Composing Deep Learning and Bayesian Nonparametric Methods

Aonan Zhang

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Composing Deep Learning and Bayesian Nonparametric Methods

Aonan Zhang

Recent progress in Bayesian methods largely focus on non-conjugate models featured with extensive use of black-box functions: continuous functions implemented with neural networks. Using *deep* neural networks, Bayesian models can reasonably fit big data while at the same time capturing model uncertainty. This thesis targets at a more challenging problem: how do we model general random objects, including discrete ones, using random functions? Our conclusion is: *many (discrete) random objects are in nature a composition of Poisson processes and random functions*. Thus, all discreteness is handled through the Poisson process while random functions captures the rest complexities of the object. Thus the title: composing deep learning and Bayesian nonparametric methods.

This conclusion is not a conjecture. In spacial cases such as latent feature models , we can prove this claim by working on infinite dimensional spaces, and that is how Bayesian nonparametric kicks in. Moreover, we will assume some regularity assumptions on random objects such as exchangeability. Then the representations will show up magically using representation theorems. We will see this two times throughout this thesis.

One may ask: when a random object is too simple, such as a non-negative random vector in the case of latent feature models, how can we exploit exchangeability? The answer is to aggregate infinite random objects and map them altogether onto an infinite dimensional space. And then assume exchangeability on the infinite dimensional space. We demonstrate two examples of latent feature models by (1) concatenating them as an infinite sequence (Section 2, 3) and (2) stacking them as a 2d array (Section 4).

Besides, we will see that Bayesian nonparametric methods are useful to model discrete patterns in time series data. We will showcase two examples: (1) using variance Gamma processes to model change points (Section 5), and (2) using Chinese restaurant processes to model speech with switching

speakers (Section 6).

We also aware that the inference problem can be non-trivial in popular Bayesian nonparametric models. In Section 7, we find a novel solution of online inference for the popular HDP-HMM model.

Table of Contents

List of Figures v

Part I Overview 1

Chapter 1 Introduction 2

1.1 A classical theory for random partitions . . . 2

1.2 Existing theory on latent feature models . . . 4

1.2.1 LFMs as random partitions . . . 5

1.2.2 LFMs with randomly permuted features . . . 6

1.3 Correlated LFMs . . . 8

Part II Composing Deep Learning and Bayesian Nonparametric Methods in Latent Feature Models 9

Chapter 2 Markov Mixed Membership Models 10

2.1 Model Description . . . 12

2.1.1 Markov mixed membership models . . . 12

2.1.2 Relationship to tree-structured models . . . 13

2.1.3 Related work . . . 14

2.2 Scalable Variational Inference . . . 15

2.2.1 Local variables . . . 16

2.2.2 Global variables . . . 17

2.2.3 Stochastic variational inference . . . 18

2.3 Experiments . . . 19

2.3.1 Document modeling . . . 19

2.3.2 Million song dataset . . . 23
Chapter 3 Markov Latent Feature Models 25
3.1 Feature Allocation via Sequences . . . 26
3.2 Markov Latent Feature Models . . . 28
3.2.1 A parametric model . . . 28
3.2.2 A nonparametric model . . . 29
3.2.3 Application to a linear Gaussian model . . . 30
3.2.4 Discussion . . . 30
3.3 Inference . . . 30
3.3.1 Batch Variational Inference . . . 31
3.3.2 Stochastic Variational Inference . . . 34
3.4 Experiments . . . 36
3.4.1 HGDP-CEPH Cell Line Panel . . . 36
3.4.2 Image denoising . . . 38
Chapter 4 Random Function Priors for Correlation Modeling 40
4.1 Z as a random matrix? . . . 43
4.2 Z as a random measure . . . 44
4.2.1 Population random measure embedding . . . 44
4.2.2 Construction via completely random measures . . . 45
4.3 An illustration on topic modeling . . . 45
4.3.1 The model . . . 45
4.3.2 Amortized variational inference . . . 47
4.3.3 Network architectures . . . 49
4.4 Experiments . . . 50
4.4.1 Batch experiments . . . 50
4.4.2 Online experiments . . . 52
4.5 Discussion . . . 54
4.5.1 Connections with other random objects . . . 54
4.5.2 Deep hierarchical Bayesian models . . . 54
4.5.3 Posterior inference bottleneck . . . 55

Part III Composing Deep Learning and Bayesian Nonparametric Methods in Time-Series Modeling **61**

Chapter 5 Deep Bayesian Nonparametric Tracking 62

5.1 Motivations . . . 63

5.2 The Model . . . 64

5.2.1 Basic setup: Dynamic matrix factorization . . . 64

5.2.2 Variance gamma process on W . . . 65

5.2.3 Temporal tracking in H and data generation . . . 66

5.2.4 Extension: A deep likelihood model . . . 66

5.3 Variational inference . . . 67

5.3.1 Linear Gaussian observational model . . . 67

5.3.2 Extension: Variational auto-encoder model . . . 68

5.4 Further discussion . . . 70

5.4.1 Modeling velocity and acceleration of drift . . . 71

5.4.2 Prediction . . . 71

5.5 Experiments . . . 72

5.5.1 Methods and evaluations . . . 72

5.5.2 Stock market crash and recovery, 2008-2012 . . . 73

5.5.3 NFL tweets . . . 76

5.6 Conclusion . . . 77

Chapter 6 Fully Supervised Speaker Diarization 79

6.1 Motivations . . . 79

6.2 Baseline system using clustering . . . 81

6.3 Unbounded interleaved-state RNN . . . 82

6.3.1 Overview of approach . . . 82

6.3.2 Details on model components . . . 83

6.3.3 MLE Estimation . . . 85

6.3.4 MAP Decoding . . . 86

6.4 Experiments . . . 87

6.4.1 Speaker recognition model . . . 87

6.4.2 UIS-RNN setup . . . 87

6.4.3 Evaluation protocols . . . 88
6.4.4 Datasets . . . 88
6.4.5 Results . . . 88
6.5 Conclusions . . . 90

Part IV Inference for Bayesian Nonparametric Models **91**

Chapter 7 Stochastic Variational Inference for the HDP-HMM 92
7.1 Motivations . . . 92
7.2 The HDP-HMM . . . 94
7.2.1 Stick-breaking construction . . . 94
7.3 Variational inference for the HDP-HMM . . . 96
7.3.1 The state transition matrix . . . 97
7.3.2 A local lower bound using $q(\mathbf{s}_d|\mathbf{z}_d)$ . . . 97
7.3.3 Mapping atoms between DP levels . . . 99
7.3.4 Stochastic variational inference . . . 99
7.4 Experiments . . . 100
7.4.1 Artificial data . . . 100
7.4.2 Alice's Adventures in Wonderland . . . 102
7.4.3 Million Song dataset . . . 104
7.5 Conclusion . . . 106
7.6 Appendix . . . 106
Bibliography 109

List of Figures

Figure 2.1 Comparison between three path-based models. (Left) The tree-structured nested Chinese restaurant process (nCRP) selects one path per group; (Mid.) the tree-based nested hierarchical Dirichlet process (nHDP) places high probability on a subtree for each group; (Right) the proposed graph-based Markov mixed membership model selects one path per group using a Markov random walk on the fully connected set of nodes (an example high-probability connectivity is depicted in the background here). 12

Figure 2.2 Held-out perplexity results. The Markov transition model (Markov M3) overall achieves best performance among parametric models. Its best performance is even better than the state-of-the-art nonparametric nHDP. 19

Figure 2.3 Predictive performance for online Markov M3 and online LDA. Markov M3 is consistantly better for various number of topics through the entire learning process. . 20

Figure 2.4 Topic paths selected by three documents. The size of the node indicates the proportion of the topic. 21

Figure 2.5 Selected topic subgraph from a 200-topic graph learned by online Markov M3 on New York Times dataset. The graph shows Markov transitions with high probability among topics. 22

Figure 2.6 Sensitivity analysis for (left) the truncation level of sticks, and (right) the stick-breaking concentration parameter. Results are shown in terms of average log likelihood on a test set. 22

Figure 2.7 Markov transition paths learned from three songs without knowing their tags (the ground truth tags are marked in the parenthesis below). The black node denotes the initial state. The size of nodes indicates the proportion for that factor, and the thickness of arrows indicates the transition probability between factors. All the paths are embedded in the graph (the gray background). 23

Figure 3.1 An illustration of the construction of a 0-1 matrix from a sequential process, which we define to be a mixture of recurrent Markov chains. On the LHS, the chain $\mathbf{Z}$ starts from a null state $Z_0 = 0$ and generates four blocks (subsequences) $\mathbf{Z}_{\psi(1)}$ to $\mathbf{Z}_{\psi(4)}$ by returning to 0 four times (shown as four colored paths on the graph). One the RHS, this sequence constructs a 0-1 matrix with four rows and the columns indicating the unique set of states visited in each block. 27

Figure 3.2 (Left) Factors learned from BPFA, nCRP, and iMLFM on the HGDP-CEPH dataset. Observations from various regions are aligned vertically, as displayed on the left side. (Right) Graph learned from iMLFM and nCRP. 32

Figure 3.3 Average predictive result on HGDP-CEPH dataset. 36

Figure 3.4 (Left) A similarity-preserving 2-d embedded graph of the learned dictionary elements using their transition probabilities. We show edges about a threshold, and the direction of an edge between two dictionary elements according to the higher transition probability. (Right) Feedback maps for dictionary elements in various regions in the graph on the left. 37

Figure 3.5 The four images used in our experiments. 38

Figure 4.1 Example of two equivalent representations: (left) Paintbox model for $[Z_{n1}, Z_{n2}, Z_{n3}]$ by partitioning a unit square into eight regions, one for each distinct value. (right) Factorize the paintbox into three feature paintboxes, one each for a latent feature. Three examples of u_n that determine Z_n are demonstrated as dots in the partition paintbox, and as lines across feature paintboxes. 42

Figure 4.2 Decoupling a separately exchangeable discrete random measure ξ into two parts. 43

Figure 4.3 (a) Graphical representation of our proposed model. Solid arrows represent the generative process and dashed arrows show the VAE part of the posterior. We organize local parameters that belong to a document/word into boxes and remove all sub-indices. We use stochastic natural gradient ascent for θ and use stochastic gradient ascent for $[\ell, V, g, f]$. 47

Figure 4.4 (b) Left: The architecture we used in our experiments. Right: Various layer designs. 47

Figure 4.5 A paintbox demonstration of salient topics learned from the one million New York Times dataset. In each paintbox on the LHS, pixel (x, y) represents the topic strength $Z_{(x,y),k}$ as a function of $h_{(x,y)}$ for a particular topic k. We also show embeddings of three articles in the same space, as well as their projection onto selected paintboxes. Each article is connected to its most-used topics. 52

Figure 4.6 (a) Online performance comparisons between DILN and PRME. (b) Online versus batch. (c) Time cost comparison between updating local and global variables. (d) Ranked topic usage proportions in the posterior, indicating nonparametric functionality. 54

Figure 4.7 Decouple separately exchangeable random measure ξ into four parts. 55

Figure 5.1 (a) Partitioning sequential data into time blocks. (b) Each block is modeled as a dynamic matrix factorization, where the left matrix uses a gamma process for change-point detection, and the right matrix is Brownian motion. Both matrices are time-evolving: the left matrix across blocks (using a jump process) and the right matrix within each block. 64

Figure 5.2 A rank-one example of the deep BNP tracking model for three data streams. A continuous-time variance gamma process is discretized into time blocks at finer resolutions than the large jumps in the process. A single Brownian motion multiplies with these variance gamma processes and is passed into a neural network, which parameterizes the mean and covariance of the observed data streams. 65

Figure 5.3 Left: Encoder network using an LSTM. Right: Re-parameterization and sequential sampling from $q_\theta(H_{t,j+1}|X_{t,1:j})$. 68

Figure 5.4 The jumps learned by our deep VGP-VAE model. (a) Summation of posterior means for all stocks $\sum_m \mathbb{E}[\gamma_{t,m}]$. This demonstrates a significant jump in the market corresponding to the crash of 2008-2009. (b) the stock price for three companies and their corresponding gamma processes. Spikes indicate large jumps of their relevant locations in the row space of W_t (not shown). 73

Figure 5.5 Top plots show raw counts for 6 different words. We can see the word frequency is very noisy in the entire period. Lower shaded plots show posterior mean of the gamma process $\mathbb{E}[\gamma_{t,m}]$ showing the jumps in word embedding locations learned by our model. Note that (i) the resolution of the plots are different, the top being hourly and the bottom per-day; (ii) the learned spikes show jumps of latent row vectors in W_t through a variance gamma process (not shown). So our method is essentially different from sparse recovery algorithms, such as soft-thresholding [Don95]. 76

Figure 5.6 Sensitivity result on nfl tweet data set with various time windows. To get both good predictive and interepretable results, the best choice is to choose an intermediate discretization. 78

Figure 6.1 The baseline system architecture [WDW+18]. 81

Figure 6.2 Generative process of UIS-RNN. Colors indicate labels for speaker segments. There are four options for y_7 given $\mathbf{x}_{[6]}, y_{[6]}$. 85

Figure 7.1 Comparison between the transition structure of the HMM and the stick-breaking construction of HDP-HMM. Each G_i represents a transition distribution from state i, and the color of a stick corresponds to one state. Left: In the HMM the state and the column index are one-to-one. Right: For the stick-breaking construction of the HDP-HMM, multiple sticks in each row may point to the same state (same color) and there is no one-to-one mapping between column and row. For example, a transition from state 3 to 5 takes place if $s = 5$ or $s = 6$. By holding the sequence $\mathbf{s}_d$ fixed and changing a DP indicator c_{km} for a selected stick (i.e., changing the color of a stick), the state transitions for *all* subsequent states may change. This presents an inference challenge for the HDP-HMM not faced by [WPB11a]. 93

Figure 7.2 Top: Log likelihood for our variational HDP-HMM and beam sampling. Bottom: Number of posterior states inferred by variational HDP-HMM and beam sampling. Results are averaged over 20 random experiments. "pos" indicates A_{pos} and "neg" indicates A_{neg} was used. 101

Figure 7.3 Posterior variational bound for each chapter ($\times 10^4$). The black plots represents results for HMM with various number of states. The red, blue, green bars separately show the average result for the HDP-HMM, HDP-HMM with direct assignment, and HDP-HMM with direct assignment and fully-factorized mean-field assumption in variational posterior, all with a truncation level of 50. The standard deviation is shown on the left hand side, for each HDP-HMM model. The dashed lines are the number of states used in the posterior of according models averaged over multiple runs. 102

Acknowledgments

Even before coming to Columbia, I was a big fan of Jim Pitman's monograph on combinatorial stochastic processes, where a series of statistical characterizations over random partitions are demonstrated. Pushing forward the boundary of those nice statistical models are always my pursuit throughout the Ph.D. study. I would like to thank Columbia university that offers me a professional and inclusive academic environment to pursue my academic goal.

I would like to especially thank my advisor John Paisley, who raised an interesting idea on modeling graph structured latent feature model, which motivates theories and new models thereafter. Professor David Blei's reading group is another source for me to dig into different areas of machine learning and social science. The "unbiased" selection of reading materials encourages me to be open-minded towards other fields and their philosophy. Moreover, I would like to thank my academic committee, Shih-Fu Chang, John Paisley, John Wright, David Blei, and John Cunningham for their comment on my research and thesis.

My research cannot be at the current stage without other's support. My co-authors, John Paisley, San Gultekin, Chong Wang, Quan Wang, Zhenyao Zhu, each offers great intuitions to my research. I'm afraid that I cannot list all people who contribute to this thesis. Many thanks to them all.

Aonan Zhang

Sep 30, 2019

New York

To my parents

Part I

Overview

Chapter 1

Introduction

Learning flexible, interpretable representations for data is a fundamental goal in modern machine learning. Often, one aims to learn some latent features from the entire dataset and then represent each object using those latent features. In this section, we setup this problem in a probabilistic way as a latent feature model (LFM), which is a random multiset over N objects (labelled as $[N]$ wlog.). LFM can be treated a simplified version of random partitions. First, we will introduce a nice, existing theory of random partitions. Then we point out that this nice theory cannot be extended to LFMs. Thus our motivation is to find new theories for LFMs.

1.1 A classical theory for random partitions

Let N denotes the number of individuals. A *partition* of $[N]$, denoted by $\pi_N = \{A_1, \ldots, A_K\}$ is a set of mutually exclusive subsets A_k whose union is $[N]$. A *random partition* Π_N is exchangeable if its distribution is invariant under any permutation of $[N]$. As we shall see, the exchangeability assumption is crucial for the theory of random partitions. We then introduce a consistent sequence[1] of random partitions $(\Pi_N)_{N\in\mathbb{N}}$ and define restriction operator $\mathcal{R}_M(\cdot)$ as restricting a random partition to $[M]$. For example, let $\pi_6 = \{\{1,2,4\},\{6\},\{3,5\}\}$. Then $\mathcal{R}_4(\pi_6) = \{\{1,2,4\},\{3\}\}$. A restriction operator links any two random partitions Π_M, Π_N where $M \leq N$ via consistency. Π_M, Π_N are called distributional consistent if $\mathcal{R}_M(\Pi_N) =_d \Pi_M$. We call a sequence of random partitions $(\Pi_N)_{N\in\mathbb{N}}$ consistent if any

[1] A single exchangeable random partition Π_N turns out to be quite uninteresting. Actually it can always be constructed via two steps: randomly fill an urn with N different colored balls and then sample without replacement from the urn [Ald85, Sch12].

pair of them are consistent, and we call a consistent sequence of random partitions an infinite random partition $\Pi_\infty \coloneqq (\Pi_N)_{N\in\mathbb{N}}$.

Theorem 1 [Kin78]. *An infinite random partition Π_∞ can be generated as follows. First sample a sequence of ranked frequencies $p_1 \geq p_2 \geq \dots$ such that $p = \sum_i p_i \leq 1$. Then i.i.d. sample individuals according to a mixture distribution $\sum_i p_i \delta_{\phi_i}(dx) + (1-p)\rho(dx)$, where ϕ_i are different points on $\mathbb{R}$ and ρ is a diffuse distribution on $\mathbb{R}$. Finally cluster individuals that coincide.*

Note that in the mixture model $\sum_i p_i \delta_{\phi_i}(dx) + (1-p)\rho(dx)$, all the points that fall in ρ will be isolated from each other. Since those singletons are not interesting for most purposes, we will assume $p = 1$. And we call an infinite random partition *regular* if it has no singleton. Theorem 1 tells that regular infinite random partitions are essentially i.i.d. sampling from discrete distributions. For practical use this is good since we only need to consider models that generate a discrete distribution.

An alternative way to think about random partitions is through their combinatorial properties. Note that the law of any exchangeable random partition Π_N can be represented as

$$\mathbb{P}(\Pi_N = \{A_1, \dots, A_K\}) = p(|A_1|, \dots, |A_K|). \tag{1.1}$$

Here $p(\cdot)$ is called the exchangeable partition probability function (EPPF) for Π_N. It is thus possible to build consistent EPPFs that are naturally derived from infinite random partitions. Let $p(\cdot)$ be a symmetric function from any composition of any integer N to $[0,1]$ that satisfies

$$p(n_1, \dots, n_K) = \sum_{i=1}^{K} p(\dots, n_i + 1, \dots) + p(n_1, \dots, n_K, 1). \tag{1.2}$$

EPPFs can be seen as a distribution of random partitions by marginalizing out the random measure $\sum_i p_i \delta_{\phi_i}(dx)$. In some special cases, it can be tractable to directly sample cluster assignment of a new incoming object given existing partitions, or using MCMC to iteratively sample assignments for objects conditioning on partitions of the rest objects. Gibbs partitions is one of the most famous examples.

Example 1 (Gibbs partitions)**.** A random partition Π_N is called a Gibbs partition if

$$\mathbb{P}(\Pi_N = \{A_1, \dots, A_K\}) = \frac{v_K \prod_{i=1}^{K} w_{|A_i|}}{B_N(v_\bullet, w_\bullet)}, \tag{1.3}$$

where $B_N(v_\bullet, w_\bullet)$ is the normalization term for non-negative sequences $v_\bullet \coloneqq (v_1, v_2, \dots)$ and $w_\bullet \coloneqq$

$(w_1, w_2, \ldots)$. To be precise, $B_N(v_\bullet, w_\bullet) = \sum_{K=1}^{N} v_K B_{N,K}(w_\bullet)$, where

$$B_{N,K}(w_\bullet) = \sum_{\{A_1,\ldots,A_K\}\in\mathcal{P}_N^K} \prod_{i=1}^{K} w_{|A_i|} \tag{1.4}$$

is called the (N, K)-th partial Bell polynomial [Pit06]. ■

It is worth noting that calculating the (N, K)-th partial Bell Polynomial only requires polynomial time according to the following recurrent rule, due to the product structure of w.

$$B_{N,K}(w_\bullet) = \sum_{i=1}^{N-K+1} \binom{N-1}{i-1} w_i B_{N-i,K-1}(w_\bullet) \tag{1.5}$$

Under special cases, even simpler iterative rule follows.

Example 2 (Stirling numbers of the second kind)**.** Let $S_{N,K} \coloneqq B_{N,K}(1^\bullet)$ be the number of possible partitions of $[N]$ into K clusters. Simpler recurrence relations exist: $S_{N,K} = S_{N-1,K-1} + K \cdot S_{N-1,K}$. Applying $S_{N,K}$ to (1.3) with $v_\bullet = 1^\bullet$ one gets a uniform random partition. ■

Unlike random partitions that enjoy nice equivalent representations as sampling from random discrete distributions, latent feature models (LFM) are much harder to manipulate in theory and practice. The fundamental reason is that LFMs allows each individual to possess *multiple latent features*, which brings about additional challenges of modelling *correlations* among latent features. This will be made precise in the next section.

1.2 Existing theory on latent feature models

Latent feature models (LFM) are natural extension of random partitions by allowing each individual possess multiple (latent) features. Let N denotes the number of individuals. We call a feature allocation over $[N]$ as $f_N = \{A_1, \ldots, A_K\}$ where A_i denotes the set of individuals that possess feature i. Here we only look at the combinatorial structure and don't index each feature. So f_N is a set of subsets of $[N]$. E.g. $f_6 = \{\{3, 5\}, \{1, 2, 3\}, \{6\}\}$. Similar as random partitions, we can introduce restriction operators $\mathcal{R}_M(\cdot)$, random feature allocations F_M, F_N, where $M \leq N$ and consistency between them as $\mathcal{R}_M(F_N) =_d F_M$.

Definition 1 (Latent feature models)**.** An infinite latent feature model $(F_N)_{N\in\mathbb{N}}$, which we will call latent feature model for short, is a consistent sequence of random feature allocations such that for each

$N \in \mathbb{N}$, F_N satisfies following assumptions:

i. (Individual Exchangeability/IE) All individuals are exchangeable.

ii. (Finiteness/F) Each individual possesses finite number of latent features almost surely. That is, K is almost surely finite for finite N. ■

Apart from above constraints, LFMs suffers from additional ill-behaved properties than random partitions. For example, [BPJ+13] shows that even an extremely simple LFM does not enjoy a "Gibbs-type" representation.

Example 3 (A trivial non Gibbs-type LFM)**.** Consider an LFM generated as follows. Given $0 < p_1, p_2 < 1$, for each individual we independently sample its features by two consecutive biased coin tosses, first with head probability p_1 and second with probability p_2. Then we assign according features to the individual if we get a head. Consider we have two individuals, then it is straight forward to see

$$\mathbb{P}(F_2 = \{\{1\},\{1\}\}) = p_1(1-p_1)p_2(1-p_2) \neq 2p_1(1-p_1)p_2(1-p_2) = \mathbb{P}(F_2 = \{\{1\},\{2\}\}). \quad (1.6)$$

That is, feature counts are "insufficient statistics" for LFMs. This is in contrast to the random partition case, given EPPFs. ■

Example 3 demonstrates that we should take extra care when considering LFMs. In the next two section, we present two existing perspectives on modelling LFMs. Unfortunately, none of them offers a satisfactory solution for correlated LFMs, LFMs that explicitly model feature correlations.

1.2.1 LFMs as random partitions

A natural way to resolve the above problem is to rephrase LFMs as random partitions and use existing random partition theory to derive representations of LFMs. Consider latent feature models as exactly random partition models by clustering according to the feature assignments. Let $\pi(\cdot)$ be a projection from feature allocations to partitions, for which two individuals belong to a single cluster if and only if they share a same feature allocation scheme. For example, suppose $f_6 = \{\{3,5\},\{1,2,3\},\{6\}\}$, then $\pi(f_6) = \{\{1,2\},\{3\},\{4\},\{5\},\{6\}\}$. It is straightforward to see that this mapping preserves individual exchangeability and finiteness.

Proposition 1. *If a latent feature model F_N satisfies (IE) and (F), so does its induced random partition $\pi(F_N)$.*

A natural thought is to derive a projective limit of $(\pi(F_N))_{N\in\mathbb{N}}$. But this totally eliminate the *additional structures* that LFMs provide. For example, let $f_6' = \{\{3\},\{4\},\{5\},\{6\}\}$. Then $\pi(f_6) = \pi(f_6')$. However, the feature assignment of f_6 and f_6' are quite different. Besides, the limit frequency provided by Theorem 1 is totally not interpretable, since there is no information among the frequencies telling us the feature sharing assignments. These are the key reasons why we want to work with random objects that are more complicated than random partitions.

