	567.84s	699.43s

Table C.2: **Run time comparison for DPMM with synthetic data** using dataset from Table 1. The run time of DPVI is slightly better than particle filtering for a single pass through the dataset.

Clustering accuracy was measured quantitatively using V-measure (Rosenberg and Hirschberg, 2007). V-measure is an entropy-based measure which explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied. Fig. C.0.1 graphically demonstrates the discovery of latent clusters for both DPVI as well as particle filtering. As shown in Table C.1, we observe only marginal improvements when the means are farthest from each other and variances are small, as these parameters leads to well-separated clusters in the training set. However, the relative accuracy of DPVI increases considerably when the clusters are overlapping, either due to the fact that the means are close to each other or the variances are high. One factor contributing to greater performance of DPVI might be the diversity term in the variational formulation.

An interesting special case is when $K = 1$. In this case, DPVI is equivalent to the greedy algorithm proposed by Daume (2007) and later extended by Wang and Dunson (2011). In fact, this algorithm was independently proposed in cognitive psychology by Anderson (1991). As shown in Table C.1, DPVI with 20 particles outperforms the greedy algorithm, as well as particle filtering with 20 particles. We also demonstrate the run-time performance of DPVI compared to particle filtering in Table C.2. It can be seen in our experiments that DPVI tends to be comparable to particle filtering or sometimes more efficient in terms of run-time for the same task, although the theoretical complexity is the same.