1.2.2 LFMs with randomly permuted features

To avoid over-simplification, one natural way to study LFMs is by re-parameterization. That is, to introduce alternative random objects $(\tilde{F}_N)_{N\in\mathbb{N}}$ that can be mapped to LFMs $(F_N)_{N\in\mathbb{N}}$. The merit of re-parameterization is to exploit nice properties of $(\tilde{F}_N)_{N\in\mathbb{N}}$ in order to derive nice representations and/or enable fast posterior inference. In this section we will follow [BPJ+13] to re-parameterize LFMs as a random sequence of subsets of $[N]$ by uniformly permuting elements of F_N. The reason for uniform permutation is to compensate for the additional combinatorial factors introduced by in-distinguishable features, as shown in Example 3. Representations similar as EPPF can thus be introduced for the resulting random sequences. Formally, for each $N \in \mathbb{N}$ assume $\tilde{F}_N \coloneqq \sigma(F_N)$, where $\sigma(F_N)$ is a uniform random permutation over subsets of (F_N) given a pre-defined ordering of the subsets (e.g. by the order of appearance). Note that this re-parameterization does not break down the exchangeability among individuals, since the permutation is uniform over features. BPJ+13 considers the following representation for $\tilde{F}_N$:

Example 4 (Exchangeable feature probability functions (EFPF))**.** A uniformly permuted random feature allocation $\tilde{F}_N$ admits an EFPF representation $p(N,\cdot)$, where $p(N,\cdot)$ is symmetric over its second to the last arguments, if

$$\mathbb{P}(\tilde{F}_N = (A_1,\ldots,A_K)) = p(N,|A_1|,\ldots,|A_K|). \tag{1.7}$$

An LFM $(F_N)_{N\in\mathbb{N}}$ is an EFPF model if for each $N \in \mathbb{N}$, $\tilde{F}_N = \sigma(F_N)$ admits an EFPF representation.■

Remark 1. For EFPF models, note that

i. One key difference between EFPF and EPPF is that EFPF should also take the total number of individuals N into account, while for EPPF the total number of individuals N is determined by summation of all entries for $p(\cdot)$.

ii. In contrast to EPPF, symmetry in EFPF $p(N, \cdot)$ is automatic from the definition in $\tilde{F}_N$.

iii. There are implicit constraints in $p(N, \cdot)$. For example, is it certainly not possible that $p(2, 2, 2) = p(2, 1, 1) = 1$. However, for certain purposes (e.g. deriving a consistent limit) there's no need to explicitly enumerate those constraints. ■

It turns out that EFPF captures several useful LFMs that was previously introduced in the machine learning literature. The most famous examples are the Indian buffet process (IBP) [GG06, GG11] and its generalizations [TG09], which form the foundation of Bayesian non-parametric latent factor analysis. EFPF models are popular because we are able to derive limit frequencies for each feature, and the LFMs can be constructed by first sample the limit frequencies and then each individual sample its allocated features independently given the limit frequencies. The first such kind of well-known result is the truth that IBP can be constructed by first sample a Beta process and each individual independently sample a Bernoulli point process given the Beta process [TJ07]. In this case the feature allocation is represented by point processes, where each point represent an individual feature. Moreover, in several cases the limit frequencies can be constructed by (conditional) independent random variables, which enables fast empirical inference methods such as variational inference [DMVGT09, PZW+10, PCB11, BJP+12, PJ16]. BPJ+13 presents a detailed discussion of those models.

Example 5 (Feature frequency models (FFM))**.** Let (V_k) be a sequence of random variables with values in $[0, 1]$ such that $\sum_{k=1}^{\infty} V_k < \infty$ almost surely. Let $\phi_k \sim_{iid} U[0, 1]$ and independent of (V_k) and $B = \sum_{k=1}^{\infty} V_k \delta_{\phi_k}$. For each data point n, independently draw its feature assignment as a point process $G_n = \sum_{k=1}^{\infty} Z_{n,k} \delta_{\phi_k}$ where Z_{nk} are i.i.d. Bernoulli draws with probability V_k. The feature allocation is induced by the points in G_n. ■

Similar as Theorem 1, there is a one-to-one correspondence between an EFPF and an FFM.

Theorem 2 [BPJ+13]. *Let λ be a non-negative random variable. We can obtain an exchangeable feature allocation by generating a feature allocation from a feature frequency model and then, for each index n, including an independent $Poisson(\lambda)$-distributed number of features of the form $\{n\}$ in addition to those features previously generated. A feature allocation of this type has an EFPF. Conversely, every feature allocation with an EFPF has the same distribution as one generated by this construction for some joint distribution of λ and the feature frequencies.*

Thus, EFPFs or FFMs are nice equivalent sub-families of LFMs. More discussions from Bayesian inference perspectives can be found in [HR15, J+17], for example. Other generalizations [ZHDC12,

J+17] and applications such as topic modelling [WWHB10] and dictionary learning [ZCP+09] of FFMs are extensively studied in the machine learning and statistics literature.

1.3 Correlated LFMs

The key restriction of FFMs is is the conditional independency assumption among $(Z_{nk})_{k\in\mathbb{N}}$ given $(V_k)_{k\in\mathbb{N}}$. Consider the following example, also introduced in [BPJ+13].

Example 6 (An LFM with two correlated features)**.** Consider an LFM with two distinguishable features. An individual has only four feature allocation cases with probability $p_{00}, p_{01}, p_{10}, p_{11}$, where for example, p_{01} is the probability for allocating only the second feature to an individual. It is straight forward to see the only constraint here is $p_{00} + p_{01} + p_{10} + p_{11}$, which does not guarantee an FFM. Actually, an FFM should in addition satisfy $p_{00}p_{11} = p_{01}p_{10}$.

Thus, FFMs is a subset in LFMs that ignores modelling feature correlations. In practice, researchers found that modelling structured correlations among features using LFMs can be useful for data organization, visualization, and prediction. For example, GJTB04 introduces nested Chinese restaurant processes (nCRP), a non-parametric model that constructs a tree hierarchical structure among all the features. Each individual is allowed to pick one path from root node down to a leaf node. The correlations among features is clear: features close to the root have larger change to be picked, and thus should contain more generally shared information. For example, in topic modelling each feature is a topic, a distribution over words in the vocabulary. The topics that are closer to the root are the ones that focus on more common words. Extensions such as nested hierarchical Dirichlet processes [PWBJ15b], allows each individual pick multiple paths down the tree.

In the following sections, we will build theory for flexible correlated LFMs that requires (1) reparameterize LFMs as alternative random objects ξ, (2) use exchangeability assumptions on ξ, and (3) automatically derive the functional form of ξ. We will see two examples. In Section 2 and Section 3, we reparameterize LFMs as an infinite sequence with exchangeable blocks. Using representation theorem, we find that our model is a mixture of Markov chains. In Section 4, we reparameterize LFMs as a separately exchangeable 2d discrete random measure. And we will figure out our model as a composition of random functions with Poisson processes.

Part II

Composing Deep Learning and Bayesian Nonparametric Methods in Latent Feature Models

Chapter 2

Markov Mixed Membership Models

In this Chapter, we present a practical parametric model called Markov mixed membership model (Markov M3) that reparameterize LFMs containing K latent features (topics as features in this case) as a sequence over $[K]$ that sequentially select topics for each document. In particular, Markov M3 models sequences as a mixture of Markov chains. Thus learns a fully connected graph (transition probability matrix) structure among mixing components. The Markov structure results in a simple parametric model that can learn a complex dependency structure between nodes, while still maintaining full conjugacy for closed-form stochastic variational inference. Empirical results demonstrate that Markov M3 performs well compared with tree structured topic models, and can learn meaningful dependency structure between topics.

Mixed membership modeling is a statistical framework for modeling grouped data where each group is represented as a unique mixture over a shared structure [ABEF14]. A wide range of data fall within the scope of mixed membership models, including documents [BNJ03a], images [LP05], and the genome [PSRD00]. Exchangeability assumptions can be relaxed to extend mixed membership models to link data [ABFX08], heterogeneous data [CB09, WB11] and matrix factorization [MWJ10]. In this chapter, we focus on the case where each group's data is assumed i.i.d. given its mixed membership mixing measure.

For discrete grouped data, the most basic mixed membership model is latent Dirichlet allocaiton (LDA) [BNJ03a], which assumes a finite set of discrete distributions, and models each group as a mixture over these distributions using a Dirichlet prior. The simple "flat" Dirichlet prior assumes no structure among the atoms, and so the model can overfit as the number of components increases

beyond a certain number. To capture finer resolution without overfitting, structure has been introduced to the atom relationships, for example by modeling pairwise correlations [BL07] or tree structures [BGJ10a].

Among these models, the tree-structured model is especially interesting for the structure it can learn [BGJ10a, LZZC12, KKKO12, AHS13, PWBJ15a]. Because the components are given a strict parent/child relationship, tree models can discover components of different granularities having top-down dependencies. In topic modeling, this is natural since topics can be more or less specific and the children of one topic can further specify the more general content of the parent topic that unites them.

To consider two Bayesian nonparametric instances, the nested Chinese restaurant process (nCRP) [BGJ10a] and nested hierarchical Dirichlet process (nHDP) [PWBJ15a] are two tree-structured models that select distributions on paths from a root node (see Figure 2.1). For example, the nCRP selects the atoms for a group by following a path from root to leaf node; the nHDP generalizes this by selecting a subtree of atoms for each group. Still, in both models it is assumed that there is a clear tree-structured relationship among nodes. In this chapter, we explore a related modeling framework that allows for a more flexible range of node connections by assuming a Markov structure among the nodes.

To this end, we present the Markov mixed membership model (Markov M3) for grouped data that learns a fully connected graph structure among mixing components. With respect to tree-structured models, our proposed Markov model is straightforward in that, rather than imposing a tree-structured transition rule between nodes, we model the nodes as a fully connected graph with a first-order Markov rule for transitioning between nodes (see Figure 2.1). We therefore refer to this as a *graph-based* mixed membership model. In the context of topic models, this means that any topic can *a priori* transition to any other topic, but using a sparse Dirichlet prior on transition distributions we learn a meaningful dependence between topics through posterior inference.

An advantage of our proposed framework is that it avoids the combinatorial issues encountered during inference by models such as the nCRP [WB09], and does not require complicated pruning procedures such as required by the nHDP [PWBJ15a]. In contrast, Markov M3 is a fully conjugate-exponential family model that allows for closed-form updates that doesn't require the complicated procedures of the nCRP or nHDP. As with those two models, our model is easily learned with stochastic variational inference allowing for processing of large data sets [HBWP13a].

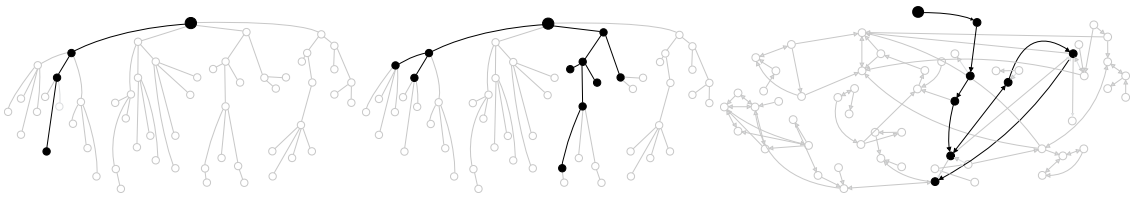


Figure 2.1: Comparison between three path-based models. (Left) The tree-structured nested Chinese restaurant process (nCRP) selects one path per group; (Mid.) the tree-based nested hierarchical Dirichlet process (nHDP) places high probability on a subtree for each group; (Right) the proposed graph-based Markov mixed membership model selects one path per group using a Markov random walk on the fully connected set of nodes (an example high-probability connectivity is depicted in the background here).

2.1 Model Description

Mixed membership models are applicable to data sets where the observations are grouped, i.e., where viewing the data on the instance-level results in subsets of the data. For example, each document in a set can be represented as a group of words. Mixed membership models assume that the groups share a data generating (global) structure, but with different distributions to account for group-level (local) differences. For example, topic models share the same set of distributions on words, but mix over them differently for each group.

A common assumption made by mixed membership models is that the data is exchangeable within and across groups. In this case, where there exists a random mixed membership measure (i.e., De-finitte's measure) that results in i.i.d. generation of the data. LDA is an example of an exchangeable mixed membership model both within and across groups, whereas dynamic topic models assume partial exchangeability since the order of documents matters [BL06]. We present a fully exchangeable model in which each group mixes on paths selected from a fully connected graph according to a Markov random walk.

2.1.1 Markov mixed membership models

We define the generative process for the Markov mixed membership model. The following procedure is also summarized in Algorithm 1. Let w_d be the set of data for group d. We model this data as an i.i.d. set drawn from a mixture G_d with mixing distribution $\nu_d \sim \text{GEM}(\gamma_0)$ and a *group-specific* sequence of atoms $(\hat{\beta}_1, \hat{\beta}_2, \dots)$. We can draw from the GEM stick-breaking distribution by sampling,

$$u_{di} \overset{iid}{\sim} \text{Beta}(1, \gamma_0), \quad \nu_{di} = u_{di} \prod_{j=1}^{i-1} (1 - u_{di}). \tag{2.1}$$

We use the mixture $G_d = \sum_{i=1}^{\infty} \nu_{di} \delta_{\hat{\beta}_i}$ to generate group $w_d = (w_{d1}, \ldots, w_{dn})$ by sampling

$$\ell_{dn} \sim \text{Discrete}(\nu_d), \quad w_{dn}|\ell_{dn} \sim f(\hat{\beta}_{\ell_{dn}}). \tag{2.2}$$

The distribution f is problem-specific. In this chapter we take f to be a discrete distribution and $\hat{\beta}$ a V-dimensional probability vector. We assume there are K atoms, where $\hat{\beta}_i \in \boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$, and define the prior

$$\beta_k \overset{iid}{\sim} \text{Dir}(\beta_0 \mathbf{1}_V). \tag{2.3}$$

In the nCRP [BGJ10a], the sequence of $\hat{\beta}_i$ is selected by following a path from the root node to the leaf node of a tree. Instead, we assume a first order Markov structure for this sequence. We construct a Markov transition distribution on $\boldsymbol{\beta}$ by drawing

$$\theta_k \overset{iid}{\sim} \text{Dir}(\alpha_0/K, \ldots, \alpha_0/K) \tag{2.4}$$

for each k. The distribution on the sequence $\hat{\boldsymbol{\beta}}_d$ is then $P(\hat{\beta}_i = \beta_{j'}|\hat{\beta}_{i-1} = \beta_j|\boldsymbol{\theta}) = \theta_{j,j'}$. The variable $\mathbf{z}_d$ shown in Algorithm 1 and used for inference indexes this sequence: $z_{di} = j'$ if $\hat{\beta}_i = \beta_{j'}$. We assume the same Dirichlet prior on the initial state distribution π as well.

We observe that if we were to set $\ell_{dn} = n$, we would assign w_{dn} to atom $\hat{\beta}_n$ and the result would be a standard hidden Markov model (HMM). What differentiates our model, and reintroduces group-level exchangeability, is that each word chooses which of the selected states it belongs to i.i.d. ν_d. The analogy to the HMM can be pursued by thinking of the words of a document first being partitioned into ordered sets according to ν_d, and then drawing each group according to an HMM. We will see how this way of considering the model leads to variational inference that builds on inference for the HMM [Bea03].

2.1.2 Relationship to tree-structured models

As Figure 2.1 illustrates, the major different between graph-based and tree-based mixed membership models is the dependence structure between nodes. Where models such as the nCRP impose a strict parent/child hierarchy, Markov M3 in a sense captures the potential for each node to be the parent of all others (which simply results from a shift of perspective about Markov chains). The limited modeling ability of the nCRP is primarily due to the rigid single path allowed per group. In topic modeling, this forces each selected topic to be a strict subset of those previously selected. This assumption was

relaxed by recent tree-structured alternatives [KKKO12, AHS13, PWBJ15a]. For example, as Figure 2.1 indicates, the nHDP allows for multiple paths per document, so two general topics can be combined in a single document by allowing words to select paths in different directions.

As seen in Figure 2.1, Markov M3 in one sense returns to the one path per group structure of the nCRP, but allows for exploration of the entire space like the nHDP. This provides another remedy to the rigidness of the nCRP. It also offers modeling capabilities not found in the nHDP, since in that model, the presence of two subtrees is not causally linked according to the prior—the presence of one subtree says nothing about the presence of another. With Markov M3, there is a causal connection between two atoms according to the Markovian generative structure (which we observe is not symmetric). Markov M3 is again like the nCRP in that, once it selects its path of atoms, it mixes on them using a probability vector drawn from a stick-breaking distribution.

The Markov mixed membership model can be viewed as a possible parametric version of the nested Chinese restaurant process, in that we can show that the nCRP is the limiting process of the model in Algorithm 1 as K goes to infinity. To roughly sketch this, consider the marginal probability measure $G_m^K = \sum_{k=1}^K \theta_{mk}\delta_{\beta_k}$ constructed for the mth node. [IZ02] proved that $\lim_{K\to\infty} G_m^K = G_m \sim \text{DP}(\alpha_0\mu)$. In the limit $K \to \infty$, $\theta_{mk} = 0$ with probability one, while $\sum_k \theta_{mk} = 1$. In this case, the nonzero probability can be shown to be on a disjoint set of atoms for each G_m with probability one, despite the fact that they share atoms in the finite approximation G_m^K. Practically speaking, this means that a state transition sequence sampled from the infinite limit of Markov M3 will never return to the same node twice, and so the model can equivalently be thought of as selecting a path in a tree. We formally state this in the following Proposition.

Proposition 2. *As K goes to infinity, the Markov mixed membership model recovers the underlying mixing measure of the nested Chinese restaurant process.*

2.1.3 Related work

Graph-based mixed membership models have been applied to grouped data in other settings. For example, mixed membership models have been applied to graph data, where instances are linked as a graph, and stochastic block models [ABFX08] have been proposed to model the links between instances through clustering. Mixed membership models have also been applied to collaborative

Algorithm 1 Generative process for Markov M3

Global variables
Draw an initial-state distribution $\pi \sim_{iid} \text{Dir}(\frac{\alpha_0}{K}\mathbf{1}_K)$.
for each atom $k \in \{1, 2, \cdots, K\}$ **do**
1. Draw parameter $\beta_k \sim_{iid} \mu$.
2. Draw transition distribution $\theta_k \sim_{iid} \text{Dir}(\frac{\alpha_0}{K}\mathbf{1}_K)$.

end for
Local variables
for each document $d \in \{1, 2, \cdots, D\}$ **do**
1. Draw a Markov chain of atoms $z_d \sim \text{MC}(\pi, \boldsymbol{\theta})$.
2. Draw a distribution on atoms, $\nu_d \sim \text{GEM}(\gamma_0)$.
for each word n in document d **do**
1. Draw assignment $\ell_{dn} \sim_{iid} \text{Disc}(\nu_d)$.
2. Draw observation $w_{dn} \sim f(\beta_{z_{d,\ell_{dn}}})$.

end for
end for

filtering [MWJ10, WB11] and link prediction [CB09]. Exploring exchangeable structures in graphs has also received recent theoretical interest [OR15a].

We also observe that mixed membership models can be applied to the more traditional use of hidden Markov models for sequential data. For example [Pau14] considers a similarly named process for the fundamentally different problem of nonexchangeable sequence modeling.

2.2 Scalable Variational Inference

We derive a variational inference algorithm for learning an approximate posterior of all model variables shown in Algorithm 1. We can factorize the joint distribution of our model as

$$p(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{u}, \boldsymbol{\ell}, \pi) = p(\boldsymbol{\beta})p(\boldsymbol{\theta})p(\pi) \ \times \prod_d p(\boldsymbol{w}_d|\boldsymbol{\beta}, \mathbf{z}_d, \boldsymbol{\ell}_d)p(\mathbf{z}_d|\boldsymbol{\theta}, \pi)p(\boldsymbol{\ell}_d|\boldsymbol{u}_d)p(\boldsymbol{u}_d). \tag{2.5}$$

We apply mean-field variational inference to approximate the posterior $p(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{u}, \boldsymbol{\ell}|\boldsymbol{w})$ by defining a factorized q distribution on these variables and locally maximizing the variational objective function

$$\mathcal{L} = \mathbb{E}_q[\ln p(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{u}, \boldsymbol{\ell})] - \mathbb{E}_q[\ln q]$$

using coordinate ascent, which approximately minimizes the KL-divergence between the true posterior and q. We restrict q to the following factorized form

$$q(\boldsymbol{\beta}, \boldsymbol{\theta}, \pi, \mathbf{z}, \boldsymbol{u}, \boldsymbol{\ell}) = q(\boldsymbol{\beta})q(\boldsymbol{\theta})q(\pi)q(\mathbf{z})q(\boldsymbol{u})q(\boldsymbol{\ell}), \tag{2.6}$$

which we further factorize as

$$q(\boldsymbol{\beta})q(\boldsymbol{\theta})q(\pi) = q(\pi|\alpha_\pi)\prod_{k=1}^{K} q(\beta_k|\lambda_k)q(\theta_k|\alpha_k),$$
$$q(\mathbf{z})q(\boldsymbol{u}) = \prod_d q(\mathbf{z}_d|\boldsymbol{\varphi}_d)\prod_i q(u_{di}|a_{di}, b_{di}),$$
$$q(\boldsymbol{\ell}) = \prod_d \prod_n q(\ell_{dn}|\phi_{dn}). \tag{2.7}$$

We select all q distributions to be in the same family as the prior defined in Algorithm 1. In the following, we use coordinate ascent to optimize the variational objective, with respect to the variational parameters. We first discuss the update of local variables $\mathbf{z}_d, \boldsymbol{u}_d, \boldsymbol{\ell}_d$, which are only dependent on a single instance. Then we discuss the global variables $\boldsymbol{\beta}, \boldsymbol{\theta}, \pi$ that depends on multiple instances. Though we have proposed a parametric model in the size of the graph, K, we have defined the Markov chain for each group to be infinite in length. For inference, we introduce a truncation of this stick-breaking construction to level T, as is typically done [BJ06]. We note that truncation-free methods are a possible remedy [WB12].

2.2.1 Local variables

The most complex part of inference is in learning the Markov sequence $\mathbf{z}_d$ that selects atoms for group d. We can derive the explicit form of its posterior from

$$q(\mathbf{z}_d) \propto \exp\Big(\sum_i \underbrace{\mathbb{E}[\ln p(z_{di}|z_{d,i-1}, \boldsymbol{\theta})]}_{pairwise\ potential} + \sum_i \underbrace{\sum_n \phi_{dn}(i)\mathbb{E}[\ln p(\mathbf{w}_d|\boldsymbol{\ell}_d, z_{di}, \boldsymbol{\beta})]}_{single\ potential}\Big). \tag{2.8}$$

In Eq. (2.8) we can break $q(\mathbf{z}_d)$ into single potentials and pairwise potentials as indicated. We can then solve using the forward-backward algorithm to infer the posterior joint marginals. In fact, this is exactly the procedure for learning the state transitions of an HMM with the important difference that the emission at step i is not predefined, but instead the soft clustering of data in $\mathbf{w}_d$ induced by the variational parameters of $\boldsymbol{\ell}_d$, which is changing with each iteration. A simple modification can be made to the forward-backward algorithm that accounts for the new emission process (please see the supplemental material).

Let $\chi^f_{di}(k)$ and $\chi^b_{di}(k)$ be the outputs of the forward and backward algorithms, respectively. As

with the HMM, we will use the quantities

$$\varphi_{di}(k) \propto \chi^f_{di}(k)\chi^b_{di}(k) \tag{2.9}$$

$$\xi_{di}(k,k') \propto \chi^f_{d,i}(k)\chi^b_{d,i+1}(k')\kappa_{di}(k,k'), \tag{2.10}$$

where we define

$$\kappa_{di}(k,k') = \exp\{\mathbb{E}[\ln\theta_{k,k'}] + \textstyle\sum_n \phi_{dn}(k')\mathbb{E}[\ln\beta_{k',w_{dn}}]\}.$$

Normalization is over k for φ and the pair (k,k') for ξ. The multinomial variational parameter ϕ_{dn} corresponds to the data allocation variable ℓ_{dn}, found by calculating

$$\phi_{dn}(i) \propto \exp\Big(\mathbb{E}[\ln\nu_{di}] + \textstyle\sum_k \varphi_{di}(k)\mathbb{E}[\ln\beta_{k,w_{dn}}]\Big). \tag{2.11}$$

We observe that the last term sums over atom assignments for the ith value in the Markov sequence. The first expectation is from the stick-break construction

$$\mathbb{E}[\ln\nu_{di}] = \mathbb{E}[\ln u_{di}] + \textstyle\sum_{j=1}^{i-1}\mathbb{E}[\ln(1-u_{dj})] \tag{2.12}$$

The final local variables are the stick-breaking proportions $\mathbf{u}_d$. Given the allocation distributions $q(\ell_{dn})$, the update of the beta q distribution of u_{di} is

$$a_{di} = 1 + \sum_n \phi_{dn}(i), \quad b_{di} = \gamma_0 + \sum_{n,i'>i}\phi_{dn}(i'). \tag{2.13}$$

We can iterate several times updating the local variables for each document before moving on to the global variables.

2.2.2 Global variables

The global variables include the Markov transition probabilities and the atoms. The update to the variational parameters of the initial state and transition probabilities, π and $\boldsymbol{\theta}$, are identical to the HMM,

$$\alpha_{\pi,k} = \frac{\alpha_0}{K} + \sum_d \varphi_{d1}(k), \tag{2.14}$$

$$\alpha_{k,k'} = \frac{\alpha_0}{K} + \sum_d\sum_i \xi_{di}(k,k'). \tag{2.15}$$

Algorithm 2 An outline of batch variational inference

Local variables: For each document d,	
Update $q(\mathbf{z}_d)$ with forward-backward.	Eq. (2.8)
Update each word allocation $q(\ell_{dn})$.	Eq. (2.11)
Update stick proportions $q(u_{di})$.	Eq. (2.13)
Global variables: For π and each atom k,	
Update the initial state distribution $q(\pi)$.	Eq. (2.14)
Update the transition distribution $q(\theta_k)$.	Eq. (2.15)
Update the atom distribution $q(\beta_k)$.	Eq. (2.16)

For Dirichlet-distributed atoms $\boldsymbol{\beta}$, the variational update to the Dirichlet q distribution of β_k is

$$\lambda_{kv} = \beta_0 + \sum_d \sum_n \mathbb{1}(w_{dn} = v) \sum_i \phi_{dn}(i)\varphi_{di}(k). \tag{2.16}$$

The right-most summation calculates the probability that word w_{dn} and topic β_k are both be assigned to the same point in the Markov sequence $(z_{d1}, z_{d2}, \dots)$, which is required for word w_{dn} to belong to component β_k.

2.2.3 Stochastic variational inference

Since Markov M3 is a conjugate-exponential family model, it is immediately amenable to stochastic variational inference (SVI) [HBWP13a]. For models such as tree-based models and the proposed graph-based model, such large data extensions can help in learning the greater level of structure defined by the model prior. As with other mixed membership models, we can exploit the fact that the variational objective function factorizes as

$$\begin{aligned} \mathcal{L} \;=\; & -\mathbb{E}_q[\ln q] + \mathbb{E}_q[\ln p(\boldsymbol{\beta}, \boldsymbol{\theta}, \pi)] \\ & + \textstyle\sum_d \mathbb{E}_q[\ln p(\boldsymbol{w}_d, \mathbf{z}_d, \boldsymbol{u}_d, \boldsymbol{\ell}_d | \boldsymbol{\beta}, \boldsymbol{\theta}, \pi)]. \end{aligned} \tag{2.17}$$

Using SVI, we stochastically optimize $\mathcal{L}$ by restricting the local calculations to a small subset C_t of the D groups at iteration t. Given the subset C_t, SVI proceeds by (1) optimizing all local variables indexed by C_t, (2) forming the global updates restricted to C_t, and (3) averaging these updates with the current values. Let $\hat{\alpha}_{\pi,k}$, $\hat{\alpha}_{k,k'}$ and $\hat{\lambda}_{kv}$ be the coordinate ascent updates restricted to C_t. These are calculated as in Eq. (2.14)–(2.16). Then the stochastic updates to the true values are

$$\begin{aligned} \alpha_{\pi,k}^{(t+1)} &= (1-\rho_t)\alpha_{\pi,k}^{(t)} + \rho_t(D/|C_t|)\hat{\alpha}_{\pi,k} \\ \alpha_{k,k'}^{(t+1)} &= (1-\rho_t)\alpha_{k,k'}^{(t)} + \rho_t(D/|C_t|)\hat{\alpha}_{k,k'} \end{aligned}$$

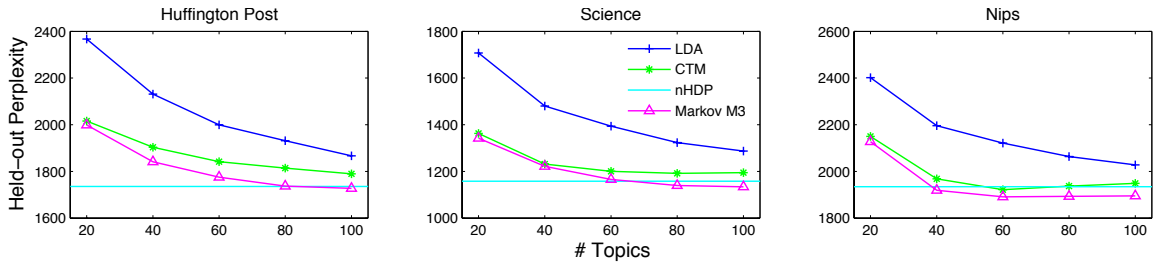


Figure 2.2: Held-out perplexity results. The Markov transition model (Markov M3) overall achieves best performance among parametric models. Its best performance is even better than the state-of-the-art nonparametric nHDP.

Table 2.1: Three datasets used for batch comparison.

Corpus	# train	# test	# vocab	# tokens
Huff Post	3.5K	589	6,313	907K
Science	4K	1K	4,403	1.39M
Nips	2.2K	300	14,086	3.3M

$$\lambda_{kv}^{(t+1)} \quad = \quad (1-\rho_t)\lambda_{kv}^{(t)} + \rho_t(D/|C_t|)\hat{\lambda}_{kv} \tag{2.18}$$

The decaying learning rate ρ_t must satisfy $\sum_{t=1}^{\infty}\rho_t = \infty$, $\sum_{t=1}^{\infty}\rho_t^2 < \infty$ to ensure convergence [Bot98]. We set $\rho_t = (\tau_0 + t)^{-\kappa}$, where $\tau_0 > 0$ and $0.5 < \kappa \leq 1$.

2.3 Experiments

Our experiments with Markov M3 focus on grouped discrete data problems. We first consider topic modeling on small and large scale problems. We then show qualitative results on a music tagging problem, where the union of quantized song features and user tags provides the discrete grouping on a song level.

2.3.1 Document modeling

Batch comparisons. We first apply our model to three datasets easily learned with batch inference: *Huffington Post*, *Science* and *NIPS* papers. The statistics from each data set is shown in Table 2.1. We split each data set into a training set and a test set, also shown in Table 2.1. For the testing set we split each document into a $90/10$ split and learned local parameters on 90% of words in the document and predicted 10% for prediction given the inferred topic proportions. We present the quantitative

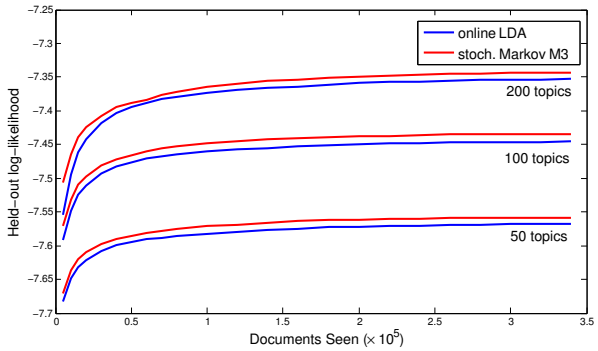


Figure 2.3: Predictive performance for online Markov M3 and online LDA. Markov M3 is consistantly better for various number of topics through the entire learning process.

performance using preplexity, which can be calculated as

$$\text{perplexity} = \exp\Big\{ -\frac{\sum_{n \in w_{TS}} \log p(w_n | w_{TR})}{|w_{TS}|} \Big\}, \tag{2.19}$$

where w_{TR}, w_{TS} represent training and test words in the test set respectively.

We compare performance of our model with LDA, the correlated topic model (CTM) [BL07] and the nested HDP (nHDP) [PWBJ15a]. The nHDP was shown to give better predictive ability than the nCRP, and so we do not compare with that algorithm. We have also noted that when K goes to infinity, the Markov M3 recovers the nCRP mixed membership model and so performance of our model tends to the nCRP.

The perplexity results using a different number of topic are shown in Figure 2.2. For the nonparametric nHDP we truncate its posterior topic number to 175, which is higher than the maximum number of topics used by the parametric models. On all datasets, the Markov M3 consistently performs better than other parametric models. The reason is that Markov M3 can more flexibly model all pairwise topic dependencies, while LDA only considers there to be a slight negative correlation among topics, and CTM considers pairwise correlation without dependency information. We observe that the best performance for Markov M3 is better than nHDP, which gives evidence that a graph structure is preferable to a tree structure for modeling topic dependencies.

Stochastic learning. For a large-scale problem, we train our model using stochastic variational inference on the *New York Times* dataset, which contains 1.8 million documents, and compare its predictive performance on a held-out test set with online LDA [HBB10]. For both models we use the same topic initialization. We also use a learning rate of $(10 + t)^{-0.75}$ for both models, and a batch size of $|C_t| = 500$. For Markov M3, we truncate the path length to 15 and set $\gamma_0 = 1$.

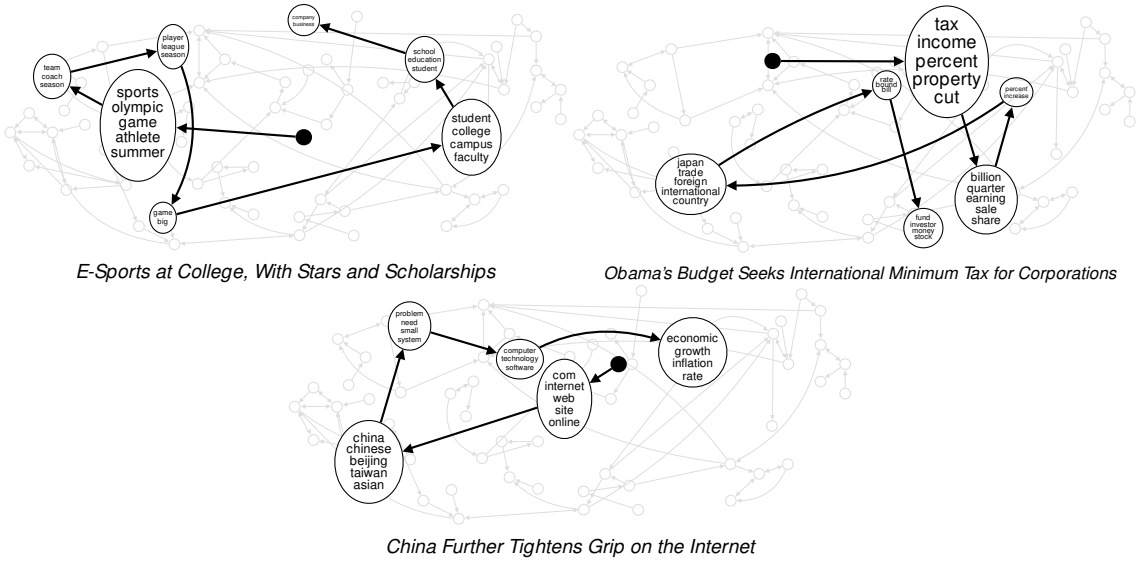


Figure 2.4: Topic paths selected by three documents. The size of the node indicates the proportion of the topic.

In Figure 2.3, we show the predictive performance throughout the learning process of both models, considering various number of topics. We see that the stochastic version of Markov M3 performs better than online LDA, which is consistent with the results on smaller scale problems.

Markov M3 learns a transition distribution over all topics. In Fig. 2.5, we depict the most probable transitions from several topics within the graph. We limit connections and directions to those above a threshold for clarity and use size to roughly indicate probability. As can be seen, most transition probabilities between topics are very low, thus are not displayed on the graph. Among all the topics, a few of them are general (e.g., topic 67 with top words 'say', 'life', 'man'), but most topics are more specific. Topics naturally form small cliques, where the transition probability within a cluster is significantly higher than transitions across clusters. There are also connections between topics in different but similar domains. For example, there is a high transition probability between a "film" topic (18) to a "music" topic (46).

We further illustrate the Markov property by focusing on the document level. Since each document learns a distribution on paths through the graph, we show the most probable path found using the Viterbi algorithm. We show the paths selected for three documents in Figure 2.4, where the arrow indicates path direction and the size of the node indicates the proportion for that topic. In general, Markov M3 tends to visit earlier topics with larger proportions, as is encouraged by the stick-breaking prior. The topics selected in these examples are clearly interpretable, and the sequence captures

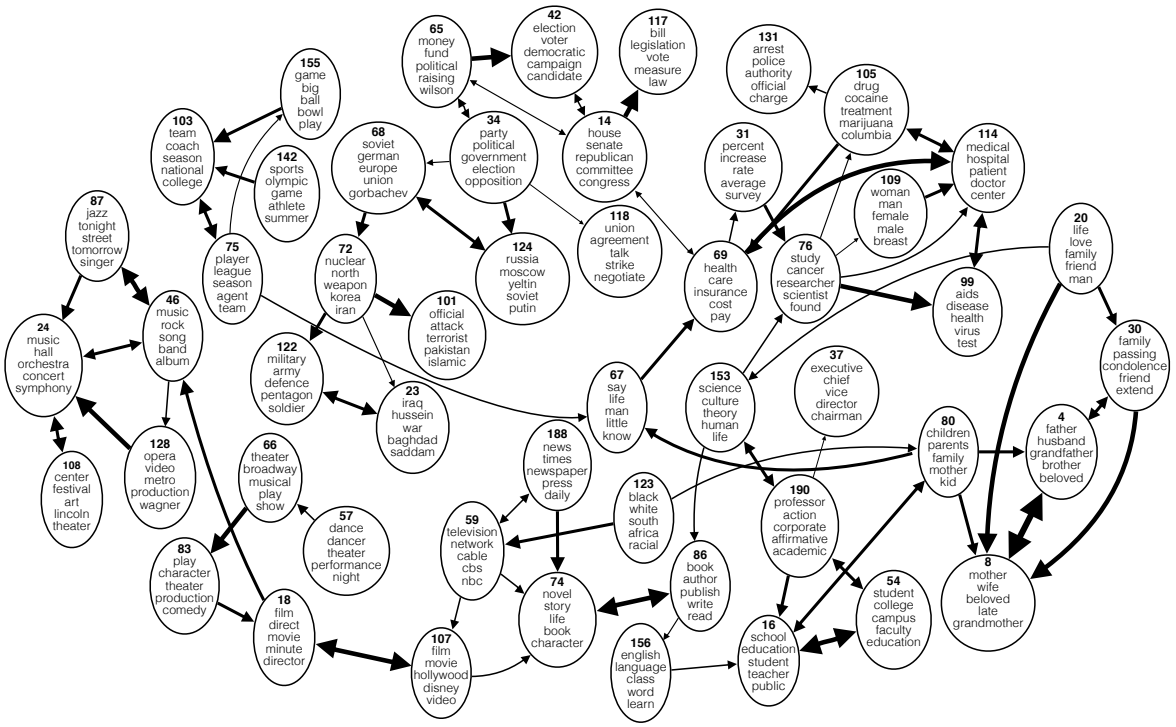


Figure 2.5: Selected topic subgraph from a 200-topic graph learned by online Markov M3 on New York Times dataset. The graph shows Markov transitions with high probability among topics.

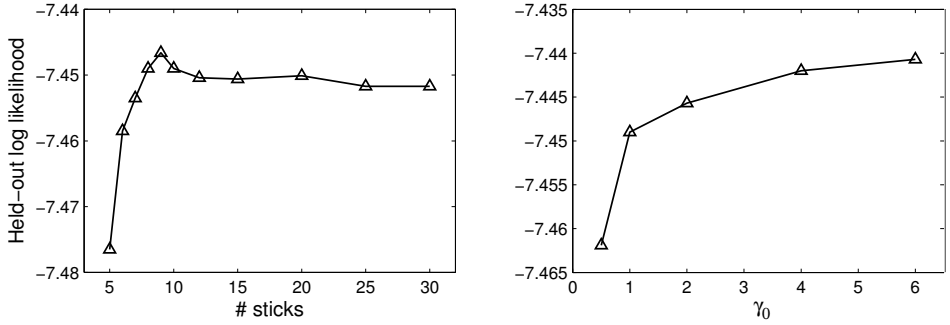
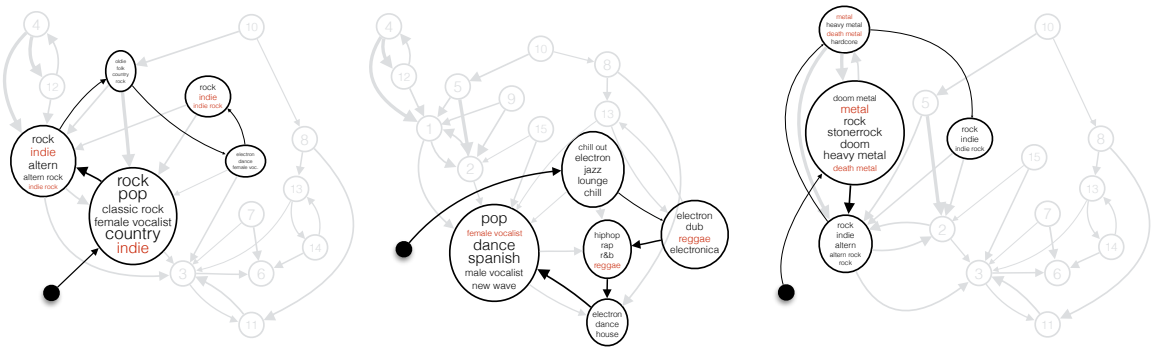


Figure 2.6: Sensitivity analysis for (left) the truncation level of sticks, and (right) the stick-breaking concentration parameter. Results are shown in terms of average log likelihood on a test set.

information about topics relations.

Sensitivity analysis. We empirically analyze the effect of the truncation level of the Markov chain and the stick-breaking concentration parameter in our model. The truncation level indicates the number of topics we allow to each document. When the truncation level is too small, each document can only explore very limited number topics.[1] As Figure 2.6 shows, the model is less accurate in this

[1]We recall that the nCRP restricted each document to 3 topics, not counting the shared root topic.



The Bristols, "Little Baby"
(indie, indie rock)

Carroll Thompson, "I'm So Sorry"
(female vocalist, reggae)

Akercocke, "Footsteps Resound In An Empty Chapel"
(black metal, death metal, metal)

Figure 2.7: Markov transition paths learned from three songs without knowing their tags (the ground truth tags are marked in the parenthesis below). The black node denotes the initial state. The size of nodes indicates the proportion for that factor, and the thickness of arrows indicates the transition probability between factors. All the paths are embedded in the graph (the gray background).

case. However, when the length of Markov random walk become too large, performance also decreases. This is because the update to the transition distributions θ_k treat the entire sequence of each document equally. If there are many empty topics in the Markov sequence, as indicated by the q distributions on u_{di}, the information in the q distributions of the transition matrix can become less informative. In this sense, truncation of the model is important and should be set so that most available topics are used by a document. The concentration parameter γ_0 defines the smoothness for the stick-breaking proportions. Figure 2.6 shows that choosing a smooth prior can help improve performance.

2.3.2 Million song dataset

We also experiment with Million Song Dataset [BMEWL11]. We first extract music audio features from 371K songs and learn a codebook of size 512 using K-means. We represent each song as a vector that can be split into two parts. The first part contains the vector-quantized audio features using the codebook. This gives a vector $\mathbf{w} \in \mathbb{N}^J$, where w_k represents the counts of audio frames that fall into cluster k for a particular song. The remaining part is a user-applied bag-of-tags $\mathbf{v} \in \{0,1\}^L$, with a total of $L = 561$ tags. We set $v_l = 1$ if tag l is observed for the given song. Thus, the entire quantized feature can be represented as $[\mathbf{w}, \mathbf{v}]$, or a document with a vocabulary size of 1,073.

Exploiting latent factors that generates audio waveforms for tagging has been studied in recent years [HBC10, LHE13]. The end goal we consider is the problem of assigning semantic tags to a song by only analyzing its audio waveform using a model learned from the weakly labeled songs (i.e.,

incomplete and noisy labeled). We apply Markov M3 with 50 nodes (topics) to this problem to learn joint audio-tag topics–each factor is a distribution over "words", which is a combination of the audio codebook and the tags. In our problem set-up, we note that the audio features dominate entire feature since the number of user-applied tags is much smaller than the number of quantized audio features, and so the topics learned will be audio-centered and the marginal tag distribution can be viewed as a weak semantic label of music style captured by that audio factor.

As with document modeling we learns a fully connected graph over music factors, which we display here by showing the path transitions for three held-out test songs. Again, the model learns a distribution on paths, and we only show the most probable path using Viterbi. For this testing problem, since we don't have any tags, we feed the quantized audio features into our model and to learn their Markov transitions over factors. We then use the tagging portion of the selected atoms to represent the path selected. From Figure 2.7, we find that Markov M3 pulls out the correct, but noisy tags (marked as red) with high probability, and also discovers other possibly relevant tags.

In this chapter, we proposed a Markov mixed membership model (Markov M3) that explores a fully connected graph structure among components. Markov M3 provides a new way of performing mixed membership modeling with structured distributions, and an alternative to similar tree-based models. We showed how Markov M3 gives a new perspective on the nCRP by showing nCRP to be a limiting case of Markov M3. We showed the effectiveness of this modeling framework on small and large datasets for discrete grouped data such as documents and quantized music. In the next chapter, we introduce an infinite dimensional view of Markov M3 derived from a sequential "feature allocation" scheme.

Chapter 3

Markov Latent Feature Models

In this chapter, we formalize the idea of modeling LFMs using sequences with a *representation theorem*. The key idea is to interpret each state of a sequential process as corresponding to a latent feature, and the set of states visited between two null-state visits as picking out features for an observation. We call a subsequence between two null-states as a "block". We show that, given exchangeability over blocks, we can represent the sequence as a mixture of recurrent Markov chains. In this way we can perform *correlated latent feature modeling* for the sparse coding problem. We demonstrate two cases in which we define finite and infinite latent feature models constructed from first-order Markov chains, and derive their associated scalable inference algorithms. We show empirical results on a genome analysis task and an image denoising task.

Latent feature models learn the unobserved factors that are shared among a collection of objects. Often a small fraction of these latent features can be used to jointly describe, or "sparsely code," a single object. This assumption is often made for a wide range of tasks [GSG07]. For example, each DNA sequence can be assigned features that measure repeating patterns of certain base pairs, or each image can be assigned features that correspond to the items it contains.

The process for making latent feature assignments can be thought of as generating a 0-1 matrix where the 1-elements in each row index the latent features assigned to the object. For example, the Indian buffet process (IBP) [GG11] defines a feature allocation whereby features are independently assigned according to a rich-get-richer scheme. The IBP is a Bayesian nonparametric model in which the number of latent features can grow to infinity with data. It also assumes exchangeability among objects, meaning the order of the data does not affect the feature assignment probabilities. With such

assumptions there always exists a mixing measure (de Finetti's measure) by which feature allocation scheme for different objects are conditionally independent. For example, the mixing measure for the IBP is the beta process [TJ07]. Employing such representations allows for simple variational inference algorithms that can easily scale to large data sets [HBWP13b, SP15, SKG15].

Models based on the IBP and beta process assume independence among the latent features allocated to an object, but in many cases these latent features may have dependencies. For example, in a natural image a car is more likely to co-occur with a bus rather than a whale. Several methods have been developed for modeling dependencies among latent features or clusters. For example, beta diffusion trees [HKG14] organize latent features in a tree to learn a multi-resolution feature structure. Other tree models, such as the nested Chinese restaurant processes [BGJ10b] and the nested hierarchical Dirichlet processes [PWBJ15b], use a discrete-path based tree structure to select latent features for each object. To avoid the rigid structure of trees in a mixed-membership framework, Markov mixed-membership models [ZP15b] propose instead modeling the pair-wise correlation of latent clusters with a Markov random walk on a fully-connected, finite graph.

We propose a *Markov latent feature model* (MLFM), which extends the idea of using Markov random walks to latent factor modeling problems such as those addressed by the IBP and beta process. The main novelty in this new framework is that we use a sequential block—a subsequence between two adjacent visits to a "null state"—to define the feature allocation process (Section 3.1). Since only a subset of states will be visited prior to returning to the null state, a MLFM is a sparse coding model. We introduce two scalable MLFM models, a parametric and nonparametric version, for which we directly define the mixing measure of the associated recurrent Markov chain (Section 3.2). This allows us to derive a scalable variational inference algorithm (Section 3.3). Finally, we apply MLFM to a genome analysis task and an image denoising task to show its effectiveness (Section 3.4).

3.1 Feature Allocation via Sequences

Before describing the specific generative processes we use, we discuss the central property of the proposed Markov latent feature model (MLFM) that distinguishes it from other approaches to sparse coding. Let $\mathbf{Z} = (Z_0, Z_1, \cdots)$ be an infinitely long stochastic process, where each $Z_i \in \mathbb{N} \cup \{0\}$ and $Z_0 = 0$, with 0 indexing the "null state". As we shall see, the null state plays the role of partitioning latent features for different objects, while $\mathbb{N}$ is a feature index set.

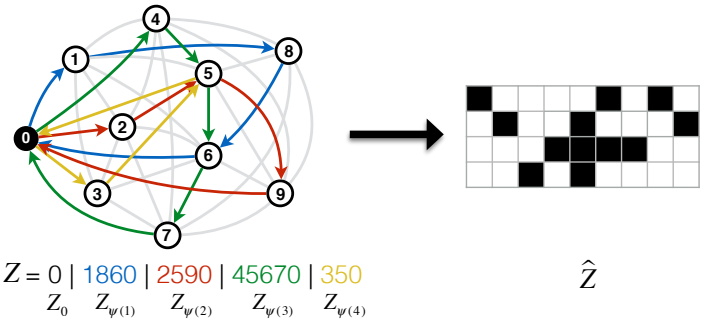


Figure 3.1: An illustration of the construction of a 0-1 matrix from a sequential process, which we define to be a mixture of recurrent Markov chains. On the LHS, the chain $\mathbf{Z}$ starts from a null state $Z_0 = 0$ and generates four blocks (subsequences) $\mathbf{Z}_{\psi(1)}$ to $\mathbf{Z}_{\psi(4)}$ by returning to 0 four times (shown as four colored paths on the graph). One the RHS, this sequence constructs a 0-1 matrix with four rows and the columns indicating the unique set of states visited in each block.

The generative process we define sequentially pick features for an object until the process returns to 0. Let $\tau(0) = 0$ be the index of the initial null state and

$$\tau(1) = \min\{n > \tau(0) \ : \ Z_n = 0\}. \tag{3.1}$$

The sequence through $Z_{\tau(1)}$ selects the features assigned to the first object as the unique set of states visited between $Z_{\tau(0)}$ and $Z_{\tau(1)}$. We call $\tau(1)$ the first return time and define $\psi(1)$ to be the sequence $(\tau(0)+1, \ldots, \tau(1))$. Therefore $\mathbf{Z}_{\psi(1)}$ is the first *block* of the process $\mathbf{Z}$ and corresponds to the features allocated to the first observation.

We continue this procedure through a second return time $\tau(2)$, constructing the second subsequence $\mathbf{Z}_{\psi(2)}$ from which we obtain the set of latent features assigned to the second observation. More generally, if we have N observations, then we read the stochastic process until step $\tau(N)$, the time we finish selecting features for the last object.

We can use this set of blocks to construct a 0-1 matrix $\widehat{\mathbf{Z}}$, where each row indicates the features associated with the corresponding observation similar to the IBP. We show an example for $N = 4$ in Figure 3.1.