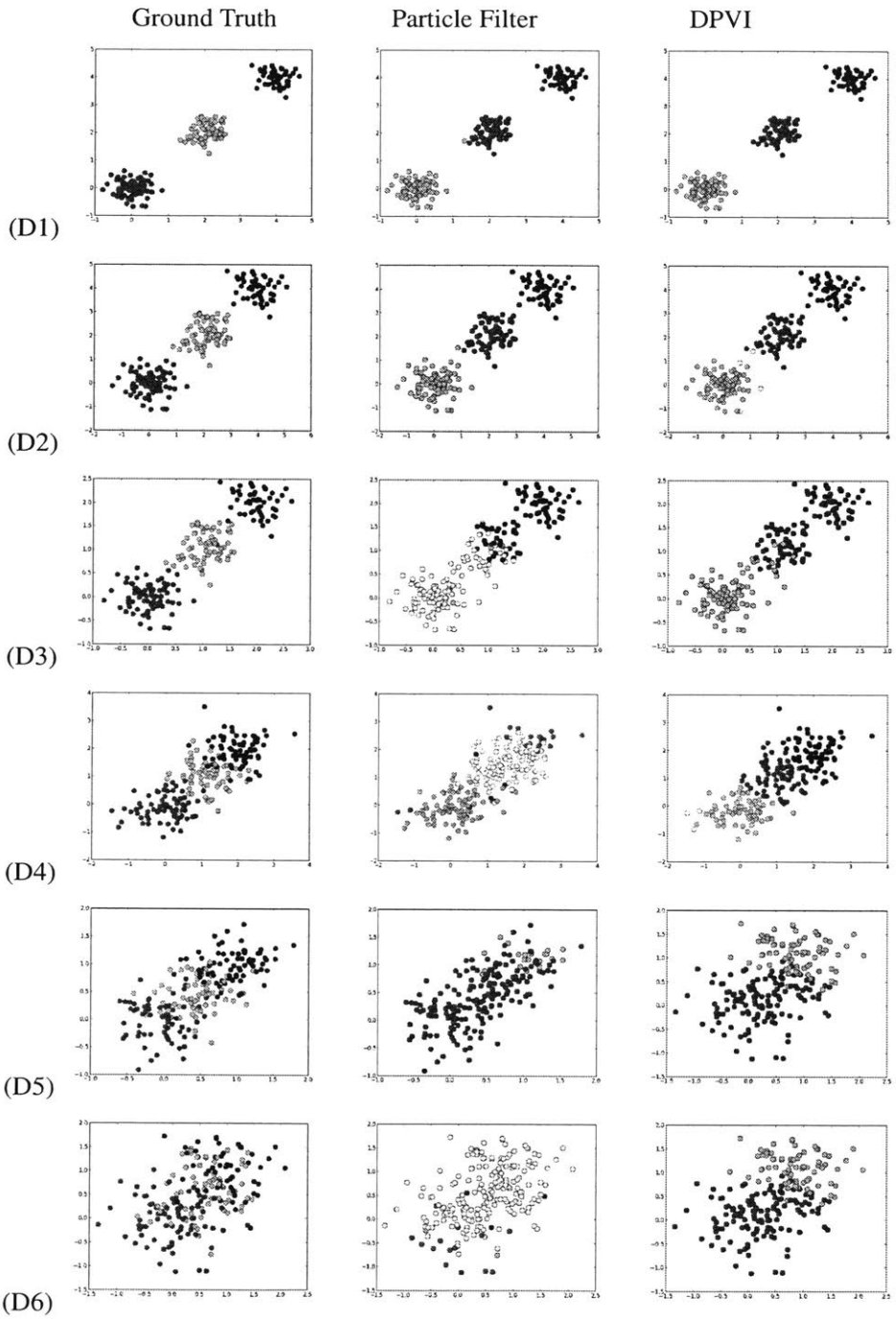


Figure C.0.1: **DPMM clustering of synthetic datasets**. We treat DPMM as a filtering problem, analyzing one randomly chosen data point at a time. Colors indicate cluster assignments. Each row corresponds to one synthetic dataset; refer to Table 1 for corresponding quantitative results. Column 1: Ground truth; Column 2: particle filtering; Column 3: DPVI. The DPVI filter scales similarly to the particle filter but does not underfit as severely.

Spike sorting

Spike sorting is an important problem in experimental neuroscience settings where researchers collect large amounts of electrophysiological data from multi-channel tetrodes. The goal is to extract from noisy spike recordings attributes such as the number of neurons, and cluster spikes belonging to the same neuron. This problem naturally motivates the use of DPMM, since the number of neurons recorded by a single tetrode is unknown. Previously, Wood and Black (2008) applied the DPMM to spike sorting using particle filtering and Gibbs sampling. Here we show that DPVI can outperform particle filtering, achieving high accuracy even with a small number of particles.

We used data collected from a multiunit recording from a human epileptic patient (Quiroga et al., 2004). The raw spike recordings were preprocessed following the procedure proposed by Quiroga et al. (2004), though we note that our inference algorithm is agnostic to the choice of preprocessing. The original data consist of an input vector with $D = 10$ dimensions and 9196 data points. Following Wood and Black (2008), we used a Normal likelihood with a Normal-Inverse-Wishart prior on the component parameters:

$$\mathbf{y}_n | x_n = k \sim \mathcal{N}(\mathbf{m}_k, \Lambda_k), \qquad \mathbf{m}_k \sim \mathcal{N}(0, \Lambda_k/\tau), \qquad \Lambda_k \sim \mathrm{IW}(\Lambda_0, \nu), \tag{C.2}$$

where $\mathrm{IW}(\Lambda_0, \nu)$ denotes the Inverse Wishart distribution with degrees of freedom ν and scale matrix Λ_0. We used the following hyperparameter values: $\nu = D + 1, \Lambda_0 = \mathbf{I}, \tau = 0.01, \alpha = 0.1$.

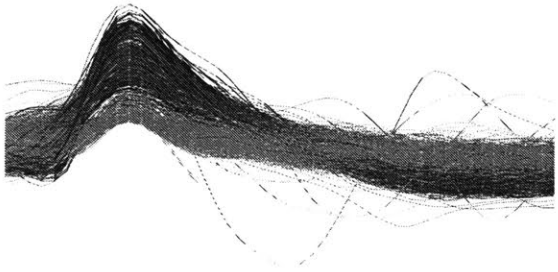
We compared our algorithm to the current best particle filtering baseline, which uses stratified resampling (Wood and Black, 2008; Fearnhead, 2004). The same model parameters were used for all comparisons. Qualitative results, shown in Fig. C.0.2B, demonstrate that DPVI is better able to separate the spike waveforms into distinct clusters, despite running DPVI with 10 particles and particle filtering with 100 particles. We also provide quantitative results by calculating the held-out log-likelihood on an independent test set of spike waveforms. The quantitative results (summarized in Fig. C.0.2C) demonstrate that even with only 10 particles DPVI can outperform particle filtering with 1000 particles.