Thus far we have not defined the distribution of $\mathbf{Z}$. However, we choose to make the following two restrictions:

1. The null state should be visited infinitely many times.

2. The rows of $\widehat{\mathbf{Z}}$ should be exchangeable.

The first restriction allows us to model an infinite number of observations and is a statement about the

recurrency of $\mathbf{Z}$. The second restriction is made to allow for simple inference using a mixing measure. Our added goal of modeling feature correlations leads us to enforce the second restriction via *Markov exchangeability*. Recall that a sequence $\mathbf{Z}$ is Markov exchangeable if the probability of two sequences $\mathbf{Z}'$ and $\mathbf{Z}''$ is the same when they share a permuted collection of subsequences. In the context of Figure 3.1, this simply states that the probability of $\mathbf{Z}$ is the same according to the chosen distribution if we permute a finite number of $Z_{\psi(i)}$. The following lemma is a direct result of these definitions.

Lemma 1. *The rows of $\widehat{\mathbf{Z}}$ are exchangeable if $\mathbf{Z}$ is Markov exchangeable.*

Theorem 1. *[DF80] A recurrent process $\mathbf{Z}$ is Markov exchangeable if and only if it is a mixture of Markov chains.*

We therefore specify $\mathbf{Z}$ as a mixture of *recurrent Markov chains*. As a result, the latent features are correlated, which can be viewed as a graph where the edges indicate Markov transitions among states (see Figure 3.1). We observe that models such as the IBP satisfy the above requirements, but without modeling correlations. Using the Markov property provides this modeling capacity in a way that allows for simple inference and has straightforward nonparametric extensions.

3.2 Markov Latent Feature Models

In Section 3.1 we proposed restricting $\mathbf{Z}$ to be a mixture of recurrent Markov chains. In principle, any formulation based on this restriction would be valid, including those whose mixing measure is unknown, but for practical purposes we would like to explicitly model the mixing measure of the recurrent Markov chain. We therefore propose two models below, one parametric and one nonparametric, both based on a simple first-order Markov assumption.

3.2.1 A parametric model

Assume we have N observations and have $K+1$ possible states (including the null state). We can formulate $\mathbf{Z}$ as

$$p(\mathbf{Z}_{1:\tau(N)}|Z_0) = \int \prod_{j=0}^{\tau(N)-1} p(Z_{j+1}|Z_j, \boldsymbol{\theta})\mu(d\boldsymbol{\theta}), \tag{3.2}$$

where $p(Z_{j+1}|Z_j, \boldsymbol{\theta}) = \theta_{Z_j, Z_{j+1}}$. We let the mixing measure μ be a prior on the Markov transition matrix $\boldsymbol{\theta}$. In particular, we let the vector $\boldsymbol{\theta}_k = (\boldsymbol{\theta}_{k,0}, \ldots, \boldsymbol{\theta}_{k,K})$ be distributed as

$$\boldsymbol{\theta}_k \sim \text{Dir}\big(\tfrac{\alpha}{K+1}, \cdots, \tfrac{\alpha}{K+1}\big), \quad 0 \leq k \leq K. \tag{3.3}$$

Notice that together with the null state we have $K+1$ states; we do not separate the null state from the other states, but jointly model them together.

We call this model the finite Markov latent feature model (MLFM in our experiments). When the expected return time to the null state is much smaller than K, MLFM will be a sparse coding model in that each observation will possess a small subset of features. Unlike models based on the beta process, in which the number of features for each observation follows a Poisson distribution, the distribution on the number of features in MLFM cannot be derived in closed-form. However, this distribution is connected to the stationary distribution of the process. We next present a bound on the expected number of features.

Proposition 3. *Suppose $\boldsymbol{\theta}$ is known and Δ is its stationary distribution. Let $\mathcal{A}_i$ be the set of unique features used by object i, then $\mathbb{E}[|\mathcal{A}_i|] \leq \frac{1}{\Delta_0}$, where Δ_0 corresponds to the null state.*

Proof. First observe that $|\mathcal{A}_i| \leq \tau(i) - \tau(i-1)$ because the sequence may return to the same feature multiple times. By the Markov exchangeablity of $\mathbf{Z}$, we have $|\mathcal{A}_i| =_d |\mathcal{A}_1|$ and $\tau(i) - \tau(i-1) =_d \tau(1)$. Since $\theta_{k,k'} > 0$, the Markov chain is regular and thus ergodic. Since $\mathbb{E}[\tau(1)]$ is the expected return time for a regular Markov chain and $\mathbb{E}[\tau(1)] = \frac{1}{\Delta_0}$ [Nor98], the result follows. □

3.2.2 A nonparametric model

In our second example, we extend the above model to an infinite number of features in the spirit of the IBP and beta process. We use the hierarchical Dirichlet processes (HDP) [TJBB06a] to model the mixing measure $\boldsymbol{\theta}$:

$$\boldsymbol{\beta} \sim \text{GEM}(\alpha), \quad \boldsymbol{\theta}_k \sim \text{Dir}(\alpha\boldsymbol{\beta}), \tag{3.4}$$

where $\beta_k = v_k \prod_{k'=0}^{k-1}(1 - v_{k'})$ and $v_{k'} \sim \text{Beta}(1, \alpha)$. We call this model the infinite Markov latent feature model (iMLFM in the experiments).

3.2.3 Application to a linear Gaussian model

We apply these models to the dictionary learning problem using a linear Gaussian model. In this case, the data matrix of N observations $\mathbf{X} = [\mathbf{X}_1, \cdots, \mathbf{X}_N]$ is modeled as

$$\mathbf{X} = \mathbf{W}(\widehat{\mathbf{Z}}^\top \circ \mathbf{C}) + \boldsymbol{\epsilon}, \tag{3.5}$$

where $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_K]$ is the dictionary matrix with K elements, $\mathbf{C}$ is a $K \times N$ matrix, $\boldsymbol{\epsilon} = [\epsilon_1, \cdots, \epsilon_N]$ is a noise matrix, and

$$w_k \sim \mathcal{N}(\mathbf{0}, \tfrac{1}{\eta} I), \quad c_{ki} \sim \mathcal{N}(0, \tfrac{1}{\lambda}), \quad \epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I).$$

As defined above, the coding matrix $\widehat{\mathbf{Z}}$ is generated from the Markov sequence $\mathbf{Z}$ described in Section 3.1 using the priors on either the finite or infinite state Markov chain define above. In the infinite-state model, $K = \infty$.

3.2.4 Discussion

The two examples of an MLFM given above are models that induce *directed* correlations among latent features. However, they are not the only choice. From Theorem 1, we can apply any mixture of recursive Markov chains to build a model. Another choice would be to use an edge-reinforced random walk (ERRW). An ERRW-induced latent feature model is *undirected*. A class of ERRWs has been shown to be a mixture of reversible Markov chains, for which Bayesian inference has recently been studied [DR06, BFT+13]. However, constructing nonparametric priors for reversible Markov chains is non-trivial. [KGP14] provides one solution, but a scalable version has not yet been developed.

Several previous works have studied the theoretical properties of other feature allocation constructions. For example, [BPJ+13] studied an exchangeable class of feature partitions using a "paintbox" characterization, while [HR13] analyzed the combinatorial structure of beta negative binomial processes and [ZPS15] investigate such constructions for feature count matrices.

3.3 Inference

We derive a variational inference algorithm for the parametric Markov latent feature model where we model the mixing measure as a Dirichlet distribution. We can extend inference to the nonparametric

case by modeling $\boldsymbol{\beta}$ in Equation (3.4) using, e.g., the direct assignment method as mentioned in [LPJK07a, JW14b] for the HDP; another recent fully Bayesian method is [ZGP16a]. We exclude this additional step in our algorithm below since the derivation is identical to the HDP in this portion of the model.

3.3.1 Batch Variational Inference

The joint distribution of the Markov latent feature model factorizes as

$$p(\boldsymbol{\theta}, \mathbf{W}, \mathbf{C}, \mathbf{Z}, \mathbf{X}) = \left[\prod_{j=0}^{K} p(\boldsymbol{\theta}_j)p(\mathbf{w}_j)\right] \times \left[\prod_{i=1}^{N} p(\mathbf{C}_i)p(\mathbf{Z}_{\psi(i)}|\boldsymbol{\theta})p(\mathbf{X}_i|\mathbf{C}_i, \mathbf{Z}_{\psi(i)}, \mathbf{W})\right].$$

We recall that $\mathbf{Z}_{\psi(i)}$ is the block of the Markov chain that selects features for the ith observation and must terminate at the null state. We restrict the posterior to a factorized form as well,

$$q(\boldsymbol{\theta}, \mathbf{W}, \mathbf{C}, \mathbf{Z}) = \Big[\prod_{j=1}^{K} q(\boldsymbol{\theta}_j)q(\mathbf{w}_j)\Big]\Big[\prod_{i=1}^{N} q(\mathbf{C}_i)q(\mathbf{Z}_{\psi(i)})\Big],$$

and we define

$$\begin{aligned} q(\boldsymbol{\theta}_j) &= \text{Dir}(\mathbf{a}_j), & q(\mathbf{w}_j) &= \delta_{\mathbf{w}_j}(\cdot), \\ q(\mathbf{C}_i) &= \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), & q(\mathbf{Z}_{\psi(i)}) &= \delta_{\mathbf{Z}_{\psi(i)}}(\cdot). \end{aligned} \tag{3.6}$$

The variational objective is

$$\mathcal{L} = \mathbb{E}_q[\ln p(\boldsymbol{\theta}, \mathbf{W}, \mathbf{C}, \mathbf{Z}, \mathbf{X})] - \mathbb{E}_q[\ln q(\boldsymbol{\theta}, \mathbf{W}, \mathbf{C}, \mathbf{Z})].$$

Using this factorization, we observe in advance that our algorithm below is equivalent to a MAP-EM algorithm for maximizing $p(\mathbf{X}, \mathbf{W}, \mathbf{Z})$, where $\mathbf{C}$ and $\boldsymbol{\theta}$ constitute the hidden data. This is because

$$q(\boldsymbol{\theta}, \mathbf{W}, \mathbf{C}, \mathbf{Z}) = q(\boldsymbol{\theta}, \mathbf{C}|\mathbf{W}, \mathbf{Z})q(\mathbf{W}, \mathbf{Z})$$

and $\boldsymbol{\theta}$ and $\mathbf{C}$ are conditionally independent and can be solved exactly given the point estimates $\mathbf{W}$ and $\mathbf{Z}$, which the delta q distribution enforces. In other words, the mean-field representation for $\boldsymbol{\theta}$ and $\mathbf{C}$ is exact and not an approximation in this case.

Update Z and C: Sparse coding with greedy search. We jointly update $\mathbf{Z}_{\psi(i)}$ and $\mathbf{C}_i$ using a new approach to sparse coding with Bayesian models. The method is similar to orthogonal matching pursuits, used in sparse coding by K-SVD [AEB06], in that it greedily selects the next feature to add by

Algorithm 3 Sparse coding with greedy search

Input: $q(\boldsymbol{\theta})$ and $\mathbf{W}$.
for $i = 1$ **to** N **do**
  1. Set $\mathbf{Z}_{\psi(i)} = \emptyset$ and $\mathcal{A}_i = \emptyset$.
  while $\max_j \xi_j > 0$ **do**
    (a) Set $\xi_j = \mathcal{L}_{\mathcal{A}_i}([\mathbf{Z}_{\psi(i)}, j, 0]) - \mathcal{L}_{\mathcal{A}_i}([\mathbf{Z}_{\psi(i)}, 0])$.
    (b) Set $j' = \arg\max_j \xi_j$.
    (c) Set $\mathbf{Z}_{\psi(i)} \leftarrow [\mathbf{Z}_{\psi(i)}, j']$ and $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{j'\}$.
  end while
  Set $\mathbf{Z}_{\psi(i)} = [\mathbf{Z}_{\psi(i)}, 0]$.
end for

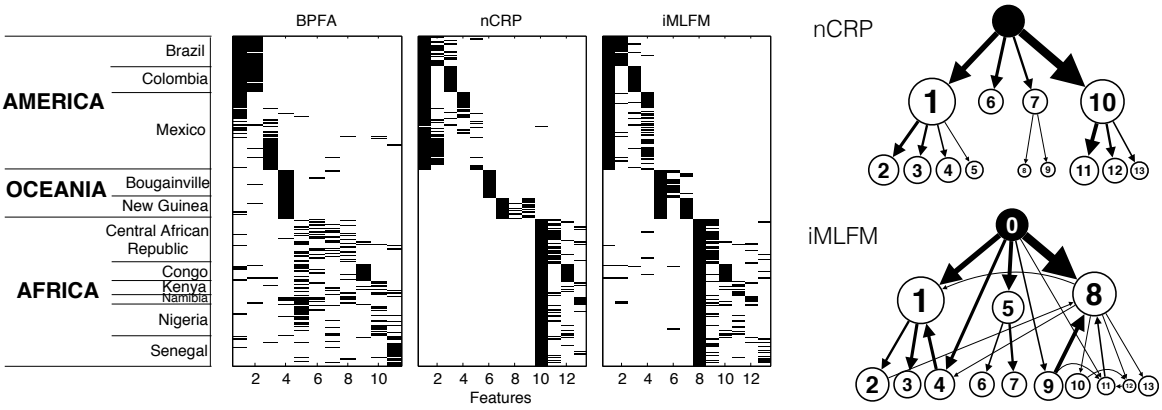


Figure 3.2: (Left) Factors learned from BPFA, nCRP, and iMLFM on the HGDP-CEPH dataset. Observations from various regions are aligned vertically, as displayed on the left side. (Right) Graph learned from iMLFM and nCRP.

integrating out the corresponding weight, followed by an update of all weights on the active features. The structure of the algorithm is also similar to MAP-EM inference for mixtures of factor analyzers [GH96a].

To sparsely code the ith observation, we can focus on the objective term

$$\mathcal{L}(\mathbf{Z}_{\psi(i)}, q(\mathbf{C}_i)) = \mathbb{E}_q \left[\ln \frac{p(\mathbf{X}_i, \mathbf{C}_i, \mathbf{Z}_{\psi(i)} | \boldsymbol{\theta}, \mathbf{W})}{q(\mathbf{C}_i)} \right]. \tag{3.7}$$

First observe that given $\mathbf{Z}_{\psi(i)}$,

$$q(\mathbf{C}_i) = p(\mathbf{C}_i | \mathbf{X}_i, \mathbf{Z}_{\psi(i)}, \mathbf{W}) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3.8}$$

is known exactly and is a multivariate Gaussian derived explicitly below. Here, we jointly update $\mathbf{Z}_{\psi(i)}$ and $q(\mathbf{C}_i)$ by incrementally extending (or terminating) the path $\mathbf{Z}_{\psi(i)}$.

Because our inference problem is equivalent to MAP-EM for the joint likelihood $p(\mathbf{X}, \mathbf{W}, \mathbf{Z})$ we can do this as follows: Let $\mathcal{A}_i$ be the current set of features selected by the path $\mathbf{Z}_{\psi(i)}$ and $q(\mathbf{C}_{\mathcal{A}_i}) =$

$\mathcal{N}(\boldsymbol{\mu}_{\mathcal{A}_i}, \boldsymbol{\Sigma}_{\mathcal{A}_i})$ be the corresponding marginal posterior over these dimensions, where

$$\boldsymbol{\Sigma}_{\mathcal{A}_i} = (\lambda I + \tfrac{1}{\sigma^2}\mathbf{W}_{\mathcal{A}_i}^\top \mathbf{W}_{\mathcal{A}_i})^{-1},\ \boldsymbol{\mu}_{\mathcal{A}_i} = \tfrac{1}{\sigma^2}\boldsymbol{\Sigma}_{\mathcal{A}_i}\mathbf{W}_{\mathcal{A}_i}^\top \mathbf{X}_i.$$

We expand the path of $\mathbf{Z}_{\psi(i)}$ using EM by constructing

$$\mathcal{L}_{\mathcal{A}_i}(\mathbf{Z}_{\psi(i)}) = \mathbb{E}_q[\ln p(\mathbf{X}_i, \mathbf{Z}_{\psi(i)} | \mathbf{C}_{\mathcal{A}_i}, \mathbf{W}, \boldsymbol{\theta})]$$

where the expectation is over $\boldsymbol{\theta}$ and the subset of $\mathbf{C}_i$ indexed by $\mathcal{A}_i$ using $q(\mathbf{C}_{\mathcal{A}_i})$ derived above. The remaining dimensions of $\mathbf{C}_i$ are marginalized out *a priori*. Since we are dealing with multivariate normal variables, all calculations are in closed form and remain Gaussian.

We then greedily pick the next state that improves this objective the most and add that state to the end of the path $\mathbf{Z}_{\psi(i)}$, or we terminate if moving to the null state and adding no more features provides the best improvement. When the algorithm terminates, we have a point estimate of the path $\mathbf{Z}_{\psi(i)}$ that selects the latent features for the ith observation, and the corresponding conditional posterior q distribution on the weight vector $\mathbf{C}_i$. By the equivalent MAP-EM construction of this algorithm, each step is guaranteed to increase the objective function. We summarize this greedy algorithm in Algorithm 1.

Proposition 4. *The sparse coding greedy search algorithm will stop in a finite number of steps.*

Proof. First, note the $q(\mathbf{C}_i)$ is a function of $\widehat{\mathbf{Z}}_i$. The objective function in Eq. (3.7) becomes

$$\mathcal{L}(\mathbf{Z}_{\psi(i)})) = \mathcal{L}_1(\mathbf{Z}_{\psi(i)}) + \mathcal{L}_2(\mathbf{Z}_{\psi(i)}), \tag{3.9}$$

where

$$\begin{aligned}\mathcal{L}_1(\mathbf{Z}_{\psi(i)}) &= \mathbb{E}_q[\ln p(\mathbf{Z}_{\psi(i)} | \boldsymbol{\theta})] \\ \mathcal{L}_2(\mathbf{Z}_{\psi(i)}) &= \mathbb{E}_q[\ln p(\mathbf{X}_i | \mathbf{C}_i, \widehat{\mathbf{Z}}_i, \mathbf{W})] + f(\widehat{\mathbf{Z}}_i).\end{aligned} \tag{3.10}$$

Here $f(\widehat{\mathbf{Z}}_i) = \mathbb{E}_q[\ln \frac{p(\mathbf{C}_i)}{q(\mathbf{C}_i)}]$, by marginalizing out $\mathbf{C}_i$. Note that $\mathcal{L}_2(\mathbf{Z}_{\psi(i)})$ can only take finite values, since there are finite configurations for $\widehat{\mathbf{Z}}_i$. We only need to prove that $\mathcal{L}_1(\mathbf{Z}_{\psi(i)})$ cannot always be improved. We have

$$\mathcal{L}_1(\mathbf{Z}_{\psi(i)}) = \sum_{j=\tau(i-1)+1}^{\tau(i)} \mathbb{E}_q[\ln \theta_{Z_{j-1}, Z_j}]. \tag{3.11}$$

Let $\mathcal{L}_1^{(1)} = \sum_{j=\tau(i-1)+1}^{\tau(i)-1} \mathbb{E}_q[\ln \theta_{Z_{j-1}, Z_j}]$ be the cost of all transitions execpt for the last one, and $\mathcal{L}_1^{(2)}$

be the rest part of $\mathcal{L}_1$. Note that as the sequence grows, $\mathcal{L}_1^{(1)}$ is monotonically decreasing, since $\theta_{Z_{j-1},Z_j} < 1$. And $\mathcal{L}_1^{(2)}$ (the cost of last transition plus a constant) can only take finite values. Thus, $\mathcal{L}_1$ cannot monotonically increase through greedy search. □

Furthermore, we observe in our experiments that the algorithm tends to terminate after a small fraction of available features have been selected, using a number comparable to IBP-based models.

Update $q(\boldsymbol{\theta})$: The distribution $q(\boldsymbol{\theta}_j) = \text{Dir}(\mathbf{a}_j)$ can be found by optimizing the Markov chain portion of the objective function below,

$$\mathcal{L}(q(\boldsymbol{\theta})) = \mathbb{E}_q[\ln p(\boldsymbol{\theta})] + \textstyle\sum_{i=1}^{N} \mathbb{E}_q[\ln p(\mathbf{Z}_{\psi(i)}|\boldsymbol{\theta})].$$

Since $\mathbf{Z}_{\psi(i)}$ is a point estimate, this is equivalent to finding the conditional posterior of $\boldsymbol{\theta}_j$, and thus

$$\mathbf{a}_{j,j'} = \frac{\alpha}{K+1} + \sum_{i=0}^{\tau(N)-1} \mathbb{1}(Z_i = j, Z_{i+1} = j').$$

Update W: For this point estimate, we want to maximize

$$\mathcal{L}(\mathbf{W}) = \ln p(\mathbf{W}) + \textstyle\sum_{i=1}^{N} \mathbb{E}_q[\ln p(\mathbf{X}_i|\mathbf{C}_i, \widehat{\mathbf{Z}}_i, \mathbf{W})].$$

Let $q(\mathbf{C}_i) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, and define $\widetilde{\mathbf{Z}}_i = \text{diag}(\widehat{\mathbf{Z}}_i)$. We can differentiate with respect to $\mathbf{W}$ to find that

$$\mathbf{W} = \Big[\sum_{i=1}^{N} \mathbf{X}_i \boldsymbol{\mu}_i^\top \widetilde{\mathbf{Z}}_i\Big] \Big[\eta\sigma^2 I + \sum_{i=1}^{N} \widetilde{\mathbf{Z}}_i(\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top + \boldsymbol{\Sigma}_i)\widetilde{\mathbf{Z}}_i\Big]^{-1}.$$

3.3.2 Stochastic Variational Inference

We use stochastic variational inference (SVI) [HBWP13b] to scale up MLFM by using stochastic optimization with natural gradients. Suppose N is large, the objective function for $\mathbf{W}$ is

$$\mathcal{L}(\mathbf{W}) = \ln p(\mathbf{W}) + \sum_{i=1}^{N} \mathbb{E}_q[\ln p(\mathbf{X}_i|\mathbf{C}_i, \widehat{\mathbf{Z}}_i, \mathbf{W})]. \tag{3.12}$$

At iteration t we sample a subset indexed by C_t and set the objective

$$\mathcal{L}_t(\mathbf{W}) = \ln p(\mathbf{W}) + \frac{N}{|C_t|} \sum_{i \in C_t} \mathbb{E}[\ln p(\mathbf{X}_i|\mathbf{C}_i, \widehat{\mathbf{Z}}_i, \mathbf{W})]. \tag{3.13}$$

Then we can perform an unbiased natural gradient decent for $\mathbf{W}$ as follows:

$$\mathbf{B}_t = \eta\sigma^2 \frac{|C_t|}{N} I + \sum_{i \in C_t} \widetilde{\mathbf{Z}}_i(\boldsymbol{\mu}_i\boldsymbol{\mu}_i^\top + \Sigma_i)\widetilde{\mathbf{Z}}_i,$$

Table 3.1: PSNR | SSIM (average # features used per patch) on various images and noise settings.