C.0.2 Ising model

So far, we have been studying inference in directed graphical models, but DPVI can also be applied to undirected graphical models. We illustrate this using the Ising model[2] for binary vectors $x \in \{-1, +1\}^N$:

$$f(x) = \exp\left\{\frac{1}{2} x W x^\top + \theta x^\top\right\}, \tag{C.3}$$

[2]These experiments are done by Sam Gershman a coauthor in Saeedi et al. (2017b).



(A)

(B)

Method	Predictive LL
DPVI ($K = 10$)	-3.2474$\times 10^5$ ($\hat{C} = 3$)
DPVI ($K = 100$)	**-1.3888$\times 10^5$ ($\hat{\mathbf{C}} = \mathbf{3}$)**
PF ($K = 10$)	-1.4771$\pm$0.21 $\times 10^6$ ($\hat{C} = 37$)
PF ($K = 100$)	-5.6757$\pm$1.14 $\times 10^5$ ($\hat{C} = 13$)
PF ($K = 1000$)	-3.2965$\times 10^5$ ($\hat{C} = 5$)

(C)

Method	Run time
DPVI ($K = 10$)	36.20s
DPVI ($K = 50$)	144.6s
DPVI ($K = 100$)	313.8s
PF ($K = 10$)	124s
PF ($K = 50$)	334.2s
PF ($K = 100$)	454.2s

(D)

Figure C.0.2: **Spike Sorting using the DPMM**. Each line is an individual spike waveform, colored according to the inferred cluster. (*A*) Result using particle filtering with 100 particles and stratified resampling as reported by Wood and Black (2008). (*B*) Result using DPVI. The same model parameters were used for both particle filtering and DPVI. (*C*) Spike sorting predictive log-likelihood scores for 200 test points. The best performance is achieved by DPVI with 100 particles. Shown in parentheses is the *maximum a posteriori* number of clusters, $\hat{C}$. (*D*) Run time comparison for DPMM obtained by using the spike sorting dataset. The run time of DPVI is slightly better than particle filtering.

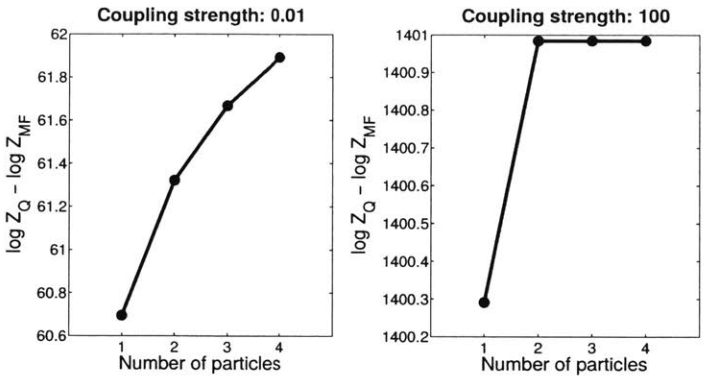


Figure C.0.3: **Ising model results**. Difference between DPVI and mean-field lower bounds on the partition function. Positive values indicates superior DPVI performance. (*A*) Low coupling strength; (*B*) high coupling strength.

where $W \in \mathbb{R}^{N \times N}$ and $\theta \in \mathbb{R}^N$ are fixed parameters. In particular, we study a square lattice ferromagnet, where $W_{ij} = \beta$ for neighboring nodes (0 otherwise) and $\theta_i = 0$ for all nodes. We refer to β as the *coupling strength*. This model has two global modes: when all the nodes are set to 1, and when all the nodes are set to 0. As the coupling strength increases, the probability mass becomes increasingly concentrated at the two modes.

We applied DPVI to this model, varying the number of particles and the coupling strength. At each iteration, we evaluated the change in log probability that would result from setting $\tilde{x}_n^k = 1$:

$$a_n^k = \sum_{n' \in c_n} W_{n,n'} x_{n'}^k + \theta_n, \tag{C.4}$$

and likewise the change for setting $\tilde{x}_n^k = 1$ can be computing by simply flipping the sign of a_n^k. Ordering these changes, we then took the top K to determine the new particle set.

To quantify performance, we computed the DPVI variational lower bound on the partition function and compared this to the lower bound furnished by the mean-field approximation (see Wainwright and Jordan, 2008). Fig. C.0.3A shows the results of this analysis for low coupling strength ($\beta = 0.01$) and high coupling strength ($\beta = 100$). DPVI consistently achieves a better lower bound than mean-field, even with a single particle, and this advantage is especially conspicuous for high coupling strength. Adding more particles improves the results, but more than 3 particles does not appear to confer any additional improvement for high coupling strength. These results illustrate how DPVI is able to capture multimodal target distributions, where mean-field approximations break down (since they cannot effectively handle multimodality).

To illustrate the performance of DPVI further, we compared several posterior approximations for the Ising model in Fig. C.0.4. In addition to the mean-field approximation, we also compared DPVI

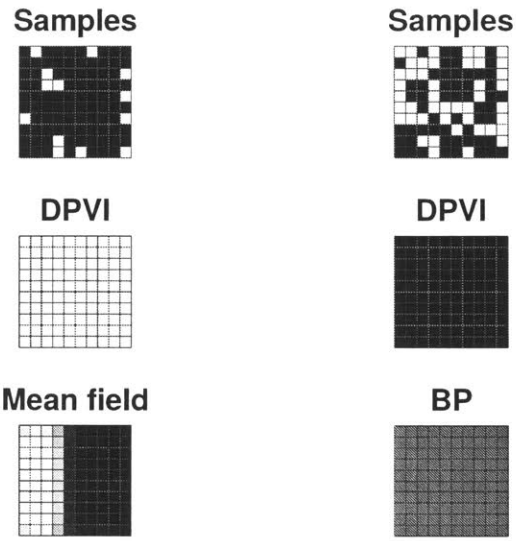


Figure C.0.4: **Ising model simulations**. Examples of posteriors for the ferromagnetic lattice at low coupling strength. (*Top*) Two configurations from a Swendsen-Wang sampler. (*Middle*) Two DPVI particles. (*Bottom left*) Mean-field expected value. (*Bottom right*) Loopy belief propagation expected value.

with two other standard approximations: the Swendsen-Wang Monte Carlo sampler (Swendsen and Wang, 1987) and loopy belief propagation (Murphy et al., 1999). The sampler tended to produce noisy results, whereas mean-field and BP both failed to capture the multimodal structure of the posterior. In contrast, DPVI with two particles perfectly captured the two modes.

Bibliography

Ardavan Saeedi, Matthew Hoffman, Matthew Johnson, and Ryan Adams. The segmented ihmm: a simple, efficient hierarchical infinite hmm. In *International Conference on Machine Learning*, pages 2682–2691, 2016.

Jonathan Huggins, Karthik Narasimhan, Ardavan Saeedi, and Vikash Mansinghka. Jump-means: Small-variance asymptotics for markov jump processes. In *International Conference on Machine Learning*, pages 693–701, 2015.

Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. Nonparametric spherical topic modeling with word embeddings. *arXiv preprint arXiv:1604.00126*, 2016.

Ardavan Saeedi, Matthew D Hoffman, Stephen J DiVerdi, Asma Ghandeharioun, Matthew J Johnson, and Ryan P Adams. Multimodal prediction and personalization of photo edits with deep generative models. *arXiv preprint arXiv:1704.04997*, 2017a.

Ardavan Saeedi, Tejas D. Kulkarni, Vikash K. Mansinghka, and Samuel J. Gershman. Variational particle approximations. *Journal of Machine Learning Research*, 18(69):1–29, 2017b. URL `http://jmlr.org/papers/v18/15-615.html`.

John Seely Brown and Richard R Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive science*, 2(2):155–192, 1978.

Geoffrey I Webb, Michael J Pazzani, and Daniel Billsus. Machine learning for user modeling. *User modeling and user-adapted interaction*, 11(1):19–29, 2001.

Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

O Bjorkoy. *User modeling on the web: an exploratory review of recommendation systems*. PhD thesis, PhD thesis, NTNU Trondheim, 2010.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

Sangwon Bae, DENZIL FERREIRA, BRIAN SUFFOLETTO, JUAN C PUYANA, RYAN KURTZ, TAMMY CHUNG, and ANIND K DEY. Detecting drinking episodes in young adults using smartphone-based sensors. 2017.

Felix Kosmalla, Florian Daiber, and Antonio Krüger. Climbsense: Automatic climbing route recognition using wrist-worn inertia measurement units. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2033–2042. ACM, 2015.

Zhicheng Liu, Himel Dev, Mira Dontcheva, and Matthew Hoffman. Mining, pruning and visualizing frequent patterns for temporal event sequence analysis.

Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y Zhao. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 225–236. ACM, 2016.

Sha Zhao, Julian Ramos, Jianrong Tao, Ziwen Jiang, Shijian Li, Zhaohui Wu, Gang Pan, and Anind K Dey. Discovering different kinds of smartphone users through their application usage behaviors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 498–509. ACM, 2016.

Sergey I Nikolenko and Anton Alekseyev. User profiling in text-based recommender systems based on distributed word representations. In *5th Conference on Analysis of Images, Social Networks, and Text (AIST 2016)*. Springer, 2016.

Sahin Cem Geyik, Ali Dasdan, and Kuang-Chih Lee. User clustering in online advertising via topic models. *arXiv preprint arXiv:1501.06595*, 2015.

Jin-Hyuk Hong, Julian Ramos, and Anind K Dey. Toward personalized activity recognition systems with a semipopulation approach. *IEEE Transactions on Human-Machine Systems*, 46(1):101–112, 2016.

Mashfiqui Rabbi, Min Hane Aung, and Tanzeem Choudhury. Towards health recommendation systems: An approach for providing automated personalized health feedback from mobile data. In *Mobile Health*, pages 519–542. Springer, 2017.

Judy Robertson and Maurits Kaptein. *Modern statistical methods for HCI*. Springer, 2016.

Jie Yin, Qiang Yang, Dou Shen, and Ze-Nian Li. Activity recognition via user-trace segmentation. *ACM Transactions on Sensor Networks (TOSN)*, 4(4):19, 2008.

Hung Hai Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research*, 17:451–499, 2002.

Anirban Roychowdhury, Ke Jiang, and Brian Kulis. Small-variance asymptotics for hidden markov models. In *Advances in Neural Information Processing Systems*, pages 2103–2111, 2013a.

Ke Jiang, Brian Kulis, and Michael I Jordan. Small-variance asymptotics for exponential family dirichlet process mixture models. In *NIPS*, pages 3167–3175, 2012.

Eytan Adar, Mira Dontcheva, and Gierad Laput. Commandspace: Modeling the relationships between tasks, descriptions and features. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 167–176. ACM, 2014.

Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)*, 35(2):11, 2016.

Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 97–104. IEEE, 2011.

Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

Mustansar Ali Ghazanfar, Adam Prügel-Bennett, and Sandor Szedmak. Kernel-mapping recommender system algorithms. *Information Sciences*, 208:81–104, 2012.

Maks Ovsjanikov and Ye Chen. Topic modeling for personalized recommendation of volatile items. *Machine Learning and Knowledge Discovery in Databases*, pages 483–498, 2010.

Diane J Hu, Rob Hall, and Josh Attenberg. Style in the long tail: Discovering unique interests with latent variable models in large scale social e-commerce. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1640–1649. ACM, 2014.

Peter Brusilovski, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web: methods and strategies of web personalization*, volume 4321. Springer Science & Business Media, 2007.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

Christian P Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

Arnaud Doucet, Nando De Freitas, Neil Gordon, et al. *Sequential Monte Carlo methods in practice*. Springer New York, 2001.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72:269–342, 2010a.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

Erik Blaine Sudderth. *Graphical models for visual object recognition and tracking*. PhD thesis, Massachusetts Institute of Technology, 2006.

Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, pages 1152–1174, 1974.

Michael D Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

David Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII1983*, pages 1–198, 1985.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006a.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006b.