BARBARA	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$
IMLFM	**38.28** | **0.958** (3.58)	**34.74** | **0.932** (2.17)	**32.47** | **0.909** (1.59)	30.81 | 0.885 (1.31)	**29.56** | 0.857 (1.16)
MLFM	37.82 | 0.953 (3.80)	34.55 | 0.930 (2.09)	**32.47** | 0.908 (1.58)	**30.86** | **0.886** (1.32)	29.53 | 0.857 (1.17)
BPFA	37.26 | 0.949 (4.50)	34.22 | 0.927 (2.40)	32.23 | 0.907 (1.72)	30.67 | 0.885 (1.45)	29.51 | **0.859** (1.27)
KSVD	38.05 | 0.956 (7.04)	34.45 | 0.930 (3.11)	32.41 | 0.907 (1.77)	**30.86** | 0.881 (1.16)	29.55 | 0.852 (0.83)
TV(ANISO)	34.17 | 0.936 (—)	29.77 | 0.877 (—)	27.49 | 0.820 (—)	26.00 | 0.770 (—)	25.07 | 0.728 (—)
TV(ISO)	34.18 | 0.936 (—)	29.77 | 0.877 (—)	27.50 | 0.822 (—)	26.01 | 0.773 (—)	25.12 | 0.734 (—)
BASELINE	34.16 | 0.887 (—)	28.14 | 0.724 (—)	24.61 | 0.594 (—)	22.11 | 0.497 (—)	20.18 | 0.422 (—)
GOLDHILL	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$
IMLFM	35.72 | 0.935 (3.22)	32.70 | 0.881 (1.65)	31.12 | **0.838** (1.22)	**30.03** | **0.799** (1.09)	**29.15** | **0.764** (1.03)
MLFM	35.20 | 0.928 (3.20)	32.64 | 0.878 (1.58)	31.15 | 0.837 (1.24)	30.00 | 0.797 (1.09)	29.14 | 0.762 (1.03)
BPFA	34.73 | 0.924 (3.05)	32.17 | 0.875 (1.39)	30.87 | 0.833 (1.30)	29.89 | 0.790 (1.16)	29.09 | 0.762 (1.08)
KSVD	**36.65** | **0.941** (6.78)	**33.25** | **0.885** (2.49)	**31.40** | 0.832 (1.25)	30.01 | 0.787 (0.77)	29.05 | 0.746 (0.53)
TV(ANISO)	34.73 | 0.908 (—)	31.43 | 0.833 (—)	29.74 | 0.776 (—)	28.61 | 0.732 (—)	27.79 | 0.696 (—)
TV(ISO)	34.81 | 0.910 (—)	31.52 | 0.836 (—)	29.83 | 0.781 (—)	28.69 | 0.736 (—)	27.87 | 0.700 (—)
BASELINE	34.16 | 0.897 (—)	28.14 | 0.727 (—)	24.61 | 0.577 (—)	22.11 | 0.461 (—)	20.18 | 0.373 (—)
LENA	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$
IMLFM	37.49 | 0.934 (2.39)	34.99 | **0.905** (1.53)	33.51 | 0.879 (1.25)	32.38 | 0.861 (1.12)	**31.43** | **0.844** (1.06)
MLFM	37.36 | 0.931 (2.29)	35.07 | **0.905** (1.53)	33.58 | 0.880 (1.26)	**32.42** | **0.862** (1.13)	31.40 | 0.843 (1.07)
BPFA	37.41 | 0.932 (2.56)	34.92 | 0.903 (1.73)	33.41 | **0.882** (1.45)	32.31 | 0.861 (1.25)	31.17 | 0.840 (1.13)
KSVD	**38.26** | **0.937** (4.22)	**35.38** | **0.905** (1.66)	**33.68** | 0.879 (0.91)	32.40 | 0.856 (0.61)	31.30 | 0.832 (0.43)
TV(ANISO)	35.92 | 0.917 (—)	32.71 | 0.874 (—)	30.96 | 0.841 (—)	29.84 | 0.816 (—)	28.87 | 0.793 (—)
TV(ISO)	35.95 | 0.917 (—)	32.78 | 0.874 (—)	31.04 | 0.843 (—)	29.93 | 0.818 (—)	28.98 | 0.796 (—)
BASELINE	34.16 | 0.855 (—)	28.14 | 0.646 (—)	24.61 | 0.493 (—)	22.11 | 0.390 (—)	20.18 | 0.317 (—)
PEPPERS	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$
IMLFM	36.47 | 0.935 (2.30)	34.23 | 0.924 (1.46)	33.00 | **0.905** (1.23)	31.94 | 0.884 (1.12)	**31.09** | **0.871** (1.05)
MLFM	35.97 | 0.931 (2.12)	**34.28** | **0.926** (1.46)	**33.01** | **0.905** (1.24)	**31.99** | **0.885** (1.13)	**31.09** | 0.869 (1.07)
BPFA	35.61 | 0.937 (2.33)	34.10 | 0.920 (1.64)	32.45 | 0.904 (1.38)	31.37 | 0.882 (1.23)	30.46 | 0.864 (1.11)
KSVD	**37.72** | **0.949** (4.82)	34.20 | 0.923 (1.73)	32.16 | 0.900 (0.90)	30.80 | 0.877 (0.58)	29.64 | 0.855 (0.42)
TV(ANISO)	35.73 | 0.938 (—)	32.40 | 0.903 (—)	30.44 | 0.872 (—)	29.25 | 0.850 (—)	28.26 | 0.828 (—)
TV(ISO)	35.85 | 0.938 (—)	32.56 | 0.905 (—)	30.59 | 0.875 (—)	29.42 | 0.853 (—)	28.40 | 0.832 (—)
BASELINE	34.16 | 0.855 (—)	28.14 | 0.642 (—)	24.61 | 0.485 (—)	22.11 | 0.382 (—)	20.18 | 0.310 (—)

$$\mathbf{W}'_t = (\textstyle\sum_{i \in C_t} \mathbf{X}_i \boldsymbol{\mu}_i^\top \widetilde{\mathbf{Z}}_i)\mathbf{B}_t^{-1},$$
$$\mathbf{W}^{(t+1)} = (1-\rho_t)\mathbf{W}^{(t)} + \rho_t \mathbf{W}'_t. \tag{3.14}$$

Similarly, to update the conditional posterior of $\boldsymbol{\theta}$, we have $q(\boldsymbol{\theta}) = \prod_{k=0}^{K} q(\boldsymbol{\theta}_k)$, where $q(\boldsymbol{\theta}_k) = \mathrm{Dir}(\mathbf{a}_k)$. We update

$$\mathbf{a}'_{j,j'} = \frac{\alpha}{K+1} + \frac{N}{|C_t|} \sum_{i \in C_t} \sum_{i'=\tau(i-1)+1}^{\tau(i)} \mathbb{1}_{\{Z_{i'}=j, Z_{i'+1}=j'\}},$$
$$\mathbf{a}_{j,j'}^{(t+1)} = (1-\rho_t)\mathbf{a}_{j,j'}^{(t)} + \rho_t \mathbf{a}'_{j,j'}, \tag{3.15}$$

where $\rho_t = (t+t_0)^{-\kappa}$, and $\kappa \in (0.5, 1]$.

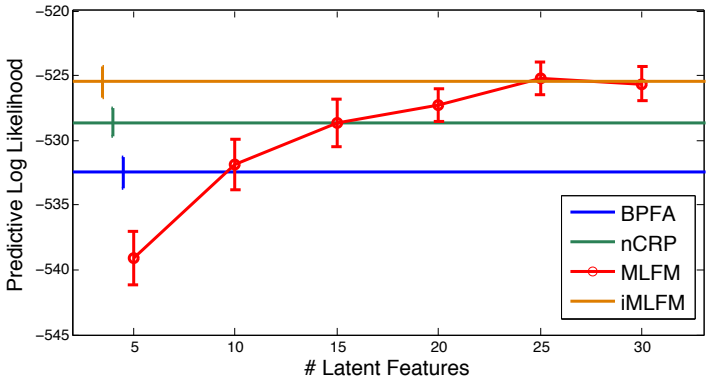


Figure 3.3: Average predictive result on HGDP-CEPH dataset.

3.4 Experiments

We demonstrate the effectiveness of our MLFM framework on two tasks. The first task is an analysis of the HGDP-CEPH cell line panel dataset [RPW+02] for which we test batch learning performance of our two models. The second task is image denoising, where batch learning is slower and so we use stochastic inference.

3.4.1 HGDP-CEPH Cell Line Panel

For this small-scale experiment, we use a subset of 266 individuals across 11 countries from the HGDP-CEPH Human Genome Diversity Cell Line Panel [RPW+02][1]. Each person is represented by their genotypes measured at $D = 377$ autosomal microsatellite loci.

We split this data into a set of 54 individuals for testing, and use the rest for training. For evaluation we use average predictive log-likelihood of the testing set, which we approximated using Monte Carlo integration over the q distributions. We experiment with MLFM letting K range from 5 to 30 features, and iMLFM truncated to 100 latent features. We compare with BPFA [PC09b] and nested CRP [BGJ10b], which we modified into a linear Gaussian model. Both models were truncated to 100 latent features as well. We set hyper-parameters to be $\eta = 1, \lambda = 1, \sigma = 0.8$. We ran each model 20 times and averaged the results. All models converge by 100 global iterations and the time cost is similar across all models.

We show the mean and variance of the predictive log-likelihood for all models in Figure 3.3. Since

[1] As reported in [RPW+02], the remaining individuals form two large heterogeneous clusters which are hard to distinguish in general.

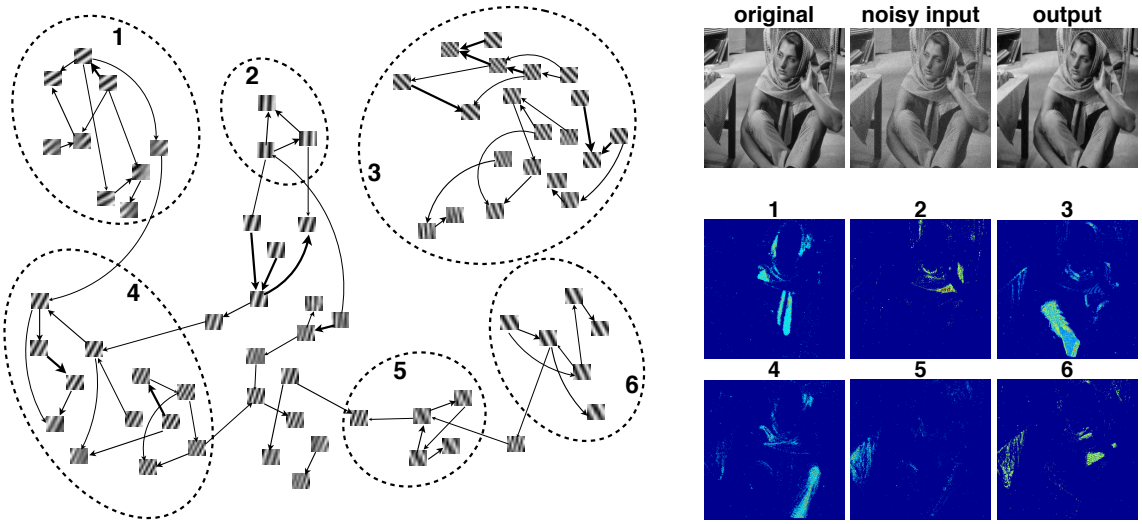


Figure 3.4: (Left) A similarity-preserving 2-d embedded graph of the learned dictionary elements using their transition probabilities. We show edges about a threshold, and the direction of an edge between two dictionary elements according to the higher transition probability. (Right) Feedback maps for dictionary elements in various regions in the graph on the left.

the nonparametric models are not a function of latent feature, we show their performance as a straight line. We observe that the performance for MLFM starts to decrease when K reaches 25 and its best performance is the same as the nonparametric iMLFM, which is not surprising. We also observe that nonparametric model performance is ordered as BPFA $<$ nCRP $<$ iMLFM. We believe that this improvement in performance is a result of the increasing ability to model the relationships between features in this sequence.

We also show some qualitative results for these models. In Figure 3.2 we show the 0-1 feature assignment matrix for the three nonparametric models. To the left we show the country of origin for the person associated with each row. In the columns we show the most heavily used features and re-order them for a better visualization. Since BPFA assumes all the features are mutually independent, it has a harder time exploiting the natural structure in the data. The nCRP gives a better result since it learns a hierarchy of features spread across countries and continents. However, since the model uses a strict tree structure as shown on the RHS of Figure 3.2, it may not be flexible enough to uncover all the hierarchical correlations.

On the RHS of Figure 3.2 we also show the graph learned by iMLFM, where the 13 significantly used features (and the null state, denoted by a dark node) are displayed. The entire graph looks like a tree when organized according to transition probability as we have done, but there are some

differences. First, the structure is not a strict hierarchy. For example, there are transitions from the null state to the "leaves." There are also transitions across "subtrees." Thus, we learn a graph structure that approximates a tree, following the structure of the data, but allows for individuals not to adhere strictly to this (in this case because, e.g., there may have been some ancestor from another region). This shows the flexibility of our Markov-structured model.

3.4.2 Image denoising

We also experiment using scalable inference for an image denoising task, where we would like to recover the original image from an image corrupted by white Gaussian noise. We demonstrate results for four 512×512 images: 'Barbara', 'Goldhill', 'Lena', and 'Peppers' (below in Figure 3.5). We extract 8×8 patches from the noise-corrupted images using a single-pixel sliding window to scan the entire image. This gives a total of 255,025 patches.

We compare with scalable BPFA [SP15], the non-probabilistic K-SVD model [AEB06], and isotropic and anisotropic total variation (TV) [GO09]. We set $\eta = 1/255^2,\ \lambda = 1/10,\ \alpha = 1,\ \gamma = 1,\ K = 256$, and online parameters $|C_t| = 1000,\ t_0 = 10,\ \kappa = 0.75$. We truncated iMLFM to 256 states. For all methods, we set σ using the method from [LTO13]; for TV we found the regularization parameter that resulted in this empirical noise variance. For the stochastic algorithms, we train using 500 iterations, which was enough for convergence; thus the number of patches seen during inference was equivalent to two passes through the entire dataset. To quantify the recovery quality, we show the peak signal-to-noise ratio (PNSR) and the structured similarity (SSIM) performance measures [WBSS04].

We show the result on various images using different noise standard deviations in Table 3.1. As a baseline performance measure, we use the original noisy image. We can see that iMLFM often performs better than BPFA and has results comparable to K-SVD. In the images 'Barbara' and 'Peppers', iMLFM performs better than K-SVD. We also show the average number of features occupied by each

Figure 3.5: The four images used in our experiments.

patch in Table 3.1. iMLFM is sparser than K-SVD when σ is small, which is a matter of balancing predictive gain and extra cost of growing paths in our greedy sparse coding step. For the running time, MLFM takes about 4 minutes to converge, which is similar to K-SVD and BPFA.

Finally, we show some qualitative result on the 'Barbara' image in Figure 3.4. We show the graph learned by iMLFM using a similarity-preserving 2-d embedding, where edges having probabilities above a threshold are displayed between dictionary elements. Since iMLFM learns a directed graph, we show the direction of edges according to the larger transition probability between two elements. For the ease of visualization, we only show the latent features that contain stripes. On the LHS of Figure 3.4, we see that the direction of stripes varies, but stripes with similar directions have greater connectivity. The graph also has a global structure as the similarity-preserving embedding shows. For example, the direction of stripes changes from the direction '$\backslash$' in the right area, to '$|$' in the center area, and to '$/$' in the left area.

On the RHS of Figure 3.4, we display the feedback map of dictionary elements in the six regions defined on the LHS. As we can see, some groups of features give a local feedback that has semantic meanings. For example, Region 1 (9 features) can be interpreted as the scarf; Region 3 (20 features) is the right leg; Region 4 (12 features) is the left leg; Region 5 (5 features) is the tablecloth. Above we show the ground truth image, the noisy input and the denoised output for the corresponding experiment.

We presented a Markov latent feature model (MLFM) using a simple sequential construction and connected this construction to the requisite Markov property of the stochastic process. The key is through the Markov exchangeability constraint, which allows for a mixing measure to be defined for easy variational inference. This procedure for constructing latent features models allows for feature correlations to be learned from the data, and so in a sense we have presented a "correlated IBP"-type model. However, MLFM pre-determines the number of parameters as K^2 it use for correlation modeling, and thus a rigid model. In the next chapter, we present a more flexible alternatives by modeling the feature allocation matrix $\widehat{\mathbf{Z}}$ as a discrete random measure.

Chapter 4

Random Function Priors for Correlation Modeling

In this chapter, we formalize LFMs as 2d arrays Z with N rows and K columns, where each row Z_n is a non-negative factor loading vector with K entries where Z_{nk} indicates the strength of a hidden feature θ_k used to express object X_n. Thus likelihood can be written as $p(X_n|Z_n, \theta)$, where $\theta := (\theta_k)_{k\in[K]}$ is a collection of hidden features shared across objects. In this chapter, we introduce random function priors for Z_n for modeling correlations among its K dimensions Z_{n1} through Z_{nK}, which we call *population random measure embedding* (PRME). Our model can be viewed as a generalized paintbox model [BPJ+13] using random functions, and can be learned efficiently with neural networks via amortized variational inference. We derive our Bayesian nonparametric method by applying a representation theorem on separately exchangeable discrete random measures.

Let $X = [X_1, \ldots, X_N]$ be a group of exchangeable high dimensional observations, where $X_n \in \mathbb{R}^d$. In this chapter, we assume X is generated by the model

$$\begin{aligned} p(X) &= \int p(X, Z, \theta)\, dZ d\theta \\ &= \int p(Z)p(\theta) \prod_{n\in[N]} p(X_n|Z_n, \theta)\, dZ d\theta, \end{aligned} \tag{4.1}$$

where $p(X_n|Z_n, \theta)$ is a likelihood model conditioned on latent features $\theta := (\theta_k)_{k\in[K]}$ that are shared across the population. $Z_n := [Z_{n1}, \ldots, Z_{nK}]$ is a non-negative vector for the nth observation, where Z_{nk} determines the extent to which θ_k is used to express X_n. For example, in topic models [BNJ03b],

Z_n is a discrete distribution over topics, where Z_{nk} represents the proportion of words in document n sampled from topic k. In sparse factor models [GG11], Z_n is a binary vector such that latent feature θ_k contributes to the likelihood if and only if $Z_{nk} = 1$. We generically refer to Z_n as a "non-negative feature loading vector." For exchangeable X, it is often assumed the Z_n are exchangeable as well. If we take Z as a feature loading matrix with Z_n as its rows, then Z is *row exchangeable*. By de Finetti's theorem, we can represent

$$p(Z) = \int \prod_{n \in \mathbb{N}} p(Z_n|\zeta) p(\zeta) d\zeta, \tag{4.2}$$

for some random object ζ. (We let $N = \infty$ in order to apply de Finetti's theorem.) The goal of this chapter is to model complex correlations among entries of Z_n. Following a common practice, we put an independent prior on θ, $p(\theta) = \prod_k p(\theta_k)$ and focus on modeling $p(Z)$.

A straightforward way to model correlation structure is to let $p(Z_n|\zeta)$ be a *parametric* exponential family model. By defining the mean/natural parameters for the model, one can handle correlations to various degrees. For example, Z_n may follow a log-normal distribution [LB06], where correlations are modeled through a covariance matrix. However, exponential family models [WJ+08] can be rigid, since the number of free parameters is fixed for a certain K. To get a more flexible model, it is tempting to consider higher-order moments $\mathbb{E}[Z_n^{\otimes M}]$ for a large M up to K, where $u^{\otimes M}$ denotes an M-th order outer product of a vector u. but in this case the number of free parameters increases exponentially, leading to intractable inference.

In this chapter, we use an alternative *Bayesian nonparametric* method to model Z_n as an outcome of random functions, which can handle complex correlations even when K and M go to infinity. Moreover, those random functions can be learned efficiently through inference/decoder networks via amortized variational inference [KW13]. In principle, arbitrarily complex neural networks can be applied to model correlations in our setting.

To give intuition why random function priors are powerful, we first show in Figure 4.1 an existing feature paintbox model for binary Z_n that illustrates how to model arbitrarily complex correlations using binary random functions [BPJ+13]. For simplicity, let $K = 3$. First, select a compact set S in Euclidean space, on which we can define a uniform distribution. For example let $S = [0, 1]^2$. Then randomly partition S into eight regions. Each partition represents a possible value for $Z_n = [Z_{n1}, Z_{n2}, Z_{n3}]$, as shown in Figure 4.1. Given the partition, we uniformly sample a point $u_n \sim U(S)$ and assign Z_n be the value defined by the region in which u_n falls. Thus, we translate the problem of

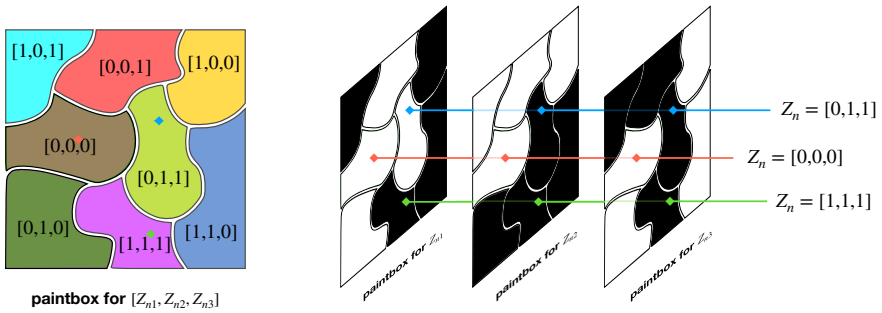


Figure 4.1: Example of two equivalent representations: (left) Paintbox model for $[Z_{n1}, Z_{n2}, Z_{n3}]$ by partitioning a unit square into eight regions, one for each distinct value. (right) Factorize the paintbox into three feature paintboxes, one each for a latent feature. Three examples of u_n that determine Z_n are demonstrated as dots in the partition paintbox, and as lines across feature paintboxes.

modeling distributions on Z_n to modeling *the random partition of* S. Following the classic analogy, we call this a partition paintbox model [Kin78, Pit06]. One can further factorize the partition paintbox into "feature paintboxes" [BPJ+13]. According to Figure 4.1, each feature paintbox for the k-th feature is randomly partitioned into two regions denoted as S_k (black) and S_k^c (white). Let $Z_{nk} = \mathbb{1}(u_n \in S_k)$. One can check that the feature paintbox model is the equivalent to the partition paintbox model for arbitrary finite K. (Note that here Z_n is a random indicator function.)

The feature paintbox model is redundant but flexible. The arbitrary order moment $\mathbb{E}[\prod_{k\in\mathcal{J}} Z_{nk}] = \mathbb{E}[\text{vol}(\cap_{k\subset\mathcal{J}} S_k)]$ for any $\mathcal{J} \subset [K]$ can be modeled once we have enough freedom for S_k. We summarize the generative process for the feature paintbox model in Algorithm 4 for arbitrary K, including $K = \infty$.

We propose a model that can be treated as a generalization of the feature paintbox model from binary to non-negative Z according to a function $Z_{nk} = f_n(\vartheta_k)$. There are two key differences between our model and the feature paintbox model. First, we use data-specific *random functions* f_n, instead of points u_n, to represent each observation. Second, we use points ϑ_k from a *Poisson process*, instead of S_k, to index each latent feature. A nice property of our model compared to the paintbox model is that we can use deep learning to model f_n through inference and decoder networks [KW13], allowing for efficient amortized variational inference.

In what follows, Section 4.1 sets up the problem of modling Z from a random matrix point of view. In Section 4.2, we embed Z as a random measure and derive the functional form of $Z_{nk} = f_n(\vartheta_k)$ through a representation theorem. In Section 4.3, we present a concrete example for Bayesian nonparametric topic modeling together with its amortized variational inference algorithm, and show empirical results in Section 4.4. Finally, we discuss related work in Section 4.5.

Algorithm 4 Feature paintboxes model

for $k \in [K]$ **do**
  Generate a random subset $S_k \subset S$.
end for
Guarantee that $\sum_{k\in[K]} \mathrm{vol}(S_k) < \infty$ almost surely.
for $n = 1, 2, \ldots$ **do**
  Independently generate $u_n \sim U(S)$.
  Let $Z_n = [Z_{n1}, \ldots, Z_{nK}]$. Set $Z_{nk} = \mathbb{1}(u_n \in S_k)$.
end for

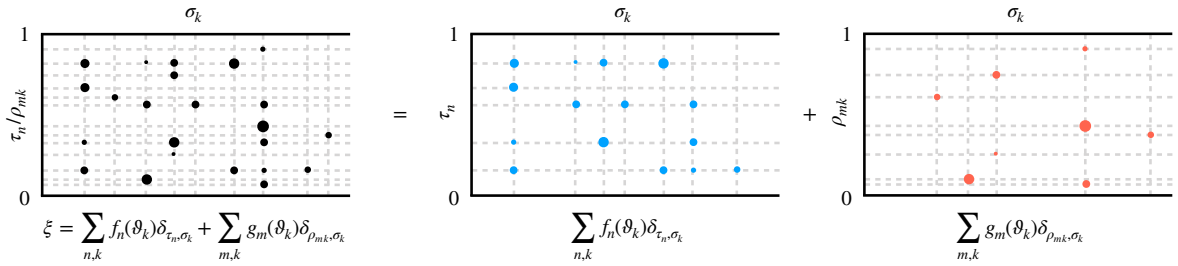


Figure 4.2: Decoupling a separately exchangeable discrete random measure ξ into two parts.

4.1 Z as a random matrix?

We will rely on representation theorems to derive the functional form of our models. This usually works out by finding an infinite dimensional random object paired with an exchanegability assumption on that random object. The choice of random objects is the key step, and we will see below that it can be hard to derive an interesting model when choosing a bad random object.

Consider modeling Z as a random matrix. Equation (4.2) above is one example that derives a mixture representation by assuming row exchangeability of Z. However, Equation (4.2) is uninformative in that, first, it does not tell us what random object ζ is, and second, it does not determine the connection between Z_n and ζ through $p(Z_n|\zeta)$. Our discussion in Section 1 will show that this provides too much freedom to choose ζ and $p(Z_n|\zeta)$.

We further restrict Z by assuming it is *column exchangeable* as well. This requires allowing both N and K to equal infinity. We call Z *separately exchangeable* if it is both row and column exchangeable. Once $K = \infty$, we need to guarantee series convergence for rows. That is, $\sum_{k\in\mathbb{N}} Z_{nk} < \infty$ with probability 1, for any $n \in \mathbb{N}$. Row sum convergence is always considered necessary. (For example, in a topic model we want to normalize Z_n.) However, the following proposition says that when Z is separately exchangeable, we will get an empty model even for a binary Z.

Proposition 5. *An infinite binary matrix Z (i) is separately exchangeable, and (ii) has finite row sums almost*

surely, if and only if $Z = \mathbf{0}$ almost surely.

Proof (Sketch). One can prove that Z is a graphon model if it is separately exchangeable [Hoo79, Ald85, OR15b]. A graphon model satisfies finite row sums if and only if $Z = \mathbf{0}$. □

When choosing a bad random object, one can either get a vacuous or an empty model through representation theorems. In the next section, we fix this problem by introducing a nice random object ξ generated by embedding Z as a random measure. Then we apply representation theorems on ξ.

4.2 Z as a random measure

4.2.1 Population random measure embedding

In this section, we embed the random matrix Z as a *discrete random measure* $\xi = \sum_{n,k} Z_{nk}\delta_{\tau_n,\sigma_k}$ on an infinite strip $[0,1] \times \mathbb{R}_+$, where $(\tau_n)_{n\in\mathbb{N}} \subset [0,1]$ distinguishes objects and $(\sigma_k)_{k\in\mathbb{N}} \subset \mathbb{R}_+$ distinguishes latent features. Both $(\tau_n)_{n\in\mathbb{N}}$ and $(\sigma_k)_{k\in\mathbb{N}}$ are random as well, and are not necessarily ordered. Note that ξ preserves the matrix structure as demonstrated in Figure 4.2; the intersection points of horizontal/vertical dashed lines indexed by $(\tau_n)_{n\in\mathbb{N}}$ and $(\sigma_k)_{k\in\mathbb{N}}$ form an "equivalent class" of matrix Z up to a re-ordering of rows and columns. The infinite strip is an abstract space introduced solely for applying representation theorems.

Next, we assume ξ is separately exchangeable. That is, $\xi(\mathcal{T}_1(A) \times \mathcal{T}_2(B)) =_d \xi(A \times B)$ for any measure-preserving transformations $\mathcal{T}_1, \mathcal{T}_2$ on $[0,1]$ and $\mathbb{R}_+$ separately for arbitrary Borel sets A, B. Even though the notion of separate exchangeability is different for ξ than for random matrix Z, they are conceptually similar, since interchanging row/column indices will not affect the joint distribution. It turns out that we can represent ξ precisely as follows:

Proposition 6. *A discrete random measure ξ on $[0,1] \times \mathbb{R}_+$ is separately exchangeable if and only if*

$$\xi = \sum_{n,k} f_n(\vartheta_k)\delta_{\tau_n,\sigma_k} + \sum_{m,k} g_m(\vartheta_k)\delta_{\rho_{mk},\sigma_k}, \tag{4.3}$$

almost surely for some random measurable functions $f_n, g_m \geq 0$ on $\mathbb{R}_+^2$, a unit rate Poisson process $\{(\vartheta_k, \sigma_k)\}$ on $\mathbb{R}_+^2$, and independent $U(0,1)$ arrays (τ_n) and (ρ_{mk}).