J. F. C. Kingman. *Poisson Processes*. Oxford Studies in Probability. Oxford University Press, 1993.

Ardavan Saeedi and Alexandre Bouchard-Côté. Priors over recurrent continuous time processes. In *Advances in Neural Information Processing Systems*, pages 2052–2060, 2011.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Michael Bryant and Erik B Sudderth. Truly nonparametric online variational inference for hierarchical dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 2699–2707, 2012.

David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

Matthew Johnson and Alan Willsky. Stochastic variational inference for bayesian time series models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1854–1862, 2014.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. The infinite pcfg using hierarchical dirichlet processes. In *EMNLP-CoNLL*, pages 688–697, 2007.

Chia-ying Lee, Yu Zhang, and James R Glass. Joint learning of phonetic units and word pronunciations for asr. In *EMNLP*, pages 182–192, 2013.

Brenden M Lake, Chia-ying Lee, James R Glass, and Joshua B Tenenbaum. One-shot learning of generative speech concepts. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society*, 2014.

Katherine A Heller, Yee W Teh, and Dilan Görür. Infinite hierarchical hidden markov models. In *International Conference on Artificial Intelligence and Statistics*, pages 224–231, 2009.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319. ACM, 2008.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. A sticky hdp-hmm with application to speaker diarization. *The Annals of Applied Statistics*, pages 1020–1056, 2011.

Matthew Beal and Praveen Krishnamurthy. Gene expression time course clustering with countably infinite hidden markov models. *arXiv preprint arXiv:1206.6824*, 2012.

Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.

Kevin P Murphy and Mark A Paskin. Linear-time inference in hierarchical hmms. *Advances in neural information processing systems*, 2:833–840, 2002.

Thomas S Stepleton, Zoubin Ghahramani, Geoffrey J Gordon, and Tai S Lee. The block diagonal infinite hidden markov model. In *International Conference on Artificial Intelligence and Statistics*, pages 552–559, 2009.

Nuria Olivera, Ashutosh Gargb, and Eric Horvitza. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96:163–180, 2004.

Nam T Nguyen, Dinh Q Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 955–960. IEEE, 2005.

Lexing Xie, Shih-Fu Chang, Ajay Divakaran, and Huifang Sun. Unsupervised discovery of multilevel statistical video structures using hierarchical hidden markov models. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 3, pages III–29. IEEE, 2003.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 256–265. Morgan Kaufmann Publishers Inc., 1998.

Miikka Ermes, Juha Parkka, Jani Mantyjarvi, and Ilkka Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):20–26, 2008.

Juha Pärkkä, Miikka Ermes, Panu Korpipää, Jani Mäntyjärvi, Johannes Peltola, and Ilkka Korhonen. Activity classification using realistic data from wearable sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):119–128, 2006.

Rosalind W Picard, Elias Vyzas, and Jennifer Healey. Toward machine emotional intelligence: Analysis of affective physiological state. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(10):1175–1191, 2001.

Jennifer A Healey and Rosalind W Picard. Detecting stress during real-world driving tasks using physiological sensors. *Intelligent Transportation Systems, IEEE Transactions on*, 6(2):156–166, 2005.

Jamey Kain, Chris Stokes, Quentin Gaudry, Xiangzhi Song, James Foley, Rachel Wilson, and Benjamin de Bivort. Leg-tracking and automated behavioural classification in drosophila. *Nature communications*, 4:1910, 2013.

Matthew James Johnson. *Bayesian time series models and scalable inference*. PhD thesis, Massachusetts Institute of Technology, 2014.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

Empatica. E4 sensor. `https://www.empatica.com/e4-wristband`, 2015. Accessed: 2015-12.

Wolfram Boucsein. *Electrodermal activity*. Springer Science & Business Media, 2012.

A Sano et al. Discriminating high vs low academic performance, selfreported sleep quality, stress level, and mental health using personality traits, wearable sensors and mobile phones. *Body Sensor Networks (BSN)(to appear)*, 2015.

Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping sub-second structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

James D Crall, Nick Gravish, Andrew M Mountcastle, and Stacey A Combes. Beetag: a low-cost, image-based tracking system for the study of animal behavior and locomotion. *PloS one*, 10(9): e0136487, 2015.