Proof. This follows from the general representation theorem for separately exchangeable random measures on $[0,1] \times \mathbb{R}_+$ [Kal06] by removing the non-atomic parts. Details are given in the appendix.

□

We briefly look at the two parts of this representation:

1. $\sum_{n,k} f_n(\vartheta_k)\delta_{\tau_n,\sigma_k}$: This is the part we are interested in. Correlations are learned through coupling of random functions f_n with a Poisson process.

2. $\sum_{m,k} g_m(\vartheta_k)\delta_{\rho_{mk},\sigma_k}$: This part is less important since the double index in ρ_{mk} means each row (object) slice $\xi(\{\rho_{mk}\},\cdot)$ contains at most one atom. We drop this part in our model.

Thus, we can represent $\xi = \sum_{n,k} f_n(\vartheta_k)\delta_{\tau_n,\sigma_k}$ as a coupling of a 2d Poisson process (ϑ_k, σ_k) and random functions f_n. As mentioned in Section 1, we derive $Z_{nk} = f_n(\vartheta_k)$. Since we model the entire population through Z by a random measure embedding, we call our model *population random measure embedding* (PRME).

4.2.2 Construction via completely random measures

Once we have a representation for Z_{nk}, we still need to guarantee series convergence $\sum_k Z_{nk} = \sum_k f_n(\vartheta_k) < \infty$. This is not obvious, since ϑ_k spans uniformly on $\mathbb{R}_+$. One remedy is to introduce a transformation $\widetilde{\vartheta}_k = T(\vartheta_k)$ that maps almost every $\widetilde{\vartheta}_k$ close to zero, leaving only finite number of $\widetilde{\vartheta}_k$ above any positive threshold. The method to introduce such a transformation T is via completely random measures (CRM) [Kin67]. In the appendix, we show the construction of T via CRMs. In addition, we show that the well-known Indian buffet process [GG06, GG11], its extensions [TG09], hierarchical Dirichlet processes (HDP) [TJBB05] and the discrete infinite logistic normal distribution (DILN) [PWB+12] are instances of population random measure embeddings. However, these models have restrictions in their model capacity. For example, [PWB+12] relies on a linear kernel to model correlations and there is no obvious extension to complex kernels. As we will show, a PRME can be more flexible by using nonlinear object-specific functions f_n such as deep neural networks.

4.3 An illustration on topic modeling

4.3.1 The model

In a topic model, we use Z_n to represents an un-normalized discrete distribution over topics, where Z_{nk} is the strength of topic k for document n. We use a PRME to model Z_{nk}, with the following

construction,

$$
\begin{aligned}
&Z_{nk} \sim \text{Gamma}(\beta p_k, \exp(f(h_n, \ell_k))), \\
&p_k = V_k \prod_{k'=1}^{k-1} (1 - V_{k'}), \quad V_k \sim \text{Beta}(1, \alpha), \\
&h_n \sim \mathcal{N}(0, aI), \quad \ell_k \sim \mathcal{N}(0, bI), \\
&f(h_n, \ell_k) \sim \mathcal{N}(\mu_f(h_n, \ell_k), \sigma_f^2(h_n, \ell_k)).
\end{aligned} \tag{4.4}
$$

We now explain how Equation (4.4) relates to the original PRME equation $Z_{nk} = f_n(\vartheta_k)$, via four steps.

1. $\boxed{f_n(\vartheta_k) \;\to\; f(h_n, \vartheta_k)}$

   We use a parametric function $f(h_n, \cdot)$ to represent $f_n(\cdot)$, where f is a random function, and h_n is an observation-specific random vector. This decomposition is necessary, since we model f as a normal distribution parameterized by *decoder networks* μ_f, σ_f^2, and h_n as the output of an *inference network*.

2. $\boxed{f(h_n, \vartheta_k) \;\to\; f(h_n, \widetilde{\vartheta}_k)}$

   We transform $\widetilde{\vartheta}_k = T(\vartheta_k)$ by transforming the original Poisson process (θ_k, σ_k) to a hierarchical Gamma process [TJBB05, WPB11b]. Then we use a stick-breaking construction over $\widetilde{\vartheta}_k$ [Set94b], where $\widetilde{\vartheta}_k \sim \text{Gamma}(\beta p_k, 1)$. β is a hyperparameter and p_k is generated by the second line of Equation (4.4).

3. $\boxed{f(h_n, \widetilde{\vartheta}_k) \;\to\; f(h_n, \widetilde{\vartheta}_k, \ell_k)}$

   We augment $\widetilde{\vartheta}_k$ to $(\widetilde{\vartheta}_k, \ell_k)$ to introduce extra randomness via ℓ_k. This operation is equivalent to augmenting the original 2d Poisson process (θ_k, σ_k) to a higher dimensional Poisson process $(\theta_k, \sigma_k, \ell_k)$.

4. $\boxed{f(h_n, \widetilde{\vartheta}_k, \ell_k) \;\to\; \widetilde{\vartheta}_k \cdot \exp(f(h_n, \ell_k))}$

   We represent $f(h_n, \widetilde{\vartheta}_k, \ell_k)$ as $\widetilde{\vartheta}_k \cdot \exp(f(h_n, \ell_k))$ and assign priors for h_n and ℓ_k (line 3 in Equation (4.4)). We get Equation (4.4) by absorbing $\exp(f(h_n, \ell_k))$ into the Gamma scale parameter.

In our construction, series convergence $\sum_{k=1}^{\infty} Z_{nk} < \infty$ can be achieved by bounding μ_f and σ_f^2 through a truncation layer in the decoder network. Given Z_n, we sample words in a document, X_{nm} for $m \in [M_n]$, by first sampling its topic assignment $C_{nm} \sim \text{Disc}(\frac{Z_{n\cdot}}{\sum_k Z_{nk}})$, and then sampling the word

from that topic, $X_{nm} \sim \text{Disc}(\theta_{C_{nm}})$, with topic prior $\theta_k \sim \text{Dir}(\gamma_0)$. We recall that in topic models, θ_k (topic k) is a discrete distribution over the vocabulary.

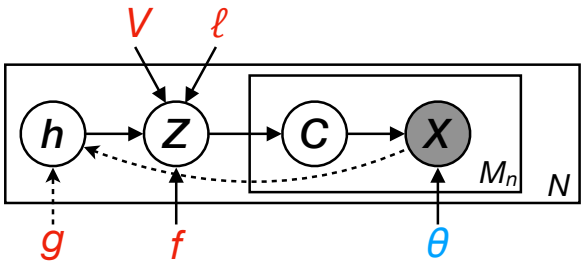


Figure 4.3: (a) Graphical representation of our proposed model. Solid arrows represent the generative process and dashed arrows show the VAE part of the posterior. We organize local parameters that belong to a document/word into boxes and remove all sub-indices. We use stochastic natural gradient ascent for θ and use stochastic gradient ascent for $[\ell, V, g, f]$.

4.3.2 Amortized variational inference

Assume we have N documents and the posterior is truncated to K topics. The joint likelihood is

$$p(\ell, V, \theta, h, Z, C, X) = \prod_{k=1}^{K} p(\ell_k)p(V_k)p(\theta_k)\prod_{n=1}^{N}\Bigg[p(h_n) \prod_{k=1}^{K} p(Z_{nk}|V, h_n, \ell_k)\prod_{m=1}^{M_n} p(C_{nm}|Z_n)p(X_{nm}|C_{nm}, \theta)\Bigg]. \tag{4.5}$$

We use variational inference to approximate the model posterior by optimizing the variational objective function

$$\max_q \mathcal{L} = \max_q \mathbb{E}_q\Big[\ln \frac{p(\ell, V, \theta, h, Z, C, X)}{q(\ell, V, \theta, h, Z, C)}\Big], \tag{4.6}$$

where we restrict q to the factorized family

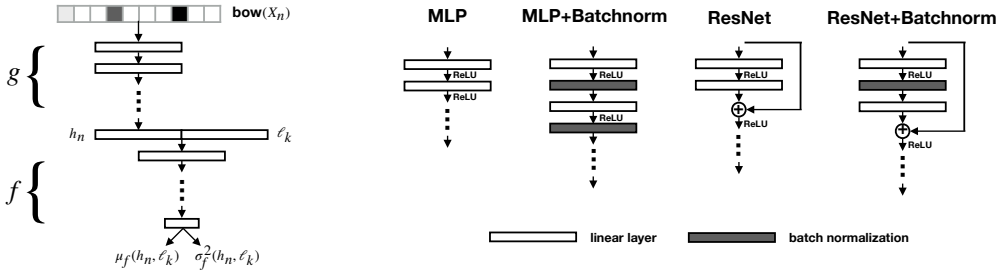


Figure 4.4: (b) Left: The architecture we used in our experiments. Right: Various layer designs.

$$q(\ell, V, \theta, h, Z, C) = \prod_{k=1}^{K} q(\ell_k) q(V_k) q(\theta_k) \prod_{n=1}^{N} \left[q(h_n | X_n) \prod_{k=1}^{K} q(Z_{nk}) \prod_{m=1}^{M_n} q(C_{nm}) \right]. \tag{4.7}$$

Further, for global variables we let

$$q(\ell_k) = \delta_{\widehat{\ell}_k}, \quad q(V_k) = \delta_{\widehat{V}_k}, \quad q(\theta_k) = \text{Dir}(\gamma_k). \tag{4.8}$$

For local variables, we introduce an inference network g and let $q(h_n | X_n) = \delta_{g(X_n)}$. For the remaining variables

$$q(Z_{nk}) = \text{Gam}(a_{nk}, b_{nk}), \; q(C_{nm}) = \text{Disc}(\phi_{nm}). \tag{4.9}$$

We use coordinate ascent to update q. Each of these updates is guaranteed to improve the objective when the gradient descent step size is small enough [Nes13]. More details are given in the appendix.

For $q(Z_{nk})$, we maximize a lower bound for $\mathcal{L}$ similar to [PWB+12], giving updates

$$\begin{aligned} a_{nk} &= \beta \widehat{p}_k + \sum_{m=1}^{M_n} \phi_{nm}(k), \\ b_{nk} &= 1 / \Big(\mathbb{E}\Big[\exp(-f(h_n, \ell_k)) \Big] + \frac{M_n}{\varepsilon_n} \Big), \end{aligned} \tag{4.10}$$

where $\varepsilon_n = \sum_{k=1}^{K} \mathbb{E}[Z_{nk}]$.

For $q(C_{nm})$ and $q(\theta)$, we have respective updates

$$\phi_{nm}(k) \propto \exp\Big(\mathbb{E}[\ln \theta_{k, X_{nm}}] + \mathbb{E}[\ln Z_{nk}] \Big), \tag{4.11}$$

$$\gamma_{kd} = \gamma_0 + \sum_{n=1}^{N} \sum_{m=1}^{M_n} \phi_{nm}(k) \cdot \mathbb{1}(X_{nm} = d). \tag{4.12}$$

For $[\ell, V, g, f]$, we do gradient ascent on $\mathcal{L}$. Batch variational inference can be done via coordinate ascent by iteratively updating the above variables. Dependencies among variables are shown in Figure 4.3.

For stochastic inference, in each global iteration we sample a subset $N_t \subset [N]$ and compute the noisy variational objective

$$\mathcal{L}_t = \mathbb{E}\Big[\ln p(\ell, V, \theta) \Big] + \frac{N}{|N_t|} \sum_{n \in N_t} \mathbb{E}\Big[\ln p(h_n, Z_n, C_n, X_n) \Big] + \mathbb{H}\Big[q(\theta) \Big] + \frac{N}{|N_t|} \sum_{n \in N_t} \mathbb{H}\Big[q(Z_n, C_n) \Big]. \tag{4.13}$$

Optimizing local variables Z, C can be done via closed-form updates exactly as in the batch case.

Algorithm 5 Stochastic inference algorithm

for $t = 1, 2, \ldots$ **do**
  Sample a subset $N_t \subset [N]$
  Update local variables
  while not converge **do**
    Closed-form update $q(Z_n)$ for $n \in N_t$. Eq. (4.10)
    Closed-form update $q(C_n)$ for $n \in N_t$. Eq. (4.11)
  end while
  Update global variables
  Noisy natural gradient step for $q(\theta)$. Eq. (4.14)
  Noisy gradient step for ℓ, V, g, f. Eq. (4.15)
end for

Table 4.1: Perplexity result for text data sets with different dictionary sparsity levels controlled by γ_0.

Model	New York Times				20Newsgroups				NeurIPS			
	$\gamma_0=0.2$	$\gamma_0=0.4$	$\gamma_0=0.6$	$\gamma_0=0.8$	$\gamma_0=0.2$	$\gamma_0=0.4$	$\gamma_0=0.6$	$\gamma_0=0.8$	$\gamma_0=0.2$	$\gamma_0=0.4$	$\gamma_0=0.6$	$\gamma_0=0.8$
HDP	2436.51	2464.74	2482.61	2501.82	5317.68	5845.90	6294.68	6665.68	1973.39	1962.90	1981.83	2009.58
DILN	2231.16	2295.12	2418.16	2509.24	5164.93	5732.12	6143.64	6389.99	1853.89	1902.88	1944.90	**1947.94**
PRME	**2203.00**	**2247.25**	**2299.60**	**2338.38**	**5102.08**	**5531.04**	**5878.39**	**5975.12**	**1753.61**	**1850.37**	**1917.21**	1953.85

For the other parameters we use stochastic gradient methods. Let $\rho^{(t)} \propto (t_0 + t)^{-\kappa}$ be the step size with some constant t_0 and $\kappa \in (0.5, 1]$. We apply the stochastic natural gradient method [HBWP13b] for θ

$$\begin{aligned} \widetilde{\gamma}_{kd}^{(t)} &= \gamma_0 + \sum_{n \in N_t} \sum_{m=1}^{M_n} \phi_{nm}(k) \cdot \mathbb{1}(X_{nm} = d), \\ \gamma_{kd}^{(t)} &= (1 - \rho^{(t)}) \gamma_{kd}^{(t-1)} + \rho^{(t)} \widetilde{\gamma}_{kd}^{(t)}. \end{aligned} \tag{4.14}$$

and stochastic gradient method for the rest,

$$[\ell, V, g, f]^{(t)} = [\ell, V, g, f]^{(t-1)} + \rho^{(t)} \nabla_{[\ell, V, g, f]} \mathcal{L}_t. \tag{4.15}$$

Since in each iteration we only do one gradient step, the cost is low. Note that through the variational autoencoder (VAE) [KW13] we transfer local updates for h_n to global update for g, which will significantly speed-up inference. We summarize the stochastic inference algorithm in Algorithm 5.

4.3.3 Network architectures

The flexibility of our model comes from the inference and decoder networks g and f. As we show in the experiments, these allow us to learn complex non-linear "paintboxes" in order to capture complex topic correlations. Since optimizing over deep neural networks is still a challenging problem in theory, we design our networks with architectures that work well in practice. Rather than directly applying

Table 4.2: Dataset description.

Corpus	# train	# test	# vocab	# tokens
New York Times	5,000	500	8,000	1.4M
20Newsgroups	11,269	7,505	53,975	2.2M
NeurIPS	2,183	300	14,086	3.3M

multilayer perceptrons [RHW85], we instead use more complex layer designs such as batch normalization [IS15] and deep residual networks (ResNet) [HZRS16] to speed-up training. For inference network g, we use the bag-of-words representation of X_n as the input feature. For decoder network f, we concatenate $h_n = g(X_n)$ and ℓ_k as inputs. Detailed architecture design is shown in Figure 4.4.

4.4 Experiments

Table 4.3: Network layer configurations for New York Times dataset.

Depth	Inference Network	Decoder Network
2 layers	$[8000 \times d_h]$	$[(d_h + d_\ell) \times 80]$ $[80 \times 2]$
4 layers	$[8000 \times 1000]$ $[1000 \times d_h]$	$[(d_h + d_\ell) \times 80]$ $[80 \times 80]$ $[80 \times 2]$
6 layers	$[8000 \times 1000]$ $[1000 \times 1000]$ $[1000 \times d_h]$	$[(d_h + d_\ell) \times 80]$ $[80 \times 80]$ $[80 \times 80]$ $[80 \times 2]$
8 layers	$[8000 \times 1000]$ $[1000 \times 1000]$ $[1000 \times 1000]$ $[1000 \times d_h]$	$[(d_h + d_\ell) \times 80]$ $[80 \times 80]$ $[80 \times 80]$ $[80 \times 80]$ $[80 \times 2]$

4.4.1 Batch experiments

We show empirical results on three text datasets: a 5K subset of New York Times, 20Newsgroups, and NeurIPS. Their basic statistics are shown in Table 4.2. For each test document X_n, we do a $90\%/10\%$ split into training words $X_{n,TR}$ and testing words $X_{n,TS}$. The perplexity is calculated based on the prediction of $X_{n,TS}$ given the model and $X_{n,TR}$,

$$\text{perplexity} = \exp\Big(-\frac{\sum_{m \in X_{n,TS}} \ln p(X_{nm}|X_{n,TR})}{|X_{n,TS}|}\Big). \tag{4.16}$$

Table 4.4: Perplexity result for various network depths.

Depth	MLP	MLP+BN	ResNet	ResNet+BN
2 layers	2325.84	2327.81	N/A	N/A
4 layers	2228.62	2203.00	2214.02	2195.72
6 layers	2219.06	**2184.44**	2202.79	2194.74
8 layers	**2196.35**	2195.68	**2199.07**	**2184.56**

Table 4.5: Perplexity result for various size of h_n/ℓ_k.

Hidden Size	MLP	MLP+BN	ResNet	ResNet+BN
$d_h = d_\ell = 2$	2287.40	2258.97	2265.53	2256.84
$d_h = d_\ell = 5$	2245.43	2243.26	2231.54	2225.64
$d_h = d_\ell = 10$	**2220.82**	2217.65	2227.04	2199.73
$d_h = d_\ell = 20$	2228.62	**2203.00**	**2214.02**	**2195.72**

Lower perplexity means better predictive performance.

In Table 4.1, we compare three Bayesian nonparametric models: hierarchical Dirichlet process (HDP) [TJBB05], discrete infinite logistic normal (DILN) [PWB+12], and our population random measure embedding (PRME) using 4-layer MLP with batch normalization.[1]

We tune γ_0 and fix the truncation level $K = 100$ and set the $a = 1, b = 1, \alpha = 1, \beta = 5$ for fair comparisons. All gradient updates are done via Adam [KB14] with learning rate 10^{-4}. As Table 4.1 shows, PRME consistently perform better than HDP and DILN. Where DILN was designed to outperform HDP by learning topic correlation structure, PRME improves upon DILN by learning a more complex kernel structure.

Since PRME encodes complex correlation patterns with a neural network, we further consider the influence of network architecture on perplexity for the New York Time dataset. We compare four layer designs: multilayer perceptron (MLP), MLP with batch normalization (MLP+BN), ResNet, and ResNet with batch normalization (ResNet+BN); see Figure 4.4 for details. In Table 4.4 and Table 4.5, we separately tune the depth of each network and the hidden size of h/ℓ while holding other parameters fixed. The details of layer sizes can be found in Table 4.3. We observe that the perplexity result tend to be better when we scale up the network depth/width. Batch normalization and ResNet both improve performance.

[1]The number of layers includes inference network and decoder network. We ignore the last layer of the decoder network.

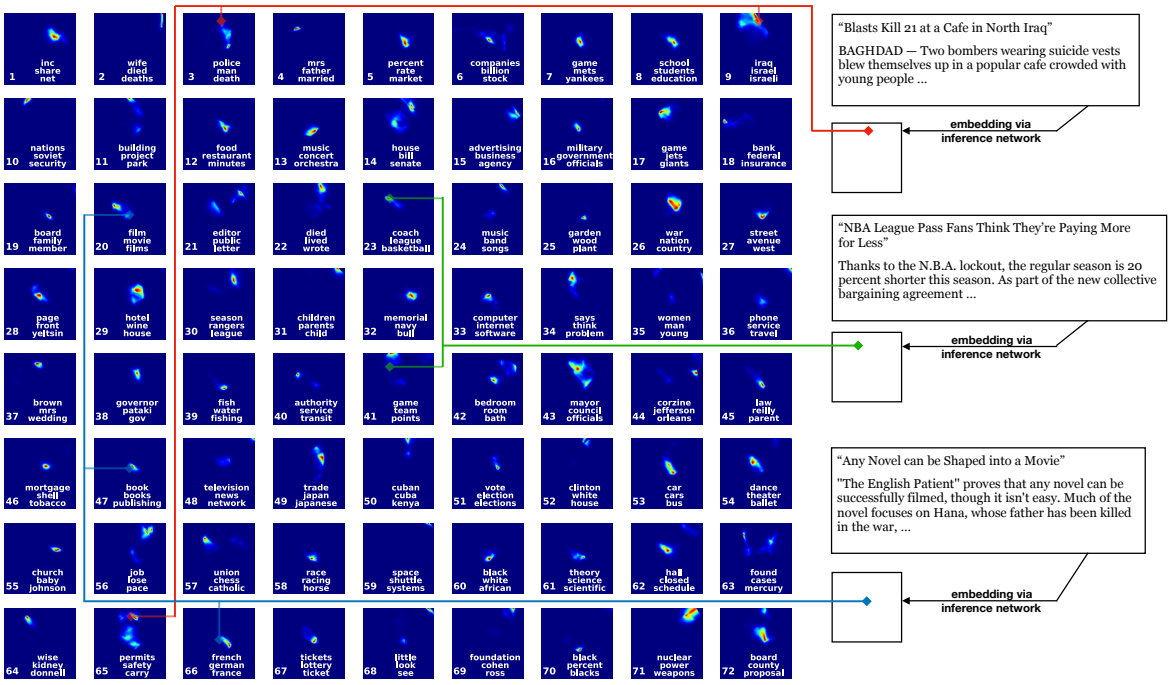


Figure 4.5: A paintbox demonstration of salient topics learned from the one million New York Times dataset. In each paintbox on the LHS, pixel (x, y) represents the topic strength $Z_{(x,y),k}$ as a function of $h_{(x,y)}$ for a particular topic k. We also show embeddings of three articles in the same space, as well as their projection onto selected paintboxes. Each article is connected to its most-used topics.

4.4.2 Online experiments

For the larger one million New York Times dataset, we show "topic paintboxes" learned with stochastic PRME in Figure 4.5.[2] In Figure 4.5, each paintbox corresponds to one topic whose top words are displayed inside the box. The color of a pixel (x, y) in the k-th paintbox ranges from blue (small value) to red (large value) and represents mean topic strength $\mathbb{E}[Z_{(x,y),k}] = \beta \widehat{p}_k \mathbb{E}[\exp(f(h_{(x,y)}, \ell_k))]$ as a function of $h_{(x,y)}$ for topic k. To define $h_{(x,y)}$ for 2d visualization, we collect the empirical embeddings $H = [h_1, \ldots, h_N]^\top = [g(X_1), \ldots, g(X_N)]^\top$ on a subset of data, subtract their mean m_h, and use the SVD to select the two most informative directions $\widetilde{h}_1, \widetilde{h}_2$ with singular values s_1, s_2. Then we plot each paintbox as the function value $\mathbb{E}[Z_{(x,y),k}] = \beta \widehat{p}_k \mathbb{E}[\exp(f(m_h + x s_1 \widetilde{h}_1 + y s_2 \widetilde{h}_2, \ell_k))]$ by tuning $(x, y) \in [-0.2, 0.2]^2$.

The correlation between topics can be read out from the paintboxes. Those paintboxes that have overlapping salient regions tend to be more correlated. For example, topic 13 [music, concert, orchestra], topic 20 [film, movie, films], and topic 47 [book, books, publishing] share a salient region, which gives

[2] We set $t_0 = 100$, $\kappa = 0.75$ and use a 6-layer MLP.

a third-order positive correlations over those topics. In principle, the paintbox can explain arbitrary order correlations as the neural network complexity increases. We observe that each paintbox in Figure 4.5 consists of multiple contiguous salient regions. This is due to the smoothness of neural networks, since $g(X_{n_1}) \approx g(X_{n_2})$ when X_{n_1} and X_{n_2} share similar words. Also, the various "modes" in each paintbox demonstrate the greater flexibility of neural networks in explaining different contexts of a topic.

In Figure 4.5, we also display three documents with their embeddings h_n projected onto the 2d paintbox space. Each embedding hits salient regions of several paintboxes. Thus, each document can be interpreted as a mixture of these corresponding topics. We again note that we only display the paintbox in 2d via post-processing, but the actual paintbox is in 20 dimension; a higher-dimensional paintbox can be more complex than what is shown.

We can compare the difference between paintboxes for PRME in Figure 4.5 and paintboxes for binary random measures in Figure 4.1. First, the paintbox for PRME is real-valued, so it is natural to use smooth functions to model it. In the binary case the paintbox is zero/one valued; in this case one can apply a threshold function over the PRME paintbox to binarize it. Second, in contrast to the binary paintbox, each PRME paintbox is unbounded. We control the area of this salient region through regularization.

Figure 4.6(a) demonstrates the perplexity of DILN and PRME with various decay speed κ on a held-out test set of size 3K. PRME converges after seeing one million documents, and it performs better than DILN. Also, online learning is much more efficient than batch learning with various training data size, as shown in Figure 4.6(b). In Figure 4.6(c), we compare run times for updating local parameters ($[Z, C]$ for PRME) and global parameters ($[\theta, \ell, V, g, f]$ for PRME) with batch size 500. Since the cost is very imbalanced between local and global, for demonstration purpose we compare the cost between five local iterations and one global iteration. In our experiments, local updates requires around 20 iterations to converge. Compared with DILN, PRME costs much less in local and costs more in global updates, since it uses the VAE to transfer local updates for h_n into global updates for g. The extra global cost ($\sim$0.35s) is significantly smaller than the reduced local cost ($\sim$4s), even when using a deep network architecture. Finally, Figure 4.6(d) demonstrates the usage proportion for all topics. PRME tends to use a subset of the 100 available topics in the truncated posterior, indicating use by the model of this nonparamteric feature.

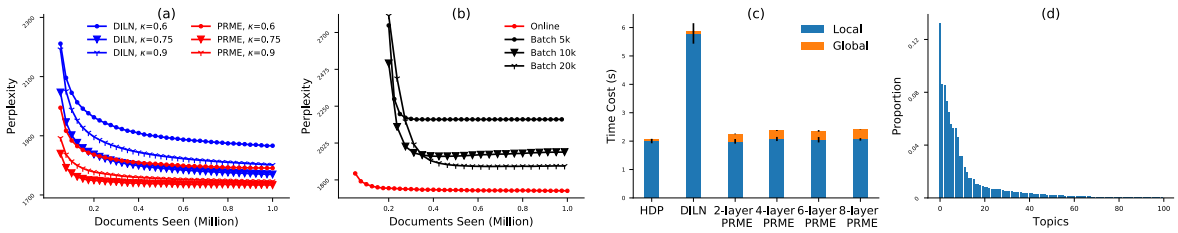


Figure 4.6: (a) Online performance comparisons between DILN and PRME. (b) Online versus batch. (c) Time cost comparison between updating local and global variables. (d) Ranked topic usage proportions in the posterior, indicating nonparametric functionality.

4.5 Discussion

4.5.1 Connections with other random objects

Another view is to treat $(Z_{nk})_{n\in[N],k\in[K]}$ as a bipartite graph over objects $[N]$ and atoms (features) $[K]$ with edge strength Z_{nk}. An important topic in random graph theory is to study the total strength of edges $|E| = \sum_{n\in[N],k\in[K]} \mathbb{E}[Z_{nk}]$ asymptotically as a function of N. There has been extensive work on random graphs, networks, and relational models [RT+08, MJG09, Car12, LOGR12, VR15, CCB16, LJC16, CD17, CR17, CF17], but these methods mainly focus on dense graphs where $|E| \sim \mathcal{O}(N^2)$, and sparse graphs where $|E| \sim \mathcal{O}(N^{1+\alpha})$ with $0 < \alpha < 1$ or $|E| \sim \mathcal{O}(N \log N)$. Our method offers a new solution to *extremely sparse hidden graphs* where $|E| \sim \mathcal{O}(N)$, by coupling random functions and a Poisson process. Our solution cannot be trivially derived from previous representations in sparse/dense graphs. There is a developed probability theory building connections between exchangeable binary random measures and functions on combinatorial structures among atoms [Pit95, Pit06, BPJ+13, BMPJ15, HR+16, CCB+18].

Our topic model construction is motivated by previous research on dependent random measures [ZYS+11, PWB+12, CRBT13, FFRW13, ZP15b, ZP16]. Our focus is to place *mild exchangeability assumptions* on a population random measure ξ and derive a very general random function model through representation theorems. Hence our use of neural networks to achieve this task. We mention that our method can also be adapted to non-exchangeable settings.

4.5.2 Deep hierarchical Bayesian models

One can scale up model capacity by stacking multiple one-layer Bayesian nonparametric models such as Dirichlet processes [TJBB05], beta processes [TJ07], and Gamma processes [ZCC15, Zho18].

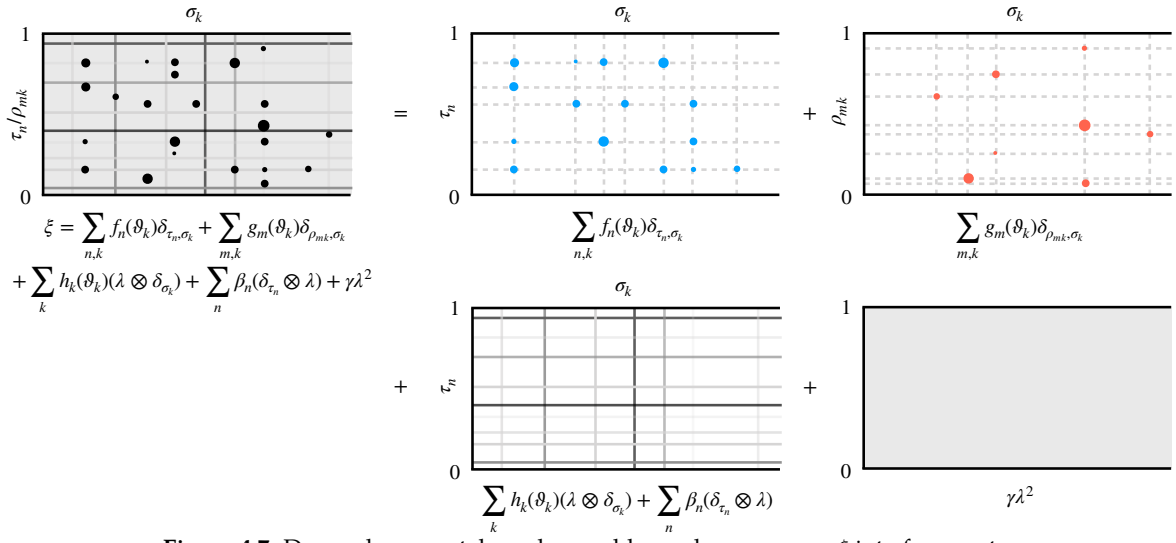


Figure 4.7: Decouple separately exchangeable random measure ξ into four parts.

Population random measure embedding uses a different strategy by constructing random measures as a coupling of random functions with a single Poisson process. In this way, we transfer all the model complexity into random functions f_n. Using amortized variational inference, we transfer posterior inference of discrete random measures into optimizing neural networks, which is much more efficient.

4.5.3 Posterior inference bottleneck

Efficient posterior inference is essential in Bayesian nonparametric methods where conjugacy often does not hold [BWJ14, ZGP16b]. In principle, one can apply a simple prior on Z and still rely on accurate posterior inference to resolve the structure. However, posterior inference for random measures is not simple because complex correlations among atoms leads to slow MCMC mixing. Instead, one can approximate the posterior using variational methods [BKM17] and try to learn a q distribution with good approximation quality [PBJ12, HB15, RTB16, TRB17]. Our method introduced a structured prior to regularize variational inference. Empirical results showed that we get an interpretable posterior.

Appendix

Proof of Proposition 5.

Proof. From [Ald85, Hoo79, OR15b], we can represent every separately exchangeable infinite binary matrix $Z = (Z_{nk})$ if and only if it can be represented as follows: There is a random function W :

$[0,1]^2 \to [0,1]$ such that

$$(Z_{nk}) \stackrel{d}{=} (\mathbb{1}(R_{nk} < W(U_n, V_k))). \tag{4.17}$$

Thus, one can reconstruct Z by first sample $W, (U_n), (V_k)$ and then sample $Z_{nk}|\ W, (U_n), (V_k) \sim$ Bernoulli$(W_{U_n V_k})$ through independent coin flips. It is straightforward to prove that the finite row sum assumption can only be satisfied when $\int_{[0,1]^2} W(u,v)dudv = 0$. When that happens, $Z = \mathbf{0}$ almost surely. □

Proof of Proposition 6.

Proof. The representation theorem in Proposition 6 is immediate from a more general result of separately exchangeable random measures. We temporarily reload notations h_k, β_n.

Theorem 3 [Kal06]. *A random measure ξ on $[0,1] \times \mathbb{R}_+$ is separately exchangeable if and only if almost surely*

$$\begin{aligned}\xi = &\underbrace{\sum_{n,k} f_n(\vartheta_k)\delta_{\tau_n,\sigma_k} + \sum_{m,k} g_m(\vartheta_k)\delta_{\rho_{mk},\sigma_k}}_{\textit{point masses}} \\ &+ \underbrace{\sum_k h_k(\vartheta_k)(\lambda \otimes \delta_{\sigma_k}) + \sum_n \beta_n(\delta_{\tau_n} \otimes \lambda)}_{\textit{line measures}} \\ &+ \underbrace{\gamma\lambda^2}_{\textit{diffuse measure}}\ ,\end{aligned} \tag{4.18}$$

for some measurable functions $f_n, g_m, h_k \geq 0$ on $\mathbb{R}_+^2$, a unit rate Poisson process $\{(\vartheta_k, \sigma_k)\}$ on $\mathbb{R}_+^2$, some independent $U(0,1)$ arrays (τ_n) and (ρ_{mk}), an independent set of random variables $\beta_n, \gamma \geq 0$, and the Lebesgue measure λ. The latter can then be chosen to be non-random if and only if ξ is extreme.

The representation theorem consists of three parts: point masses, line measures, and a diffuse measure. We select the point masses part for discrete separately exchangeable random measures. Decomposition of the entire measure ξ is demonstrated in Fig 4.7. □

Discussion on Section 4.2.2. Existing models as special cases of PRME model.

Let $\xi = \sum_{n,k} f_n(\vartheta_k)\delta_{\tau_n,\sigma_k}$ be our PRME model. We focus on a specific object n, remove the redundant τ_n, and directly work on random measures on Θ. This transformation let us be on the same page of other research on completely random measures.

We have $\xi_n = \sum_k f_n(\vartheta_k)\delta_{\theta_k}$ be a population random measure embedding model, where (ϑ_k, θ_k) is a Poisson process on $\mathbb{R}_+ \times \Theta$ with mean measure $p(\theta)d\vartheta d\theta$. The according CRM is $\lambda = \sum_k \widetilde{\vartheta}_k \delta_{\theta_k}$ with LÃľvy measure $\nu(d\widetilde{\vartheta}, d\theta) = \mu(\widetilde{\vartheta})p(\theta)d\widetilde{\vartheta}d\theta$. Assume the tail function $T(\widetilde{\vartheta}) = \nu((\widetilde{\vartheta}, \infty), \Theta)$ is invertible. One can do a transformation between atoms by $(\vartheta_k, \theta_k) \to (T^{-1}(\vartheta_k), \theta_k) = (\widetilde{\vartheta}_k, \theta_k)$. The following examples are just special cases of this transformation, as we shall see.

IBP and extensions

The Indian buffet process (IBP) take a particular form $\xi_n = \sum_k f \circ T^{-1}(\vartheta_k)\delta_{\theta_k}$, where f are independent Bernoulli random variables with success rate $T^{-1}(\vartheta_k)$. IBP uses a particular transformation $T^{-1}(\vartheta_k) = e^{-\vartheta_k}$ [TJ07]. [TG09] gives a power-law extension of IBP with three parameters (3IBP) with an application in language models. However, 3IBP does not enjoy an analytical form for T^{-1}. But we can safely work on the CRM directly, given the generality of the existence of T^{-1} [OW11]. One can observe that the sampling function $f \circ T^{-1}$ does not change with n. This is the main limitation for IBP and 3IBP. A MCMC sampling solution can be found in [GG11, TG09].

Correlated random measures

The key restrictions of IBP and 3IBP is that $\mathbb{E}[\xi_n(\{\theta_{k_1}\}) \cdot \xi_n(\{\theta_{k_2}\})|\widetilde{\vartheta}] = \mathbb{E}[\xi_n(\{\theta_{k_1}\})|\widetilde{\vartheta}] \cdot \mathbb{E}[\xi_n(\{\theta_{k_2}\})|\widetilde{\vartheta}]$. In order to model feature correlations, [PWB+12] model $f_n(\vartheta_k)$ as exchangeable random functions. The extra randomness besides $\widetilde{\vartheta}$ can be modelled by augmenting the Poisson process $(\widetilde{\vartheta}_k, \theta_k)$ on $\mathbb{R}_+ \times \Theta$ to higher dimension $(\ell_k, \widetilde{\vartheta}_k, \theta_k)$ on $\mathbb{R}^d \times \mathbb{R}_+ \times \Theta$ with mean measure $\nu(d\ell, d\widetilde{\vartheta}, d\theta) = p(\ell)\mu(\widetilde{\vartheta})p(\theta)d\ell d\widetilde{\vartheta}d\theta$. The discrete infinite logistic normal distribution (DILN) [PWB+12] further proposes an example $\xi_n = \sum_k Z_{nk}(\beta\widetilde{\vartheta}_k, \exp(-h_n(\ell_k)))\delta_{\sigma_k, \tau_n}$, where $h_n(\cdot) \sim \mathrm{GP}(m(\cdot), K(\cdot, \cdot))$ and Z_{nk} is a gamma distribution parameterized by its shape and scale parameters. However, DILN is restricted to use linear kernels, which is very restrictive. [RB18] proposed general correlated random measures with examples for the binary, discrete, and continuous cases.

Section 4.3. Detailed derivations.

The variational objective function can be decoupled as

$$\mathcal{L} = \sum_{k=1}^{K} \mathbb{E}\Big[\ln p(\ell_k) + \ln p(V_k) + \ln p(\theta_k)\Big]$$

$$+\sum_{n=1}^{N}\mathbb{E}\Big[\ln p(h_n)\Big]+\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}\Big[\ln p(Z_{nk}|V,h_n,\ell_k)\Big]$$
$$+\sum_{n=1}^{N}\sum_{m=1}^{M_n}\mathbb{E}\Big[\ln p(C_n^{(m)}|Z_n)+\ln p(X_n^{(m)}|C_n^{(m)},\theta)\Big]$$
$$+\mathbb{H}\Big[q(\ell,V,\theta,h,Z,C)\Big]. \tag{4.19}$$

We expand each term in Eq. (4.19) as follows.

$$\mathbb{E}[\ln p(\ell_k)]=\ln p(\widehat{\ell}_k)=-\frac{r_\ell\ln(2\pi b)}{2}-\frac{\widehat{\ell}_k^\top\widehat{\ell}_k}{2b}. \tag{4.20}$$

$$\mathbb{E}[\ln p(V_k)]=\ln p(\widehat{V}_k)=\ln\alpha+(\alpha-1)\ln(1-\widehat{V}_k). \tag{4.21}$$

$$\mathbb{E}[\ln p(\theta_k)]=\ln\Gamma(D\gamma_0)-D\ln\Gamma(\gamma_0)+\sum_{d=1}^{D}(\gamma_0-1)\mathbb{E}[\ln\theta_{kd}],\ \text{where}\ \mathbb{E}[\ln\theta_{kd}]=\psi(\gamma_{kd})-\psi(\sum_{d=1}^{D}\gamma_{kd}). \tag{4.22}$$

$$\mathbb{E}[\ln p(h_n)]=\ln p(\widehat{h}_n)=-\frac{r_h\ln(2\pi a)}{2}-\frac{\widehat{h}_n^\top\widehat{h}_n}{2a}. \tag{4.23}$$

$$\mathbb{E}[\ln p(Z_{nk}|V,h_n,\ell_k)]=-\ln\Gamma(\beta\widehat{p}_k)-\beta\widehat{p}_k\mathbb{E}[f(h_n,\ell_k)]+(\beta\widehat{p}_k-1)\mathbb{E}[\ln Z_{nk}]-\mathbb{E}[Z_{nk}]\mathbb{E}[\exp(-f(h_n,\ell_k))],$$

$$\text{where}\ \mathbb{E}[f(h_n,\ell_k)]=\mu_f(\widehat{h}_n,\widehat{\ell}_k),$$

$$\mathbb{E}[\exp(-f(h_n,\ell_k))]=\exp\Big(-\mu_f(\widehat{h}_n,\widehat{\ell}_k)+\frac{1}{2}\sigma_f^2(\widehat{h}_n,\widehat{\ell}_k)\Big),$$

$$\mathbb{E}[\ln Z_{nk}]=\ln(b_{nk})+\psi(a_{nk}),\ \mathbb{E}[Z_{nk}]=a_{nk}b_{nk}. \tag{4.24}$$

$$\mathbb{E}[\ln p(C_{nm}|Z_n)]=\sum_{k=1}^{K}\phi_{nm}(k)\mathbb{E}\Big[\ln Z_{nk}-\ln\sum_{k'=1}^{K}Z_{nk'}\Big]. \tag{4.25}$$

$$\mathbb{E}[\ln p(X_{nm}|C_{nm},\theta)]=\sum_{k=1}^{K}\phi_{nm}(k)\mathbb{E}[\ln\theta_{k,X_{nm}}],\ \text{where}\ \mathbb{E}[\ln\theta_{k,X_{nm}}]=\psi(\gamma_{k,X_{nm}})-\psi(\sum_{d=1}^{D}\gamma_{kd}). \tag{4.26}$$

$$\mathbb{H}[q(\ell_k)]=0. \tag{4.27}$$

$$\mathbb{H}[q(V_k)]=0. \tag{4.28}$$

$$\mathbb{H}[q(\theta_k)]=\sum_{d=1}^{D}\ln\Gamma(\gamma_{kd})-\ln\Gamma(\sum_{d=1}^{D}\gamma_{kd})-\sum_{d=1}^{D}(\gamma_{kd}-1)\mathbb{E}[\ln\theta_{kd}]. \tag{4.29}$$

$$\mathbb{H}[q(h_n)]=0. \tag{4.30}$$

$$\mathbb{H}[q(Z_{nk})]=a_{nk}+\ln(b_{nk})+\ln\Gamma(a_{nk})+(1-a_{nk})\psi(a_{nk}). \tag{4.31}$$

$$\mathbb{H}[q(C_{nm})]=-\sum_{k=1}^{K}\phi_{nm}(k)\ln\phi_{nm}(k). \tag{4.32}$$

Variational inference for ℓ_k, V_k and network parameters can be done by directly plug-in and take gradients. Updating $q(\theta_k)$ and $q(C_{nm})$ follows the general variational update rule. Updating $q(Z_{nk})$ requires lower-bounding $\mathcal{L}$.

For ℓ, we use gradient ascent:

$$\nabla_\ell \mathcal{L} = \sum_{k=1}^{K} \nabla_\ell \mathbb{E}\Big[\ln p(\ell_k)\Big] + \sum_{n=1}^{N}\sum_{k=1}^{K} \nabla_\ell \mathbb{E}\Big[\ln p(Z_{nk}|V, h_n, \ell_k)\Big]. \tag{4.33}$$

For V, we use gradient ascent:

$$\nabla_V \mathcal{L} = \sum_{k=1}^{K} \nabla_V \mathbb{E}\Big[\ln p(V_k)\Big] + \sum_{n=1}^{N}\sum_{k=1}^{K} \nabla_V \mathbb{E}\Big[\ln p(Z_{nk}|V, h_n, \ell_k)\Big]. \tag{4.34}$$

For θ, we have a closed-form update:

$$\gamma_{kd} = \gamma_0 + \sum_{n=1}^{N}\sum_{m=1}^{M_n} \phi_{nm}(k) \cdot \mathbb{1}(X_{nm} = d) \tag{4.35}$$

For h_n, we update the inference network g:

$$\nabla_g \mathcal{L} = \sum_{n=1}^{N} \nabla_g \mathbb{E}\Big[\ln p(h_n)\Big] + \sum_{n=1}^{N}\sum_{k=1}^{K} \nabla_g \mathbb{E}\Big[\ln p(Z_{nk}|V, h_n, \ell_k)\Big] \tag{4.36}$$

For Z_{nk}, we maximize a lower bound for $\mathcal{L}$ similar as [PWB+12]. Related terms in $\mathcal{L}$ are:

$$\begin{aligned}\mathcal{L}(q(Z_{nk})) = (\beta\widehat{p}_k - 1 + \sum_{m=1}^{M_n} \phi_{nm}(k))\mathbb{E}[\ln Z_{nk}] \\ - \mathbb{E}[Z_{nk}]\mathbb{E}[\exp(-f(\widehat{h}_n, \widehat{\ell}_k))] \\ - M_n \mathbb{E}\Big[\ln \sum_{k'=1}^{K} Z_{nk'}\Big] + \mathbb{H}[q(Z_{nk})].\end{aligned} \tag{4.37}$$

The term that make closed-form update intractable is $\mathbb{E}[\ln \sum_{k'=1}^{K} Z_{nk'}]$. We use the bound:

$$\mathbb{E}[\ln \sum_{k'=1}^{K} Z_{nk'}] \leq \ln \varepsilon_n + \frac{\sum_{k'=1}^{K} \mathbb{E}[Z_{nk'}] - \varepsilon_n}{\varepsilon_n}. \tag{4.38}$$

This bound is correct for any $\varepsilon_n > 0$, and here we precompute $\varepsilon_n = \sum_{k=1}^{K} \mathbb{E}[Z_{nk}]$ and treat ε_n as a constant in the above equation. After Plugging-in the bound and some algebra, we solve $q(Z_{nk})$ as:

$$a_{nk} = \beta\widehat{p}_k + \sum_{m=1}^{M_n} \phi_{nm}(k),$$

$$1/b_{nk} = \mathbb{E}\Big[\exp(-f(h_n, \ell_k))\Big] + \frac{M_n}{\varepsilon_n}. \tag{4.39}$$

For decoder network f:

$$\nabla_f \mathcal{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} \nabla_f \mathbb{E}\Big[\ln p(Z_{nk}|V, h_n, \ell_k)\Big]. \tag{4.40}$$

For C_{nm}, we have a closed-form update:

$$\phi_{nm}(k) \propto \exp\Big(\mathbb{E}[\ln \theta_{k,X_{nm}}] + \mathbb{E}[\ln Z_{nk}]\Big). \tag{4.41}$$

Part III

Composing Deep Learning and Bayesian Nonparametric Methods in Time-Series Modeling

Chapter 5

Deep Bayesian Nonparametric Tracking

In this chapter and the following one, we present two applications of Bayesian nonparametric models to handle discrete patterns. Moreover, we observe that these models can work well with deep neural networks. Thus we are able to get the best of both worlds: train accurate models and at the same time figure out discrete patterns.

The first example is to handle time-series data often exhibit irregular behavior, making them hard to analyze and explain with a simple dynamic model. For example, information in social networks may show change-point-like bursts that then diffuse with smooth dynamics. Powerful models such as deep neural networks learn smooth functions from data, but are not as well-suited (in off-the-shelf form) for discovering and explaining sparse, discrete and bursty dynamic patterns. Bayesian models can do this well by encoding the appropriate probabilistic assumptions in the model prior. We propose an integration of Bayesian nonparametric methods within deep neural networks for modeling irregular patterns in time-series data. We use Bayesian nonparametrics to model change-point behavior in time, and a deep neural network to model nonlinear latent space dynamics. We compare with a non-deep linear version of the model also proposed here. Empirical evaluations demonstrates improved performance and interpretable results when tracking stock prices and Twitter trends.

5.1 Motivations

Irregular behaviors such as bursts and nonlinearity repeatedly show up in time-series data. For example, the past decades have seen several crashes of the stock market, in which a latent continuous-like process is interrupted by a change-point-like event [AM07]. Another example is information diffusion in social media, which exhibits initial bursts followed by smoother diffusion [GHFZ13, LBK09]. Traditional tracking models such as the linear Kalman filter [BW+01] are not ideally-suited for these tasks, in part because of their incompatibility with bursts and underlying nonlinearity [DJ09, ADH10]. More complex machine learning techniques such as deep neural networks [LBH15] usually fit smooth nonlinear functions to the data, and require more thought when handling bursty patterns in dynamic data.

On the other hand, Bayesian methods are good at handling discrete and bursty latent structures in data through explicit probabilistic modeling. Typical examples such as learning latent features [GG11] and latent state transitions [FSJW11] have proven useful in multiple applications. Moreover, Bayesian nonparametric (BNP) methods are naturally suited for large scale learning problems due to their infinite dimensional nature. Recent inference techniques such as online learning [HBWP13b], streaming learning [BPJ+13], and parallelization [GCWG15] further scale up these methods. To model complex and nonlinear structures, recent probabilistic generative models mimic the behavior of neural networks [RTCB15, SWZ16]. However, these models require carefully designed inference methods. Currently ongoing research largely focus on generalizing the scope of these models [RTB16, TRB17]. Other methods incorporate the power of neural networks in Bayesian models, such as the variational auto-encoder (VAE) [KW13]. However, in contrast to traditional Bayesian methods, which are good at learning interpretable patterns through latent variables, VAE's sacrifice interpretability for inference tractability.

In this chapter, we aim to integrate the merits of Bayesian models and neural networks when analyzing irregular time-evolving data. Our strategy is to modeling streaming data at two different resolution levels by proposing a dynamic change-point model. To be precise, we partition the entire time horizon into mini-batches and model the bursty dynamics between mini-batches through BNP methods. We model nonlinearity within each mini-batch through deep neural networks. In this way we are able to capture bursts and nonlinearity while also achieving interpretable results.

In Section 5.2 we introduce the modeling idea. In Section 5.3 we discuss a variational inference

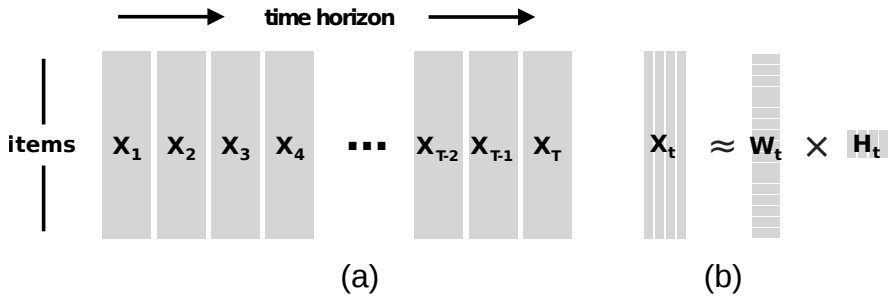


Figure 5.1: (a) Partitioning sequential data into time blocks. (b) Each block is modeled as a dynamic matrix factorization, where the left matrix uses a gamma process for change-point detection, and the right matrix is Brownian motion. Both matrices are time-evolving: the left matrix across blocks (using a jump process) and the right matrix within each block.

method. Section 5.4 and 5.4.2 introduce extensions and predictive distributions. Section 5.5 demonstrates empirical results on stock and Twitter data.