Jeremy Freeman, Nikita Vladimirov, Takashi Kawashima, Yu Mu, Nicholas J Sofroniew, Davis V Bennett, Joshua Rosen, Chao-Tsung Yang, Loren L Looger, and Misha B Ahrens. Mapping brain activity at scale with cluster computing. *Nature methods*, 11(9):941–950, 2014.

Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99): 20140672, 2014.

Micha Mandel. Estimating disease progression using panel data. *Biostatistics*, 11(2):304–316, 2010.

Monir Hajiaghayi, Bonnie Kirkpatrick, Liangliang Wang, and Alexandre Bouchard-Côté. Efficient Continuous-Time Markov Chain Estimation. In *International Conference on Machine Learning*, 2014.

Mogens Bladt and Michael Sørensen. Efficient estimation of transition rates between credit ratings from observations at discrete time points. *Quantitative Finance*, 9(2):147–160, 2009.

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23): e215–e220, 2000.

V. Rao and Y W Teh. Fast MCMC sampling for Markov jump processes and extensions. *The Journal of Machine Learning Research*, 14(1):3295–3320, 2013.

Theodore J Perkins. Maximum likelihood trajectories for continuous-time markov chains. In *NIPS*, pages 1437–1445, 2009.

Brian Kulis and M I Jordan. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. In *International Conference on Machine Learning*, 2012.

T. Broderick, Brian Kulis, and M I Jordan. MAD-Bayes: MAP-based Asymptotic Derivations from Bayes. In *International Conference on Machine Learning*, 2013.

Anirban Roychowdhury, Ke Jiang, and Brian Kulis. Small-Variance Asymptotics for Hidden Markov Models. In *Advances in Neural Information Processing Systems*, pages 2103–2111, 2013b.

Jane Lange. *Latent Continuous Time Markov Chains for Partially-Observed Multistate Disease Processes*. PhD thesis, 2014.

Mogens Bladt and Michael Sørensen. Statistical inference for discretely observed markov jump processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3): 395–410, 2005.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3): 269–342, 2010b.

Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. In *Journal of Machine Learning Research*, pages 1345–1382, 2005.

DLMF. NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/, Release 1.0.8 of 2014-04-25. URL `http://dlmf.nist.gov/`. Online companion to Olver et al. (2010).

Y W Teh, M I Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006c.

Hobolth Asger and Jensen Jens Ledet. Statistical inference in evolutionary models of dna sequences via the em algorithm. *Statistical Applications in Genetics and Molecular Biology*, 4(1):1–22, 2005.

G.B. Moody and R.G. Mark. A database to support development and evaluation of intelligent intensive care monitoring. In *Computers in Cardiology, 1996*, pages 657–660, Sept 1996.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press, 2004.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics*, 2015.

Inderjit S Dhillon and Suvrit Sra. Modeling data using directional distributions. Technical report, Technical Report TR-03-06, Department of Computer Sciences, The University of Texas at Austin. URL ftp://ftp. cs. utexas. edu/pub/techreports/tr03-06. ps. gz, 2003.

Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J Mooney. Spherical topic models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 903–910, 2010.

John Paisley, Chingyue Wang, David M Blei, and Michael I Jordan. Nested hierarchical dirichlet processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2):256–270, 2015.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

Yulan He, Chenghua Lin, Wei Gao, and Kam-Fai Wong. Dynamic joint sentiment-topic model. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):6, 2013.

Longqi Yang, Chen Fang, Hailin Jin, Matthew D Hoffman, and Deborah Estrin. Personalizing software and web services by integrating unstructured application usage traces. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 485–493. International World Wide Web Conferences Steering Committee, 2017.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 2012.

Siddarth Gopal and Yiming Yang. Von mises-fisher clustering models. 2014.

Anjie Fang, Craig Macdonald, Iadh Ounis, and Philip Habel. Using word embedding to evaluate the coherence of topics from twitter data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1057–1060. ACM, 2016.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics, 2010.

Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *EACL*, pages 530–539, 2014.

Christopher M Bishop. Mixture density networks. 1994.

Yichuan Tang and Ruslan R Salakhutdinov. Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, pages 530–538, 2013.

Yann N Dauphin and David Grangier. Predicting distributions with linearizing belief networks. *arXiv preprint arXiv:1511.05622*, 2015.

Gaurav Sharma and Raja Bala. *Digital color imaging handbook*. CRC press, 2002.

Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36 (4):118, 2017.

Joon-Young Lee, Kalyan Sunkavalli, Zhe Lin, Xiaohui Shen, and In So Kweon. Automatic content-aware color and tone stylization. *arXiv preprint arXiv:1511.03748*, 2015.

Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. Selph: Progressive learning and support of manual photo color enhancement. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2520–2532. ACM, 2016.

Radford M Neal. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

George Tucker, Andriy Mnih, Chris J Maddison, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv preprint arXiv:1703.07370*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Alexandre Bouchard-Côté and Michael I Jordan. Optimization of structured mean field objectives. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 67–74. AUAI Press, 2009.

Tommi S Jaakkola and Michael I Jordan. Improving the mean field approximation via the use of mixture distributions. In MI Jordan, editor, *Learning in Graphical Models*, pages 163–173. Springer, 1998.

Natalia Flerova, Emma Rollon, and Rina Dechter. Bucket and mini-bucket schemes for m best solutions over graphical models. In *Graph Structures for Knowledge Representation and Reasoning*, pages 91–118. Springer, 2012.

Chen Yanover and Yair Weiss. Finding the M most probable configurations in arbitrary graphical models. In *Advances in Neural Information Processing Systems*, page None, 2003.

Kerrie L Mengersen, Richard L Tweedie, et al. Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24:101–121, 1996.

Sean P Meyn and Richard L Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993.

Peter Hall, John T Ormerod, and MP Wand. Theory of Gaussian variational approximation for a Poisson mixed model. *Statistica Sinica*, 21:369–389, 2011.

Bo Wang, DM Titterington, et al. Convergence properties of a general algorithm for calculating variational bayesian estimates for a normal mixture model. *Bayesian Analysis*, 1:625–650, 2006.

Neil D Lawrence. *Variational inference in probabilistic models*. PhD thesis, University of Cambridge, 2000.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48:259–302, 1986.

Dorothy M Greig, Bruce T Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

Samuel Gershman, Matt Hoffman, and David Blei. Nonparametric variational inference. *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Andrew C Miller, Nicholas Foti, and Ryan P Adams. Variational boosting: Iteratively refining posterior approximations. *arXiv preprint arXiv:1611.06585*, 2016.

Andrew Frank, Padhraic Smyth, and Alexander T Ihler. Particle-based variational inference for continuous systems. In *Advances in Neural Information Processing Systems*, 2009.

Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17:295–311, 2008.

Philip Schniter, Lee C Potter, and Justin Ziniel. Fast Bayesian matching pursuit. In *Information Theory and Applications Workshop*, pages 326–333. IEEE, 2008.

Beatrix Jones, Carlos Carvalho, Adrian Dobra, Chris Hans, Chris Carter, and Mike West. Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science*, 20: 388–400, 2005.

Matthew J Beal, Zoubin Ghahramani, and Carl E Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 577–584, 2002.

Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. Beam sampling for the infinite hidden Markov model. In *Proceedings of the 25th international conference on Machine learning*, pages 1088–1095. ACM, 2008.

Daniel N Osherson, Joshua Stern, Ormond Wilkie, Michael Stob, and Edward E Smith. Default probability. *Cognitive Science*, 15:251–269, 1991.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Composing graphical models with neural networks for structured representations and fast inference. In *Neural Information Processing Systems*, 2016.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

Hal Daume. Fast search for Dirichlet process mixture models. In *International Conference on Artificial Intelligence and Statistics*, pages 83–90, 2007.

Lianming Wang and David B Dunson. Fast Bayesian inference in Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 20, 2011.

John R Anderson. The adaptive nature of human categorization. *Psychological Review*, 98:409–429, 1991.

Frank Wood and Michael J Black. A nonparametric Bayesian alternative to spike sorting. *Journal of Neuroscience Methods*, 173:1–12, 2008.

R Quian Quiroga, Z Nadasdy, and Y Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16:1661–1687, 2004.

Paul Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14:11–21, 2004.

Robert H Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58:86–88, 1987.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, editors. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010. Print companion to DLMF.