5.2 The Model

Basic linear tracking models such as the Kalman filter can be formulated as a matrix factorization problem $X \approx WH$, in which the left matrix W remains fixed and the column sequence of the right matrix H is modeled as dynamically evolving. When W is defined by the underlying physics of the problem, this is natural. But for streaming data such as stock or text data, both matrices are unknown and sequential evolution may be expected in both. In this section, we discuss our approach to the modeling problem $X \approx f(WH)$, in which W is modeled by a jump process that detects change-points in the "global" structure, while H is continuously evolving to model "local" temporal variations. We present two models: the first based on a more traditional BNP approach for which $f(\cdot)$ is the identity function, which we then extend to a neural network in a straightforward way.

5.2.1 Basic setup: Dynamic matrix factorization

In principle, our dynamic model is a continuous-time model. For practical application, we will group the data into blocks, where X_t is a $M \times N_t$ matrix of data at time block t. Each row could correspond to a stock or a word, and the columns contain the measured time-sequence at N_t points in block t (e.g., hourly measurements in week t). Our basic matrix factorization model is of the form $X_t \approx f(W_t H_t)$, where W_t is evolving in t and the columns of H_t are evolving (see Figure 5.1). In this sense, an *entire* data matrix X can be viewed as being factorized where H is one process evolving along the columns

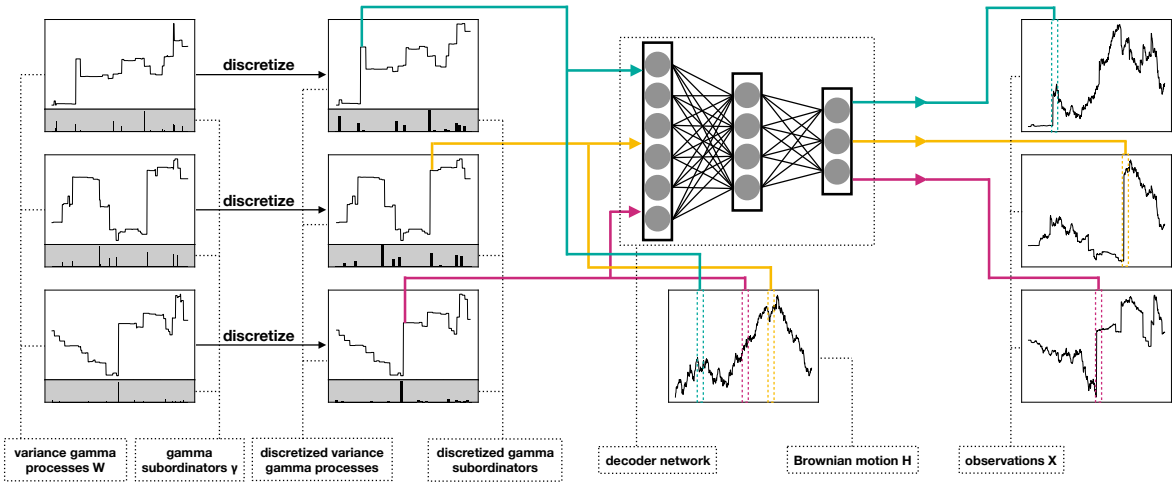


Figure 5.2: A rank-one example of the deep BNP tracking model for three data streams. A continuous-time variance gamma process is discretized into time blocks at finer resolutions than the large jumps in the process. A single Brownian motion multiplies with these variance gamma processes and is passed into a neural network, which parameterizes the mean and covariance of the observed data streams.

and H_t only selects the relevant submatrix of this process. The entire matrix W is changing depending on what column subset of X is being modeled, but also evolving according to a dynamic process in t.

In Section 5.2.2 we discuss the process for W as an infinitely divisible continuous-time jump process, then discuss the discrete time analog that we use for inference. We discuss the process for H in Section 5.2.3, which results in our proposed non-deep BNP tracking model (for which $f(A) = A$). We make a deep extension in Section 5.2.4.

5.2.2 Variance gamma process on W

In the continuous-time setting, we define the matrix W_t to be a Brownian motion subordinated to a gamma process. Let $R = (R_t)$ be a gamma process on state space $\mathbb{R}_+$ with shape rate a and scale c, and $Z = (Z_s)$ be a standard Brownian motion (in matrix form) with state space $\mathbb{R}^{M \times K}$. Then $W_t = Z_{R_t}$ is obtained by subordinating Z to R. Since both Z and R are Lévy processes, W is also a Lévy process, and thus can be represented as summation of independent increments and evaluated at discrete time points through simple marginalization. The gamma process is a pure jump process with occasional large jumps, and therefore the Brownian motion is also a jump process with little motion interrupted by occasional large jumps [Çın11]. We use this process as a change-point model for W_t, which will allow it to remain nearly fixed over a period of time, with occasional large jumps representing a shock in the dynamic system (e.g., caused by a market event) [MCC98].

Mathematically, if we partition X into time blocks t, then Lévy process theory provides a simple generative process for W_t. Let Δs_t be the time lapse between block $t-1$ and t. Then $R_t - R_{t-1} \sim \text{Gam}(a\Delta s_t, c)$. Using a new variable for this difference, the discrete time evolution of W_t can be simply represented as

$$W_t \sim \mathcal{N}(W_{t-1}, \gamma_t I), \quad \gamma_t \sim \text{Gam}(a_0, c), \tag{5.1}$$

where $a_0 = a\Delta s_t$, assuming a constant time shift. (We've also overloaded the normal distribution.) When the partition is over a small window of time compared with the dynamics of the process, a_0 will be small and therefore γ_t will likely be small and W_t will have little change. However, γ_t will occasionally be large, which will allow for a change in the system. As written, we assume one gamma process; generalization to row-specific gamma processes is straightforward and will allow for each data stream (e.g., stock) to have its own change-points. The BNP aspect is in inferring these change-points from the data. An example of this row-specific process for W is illustrated in the left column of Figure 5.2 for a rank-one factorization.

5.2.3 Temporal tracking in H and data generation

For the local tracking model of H we use discretized Brownian motion. If $H_{t,j}$ is the jth column of H_t, then we model its dynamics and resulting data generation as

$$H_{t,j} \sim \mathcal{N}(H_{t,j-1}, \lambda I), \quad X_t \sim \mathcal{N}(W_t H_t, \sigma^2 I), \tag{5.2}$$

where λ represents the time interval between data points (assumed constant), and $H_{t,1}$ uses the last point in H_{t-1}. We've again overloaded notation of the second Gaussian distribution. The columns of H_t follow a continuous process and therefore only allows "smooth" change, although the process is nowhere differentiable.

5.2.4 Extension: A deep likelihood model

In the previous sections we proposed a linear Gaussian matrix factorization model for tracking. We next extend this to a deep model in a simple way. Here we use the variational autoencoder framework to motivate the model development, and the inference algorithm discussed later [KW13, KSS15]. We use a neural network, denoted by its parameters ϕ, to define the decoder model $p_\phi(X_t|W_t, H_t)$ for

block t as a Gaussian additive noise model,

$$X_t \sim \mathcal{N}\left(\mu_\phi(W_t H_t), \Sigma_\phi(W_t H_t)\right). \tag{5.3}$$

Here we have again overloaded the Gaussian notation. In this likelihood model, we define a multivariate Gaussian on the jth column of X_t with mean and covariance a neural network that is a function of the jth column of $W_t H_t$. The benefit of this formulation can be seen in the inference step, where we are able to easily handle the stochastic gradient of ϕ [KW13]. We discuss the design of networks μ_ϕ and Σ_ϕ in Section 5.3.2. We illustrate this deep model in Figure 5.2.

5.3 Variational inference

5.3.1 Linear Gaussian observational model

We derive variation inference algorithms for the two models proposed in Section 5.2. We first discuss inference for the simpler linear Gaussian model. By our choice of q distributions, the algorithm nearly reduces to an EM algorithm in which one component is the traditional Kalman filter. To approximate the full posterior, we use the factorized q distribution of the form

$$q(\gamma, W, H) = \prod_{t=1}^{T} q(\gamma_t)\delta(W_t)q(H_t). \tag{5.4}$$

Note that within a time block, we do not further factorize $q(H_t)$. Our q on W_t is a delta function, meaning we actually learn a point estimate of this variable. As a result, the conditional posterior $p(\gamma, H|W, X) = p(H|W, X)\prod_t p(\gamma_t|W_t)$. Our only approximation is therefore in the factorization $\prod_t q(H_t)$, which only breaks dependence in the transition across time blocks. The variational lower bound is,

$$\mathcal{L} = \mathbb{E}_q\Big[\ln \frac{p(\gamma, W, H, X)}{q(\gamma, W, H)}\Big] = \mathcal{L}_H + \mathcal{L}_{W,\gamma}. \tag{5.5}$$

The H portion is $\mathcal{L}_H = \sum_{t=1}^{T} \mathcal{L}_H^{(t)}$, where

$$\mathcal{L}_H^{(t)} = \sum_{j=1}^{N_t} \mathbb{E}\Big[\ln \frac{p(X_{t,j}|W_t, H_{t,j})p(H_{t,j}|H_{t,j-1})}{q(H_{t,j}|H_{t,j-1})}\Big]. \tag{5.6}$$

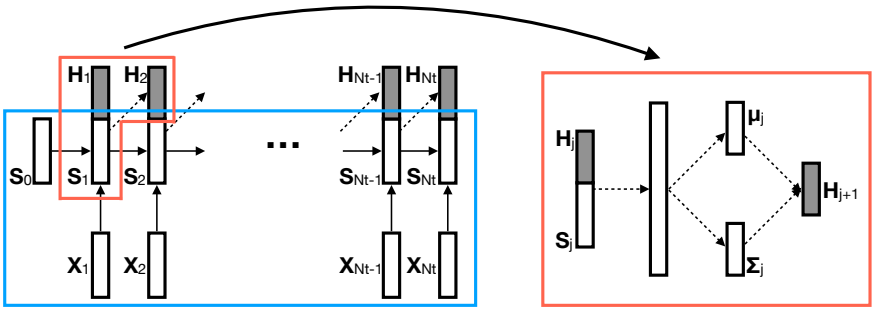


Figure 5.3: Left: Encoder network using an LSTM. Right: Re-parameterization and sequential sampling from $q_\theta(H_{t,j+1}|X_{t,1:j})$.

The W, γ portion is $\mathcal{L}_{W,\gamma} = \sum_{t=1}^{T} \mathcal{L}_{W,\gamma}^{(t)}$, where

$$\mathcal{L}_{W,\gamma}^{(t)} = \mathbb{E}\Big[\ln \frac{p(\gamma_t)p(W_t|W_{t-1},\gamma_t)}{q(\gamma_t)}\Big]. \tag{5.7}$$

We use coordinate ascent to iterative between the following closed-form updates. All expectations are with respect to q.

Update $q(\gamma_t)$: The conditional posterior of γ_t is a generalized inverse Gaussian. Thus $q(\gamma_t) = \text{GIG}(a_t, b_t, p_t)$, where $a_t = 2c, b_t = \|W_t - W_{t-1}\|_F^2, p_t = a_0 - Md/2$.

Update W_t: This is a closed-form update,

$$W_t = \left(M_1 + \frac{X_t \mathbb{E}[H_t^\top]}{\sigma^2}\right)\left(M_2 + \frac{\mathbb{E}[H_t H_t^\top]}{\sigma^2}\right)^{-1}, \tag{5.8}$$

$M_1 = \mathbb{E}[\gamma_t]W_{t-1} + \mathbb{E}[\gamma_{t+1}]W_{t+1}$, $M_2 = (\mathbb{E}[\gamma_t] + \mathbb{E}[\gamma_{t+1}])I$.

Update $q(H_t)$: This is the classic Kalman filtering problem using the current value of W_t. The standard forward-backward (filtering-smoothing) algorithm is used to solve this linear dynamic system [GH96b].

5.3.2 Extension: Variational auto-encoder model

In Section 5.2.4 we introduced a neural network based likelihood model $p_\phi(X_t|W_t, H_t)$ with nonlinear dependencies on the product $W_t H_t$. Variational inference for this model is non-trivial because of the clear non-conjugacy. We apply the variational auto-encoder by introducing a variational posterior

Algorithm 6 Sampling from $q_\theta(H_t|X_t)$

Get S_t by passing X_t to an LSTM.
(Figure 5.3 blue part)
Sample $\widehat{H}_{t,1}$ from an initial distribution.
for $j = 2, \ldots, N_t$ **do**
    Get parameters $(\mu_{t,j}, (\Sigma_{t,j}^{1/2}))$ for $H_{t,j}$ by passing $(\widehat{H}_{t,j-1}, S_{t,j-1})$ to a feed-forward network.
(Figure 5.3 red part)
    Sample $\widehat{H}_{t,j} = \mu_{t,j} + \Sigma_{t,j}^{1/2}\varepsilon, \;\; \varepsilon \sim \mathcal{N}(0, I)$.
(Figure 5.3 red part)
end for

conditioning on the context X,

$$q(\gamma, W, H|X) = \prod_{t=1}^{T} q(\gamma_t)\delta(W_t)q_\theta(H_t|X_t). \tag{5.9}$$

Note that $q_\theta(H_t|X_t)$ is a neural network encoder model for the decoder $p_\phi(X_t|W_t, H_t)$. To match the posterior structure to the prior one, we further factorize

$$q_\theta(H_t|X_t) = \prod_{j=1}^{N_t} q_\theta(H_{t,j}|H_{t,j-1}, X_t). \tag{5.10}$$

Encoder design. To exploit the sequential structure of the data, we use LSTM [HS97] to encode X_t. The pipeline of this encoder network is shown in Figure 5.3. In particular, we introduce another hidden layer S_t as the LSTM output. Then we sample $H_{t,j}$ sequentially by conditioning on $(S_{t,j-1}, H_{t,j-1})$. In this case we concatenate $(S_{t,j-1}, H_{t,j-1})$ into a single vector and pass that vector to two feed-forward neural networks to obtain the Gaussian parameters $\mu_\theta(S_{t,j-1}, H_{t,j-1}), \Sigma_\theta(S_{t,j-1}, H_{t,j-1})$ as outputs[1]. Finally we sample from the distribution given above. The entire process is summarized in Algorithm (6).

Decoder design. As previously discussed, we define

$$p_\phi(X_{t,j}|W, H) = \mathcal{N}(\mu_\phi(W_t H_{t,j}), \Sigma_\phi(W_t H_{t,j})), \tag{5.11}$$

where parameters $\mu_\phi(W_t H_{t,j}), \Sigma_\phi(W_t H_{t,j})$ are modeled by separate feed-forward neural networks. As we will see, we can exploit this structure to simplify inference using re-parameterization.

[1]In our experiments we restrict Σ_ϕ to be a diagonal matrix and use neural networks to model the logarithm of each diagonal entries. We use this trick for both the encoder network and the decoder network to refrain from doing an additional projection steps, which can be time-consuming.

Variational inference. Since the variational auto-encoder model is defined locally, we have a similar learning framework. We can again re-write $\mathcal{L} = \mathcal{L}_H + \mathcal{L}_{W,\gamma}$, where $\mathcal{L}_H = \sum_{t=1}^{T} \mathcal{L}_H^{(t)}$ and $\mathcal{L}_{W,\gamma} = \sum_{t=1}^{T} \mathcal{L}_{W,\gamma}^{(t)}$. While $\mathcal{L}_{W,\gamma}^{(t)}$ is as in Equation (5.7), the local bound is slightly different:

$$\mathcal{L}_H^{(t)} = \sum_{j=1}^{N_t} \mathbb{E}\left[\ln \frac{p_\phi(X_{t,j}|W_t, H_{t,j})p(H_{t,j}|H_{t,j-1})}{q_\theta(H_{t,j}|H_{t,j-1}, X_t)}\right]. \tag{5.12}$$

Update $q(\gamma_t)$: The same as the linear likelihood model.

Update W_t: This time we do not have a closed-form solution since we can only get samples for the hidden variables in the neural network. Instead we will use SGD, where

$$\nabla_{W_t}\mathcal{L} \approx \nabla_{W_t} \frac{1}{M} \sum_{j=1}^{N_t} \sum_{m=1}^{M} \ln p_\phi(X_{t,j}|W_t, \widehat{H}_{t,j}^{(m)}) + \tag{5.13}$$

$$\nabla_{W_t}\mathbb{E}[\ln p(W_t|W_{t-1}, \gamma_t) + \ln p(W_{t+1}|W_t, \gamma_{t+1})],$$

and $\widehat{H}_{t,j}^{(m)}$ are i.i.d. samples of $H_{t,j}$, $m = 1, \ldots, M$. We observe that the first line is approximated by Monte Carlo, while the second is in closed form.

Update $q_\theta(H_t|X_t)$: We again use SGD,

$$\nabla_\theta \mathcal{L} = \sum_{j=1}^{N_t} \nabla_\theta \mathbb{E}\left[\ln \frac{p(H_{t,j}|H_{t,j-1})}{q_\theta(H_{t,j}|H_{t,j-1}, X_t)}\right]. \tag{5.14}$$

To estimate this gradient, we observe that we are able to move the gradient inside the integral through the re-parameterization trick. By sampling H_t according to Algorithm 6, we are able to get an unbiased estimation of the gradient. In this case we found that using only one sample $\widehat{H}_t$ is enough to get an estimation of this direction with reasonable variance.

Update $p_\phi(X_t|W_t, H_t)$: We again use SGD to approximate $\nabla_\phi \mathcal{L}$ in a nearly identical way as updating $q_\theta(H_t|X_t)$.

5.4 Further discussion

We briefly discuss a straightforward motion modeling extension that can better capture latent trajectories, modeling data with missing values, and also the prediction equations we use for our experiments.

5.4.1 Modeling velocity and acceleration of drift

The Brownian motion H_t discussed above cannot project future trajectories in the latent space. We augment the model with standard tracking methods that imposes "Earth's physics" upon this space. Therefore, we augment H_t by tripling the number of rows and modeling the drift Δs into the future using the kinematic equations as follows,

$$\begin{aligned} H_{t,j}^{(pos)} &= H_{t,j-1}^{(pos)} + \Delta s \cdot H_{t,j-1}^{(vel)} + \frac{\Delta s^2}{2} \cdot H_{t,j-1}^{(acc)}, \\ H_{t,j}^{(vel)} &= H_{t,j-1}^{(vel)} + \Delta s \cdot H_{t,j-1}^{(acc)}, \\ H_{t,j}^{(acc)} &= e^{-\alpha \Delta s} \cdot H_{t,j-1}^{(acc)}. \end{aligned} \tag{5.15}$$

$\alpha > 0$ is a damping factor. In this way we are able to explicitly model the evolving position, velocity and acceleration of H according to basic physics properties. By introducing two deterministic matrices

$$G_1 = \begin{bmatrix} I & \Delta s \cdot I & \frac{1}{2}\Delta s^2 \cdot I \\ 0 & I & \Delta s \cdot I \\ 0 & 0 & e^{-\alpha \Delta s} \cdot I \end{bmatrix}, \quad G_2 = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{5.16}$$

we can rewrite the transitions and observations as

$$H_{t,j} \sim \mathcal{N}(G_1 H_{t,j-1}, \lambda I), \; X_t \sim \mathcal{N}(W_t G_2 H_t, \sigma^2 I). \tag{5.17}$$

We see that G_1 projects $H_{t,j-1}$ into the future. (The velocity and acceleration are directly learned from data.) G_2 picks out the current position to generate the observation. The above inference algorithms can be easily modified by inserting G_1 and G_2 at the appropriate places.

$$X_t \sim \mathcal{N}(A_t W_t G_2 H_t, \sigma^2 \text{diag}(A_t) I) \tag{5.18}$$

Nothing change with the remaining parts.

5.4.2 Prediction

There are two scenarios. The first one is "in-matrix" prediction, which aims at predicting missing values in X_t. The second is "sequential prediction," which predicts the next columns of X_t given previous column. For in-matrix prediction,

we have information from both the past and the future.

Given q, for model with drift, we select the position from H with G_2 and set $\widehat{X}_{t,j} = W_t G_2 \mathbb{E}_q[H_{t,j}]$.

The missing values in X_t are filled with the corresponding value in $\widehat{X}_t$. For sequential prediction, for model with drift we project into the future with G_1 and then select the position from H with G_2, and predict $X_{t,j} = W_t G_2 G_1 \mathbb{E}_q[H_{t,j-1}]$.

5.5 Experiments

5.5.1 Methods and evaluations

We empirically evaluate several approaches to this matrix factorization tracking problem. We summarize their acronyms in Table 5.1. One immediate comparison is with a model that uses Brownian motion to model both W and H— in other words the linear model of this chapter without the gamma process, but a fixed variance on W_t. We refer to this model as the collaborative Kalman filter (CKF) [GP14]. Furthermore, we can incorporate the dynamics of Section 5.4.1 on H_t, and also use the VAE approach discussed above with the CKF.

Table 5.1: Methods evaluated in our experiments.

Notation	Description
Interp	Piecewise linear interpolation
CKF	Collaborative Kalman filter
CKF-drift	CKF with velocity and acceleration
CKF-VAE	CKF-drift, nonlinear VAE version
VGP	Variance gamma process model
VGP-drift	VGP with velocity and acceleration
VGP-VAE	VGP-drift, nonlinear VAE version

Table 5.2: Linear likelihood model predictive results for stock data set.

Method	In-matrix RMSE						Sequential RMSE					
	d=5	d=10	d=15	d=20	d=30	d=50	d=5	d=10	d=15	d=20	d=30	d=50
CKF	0.7815	0.7483	0.7198	0.6960	0.6830	0.7124	0.7103	0.5456	0.5295	0.5030	0.4933	0.4874
CKF-drift	0.7618	0.7325	0.7001	0.6672	**0.6620**	0.6853	0.6853	0.5395	0.5291	0.4982	0.4915	**0.4839**
VGP	0.7777	0.7216	0.7038	0.6678	0.6342	0.6241	0.6874	0.5356	0.5205	0.5013	0.4955	0.4845
VGP-drift	0.7416	0.7210	0.6727	0.6463	0.6247	**0.6077**	0.6760	0.5321	0.5191	0.4999	0.4933	**0.4820**
Interp	0.6704						0.4983					

Table 5.3: Deep likelihood model predictive results for stock data, using various LSTM latent unit sizes.

Method	In-matrix	Sequential
CKF-VAE-50	0.6108±0.0245	0.4824±0.0122
CKF-VAE-200	0.5934±0.0205	0.4735±0.0082
VGP-VAE-50	0.5846±0.0132	0.4742±0.0217
VGP-VAE-200	**0.5364±0.0153**	**0.4628±0.0102**

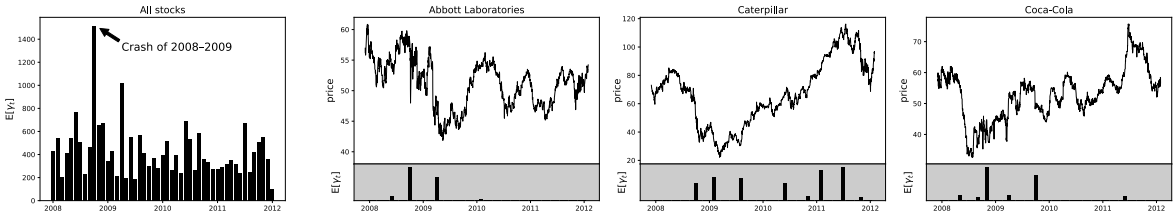


Figure 5.4: The jumps learned by our deep VGP-VAE model. (a) Summation of posterior means for all stocks $\sum_m \mathbb{E}[\gamma_{t,m}]$. This demonstrates a significant jump in the market corresponding to the crash of 2008-2009. (b) the stock price for three companies and their corresponding gamma processes. Spikes indicate large jumps of their relevant locations in the row space of W_t (not shown).

We also consider three approaches from this chapter: the variance gamma process linear model, that model adding dynamics, and a VAE approach to the model with dynamics. For our experiments, we allow each row of W to have its own associated gamma process, which represents the fact that items, such as stocks or words, do not share the same change-points. For notation simplicity we presented the algorithm for a single variance gamma process on W, but in our experiments we use row-specific variance gamma processes with straightforward modifications to account for $q(\gamma_t) = \prod_{m=1}^{M} q(\gamma_{t,m})$.

As a baseline, we compare with piece-wise linear interpolation, which predicts the next value to be the current value, or locally averages two adjacent values. For all methods, we use root-mean-square error (RMSE) as our performance metric using the predictions discussed in Section 5.4.2. For each method we run five experiments with random initialization.

5.5.2 Stock market crash and recovery, 2008-2012

We present quantitative and qualitative evaluation of our model on stock market data that measures daily stock prices at opening and closing times for $M = 1429$ companies from the AMEX exchange, NASDAQ, and NYSE, giving 2.86 million total measurements from 2008-2012.[2] In particular, we partition the entire time horizon into four-week blocks. This gives $T = 50$ segments in total. Within each block we have $N_t = 40$ measurements for each stock.

For the CKF models, we set the transition variance $\lambda = 0.1$ and the likelihood variance $\sigma^2 = 1$. For models with drift, we set the damping parameter $\alpha = 0.1$. For VGP models, we set $\gamma_{t,m} \sim_{iid} \mathrm{Gam}(1, 1)$, meaning $c = 1$ and the integral of the Lévy measure over four weeks equals 1. For the encoder part of VAE models (see Figure 5.3), we set the LSTM latent dimension to be 200. We set the output part of the encoder network (see the red box in Figure 5.3) to be one feed-forward layer followed

[2]Data source: http://ichart.finance.yahoo.com/

Table 5.4: In-matrix prediction results for encoder/decoder choices.

		Decoder		
		ff	recur	conv
Encoder	ff	0.592	0.735	>1
	recur	0.536	0.584	>1
	conv	>1	>1	>1

Table 5.5: In-matrix prediction results for various choices of decoder.

$f(W_t, H_t)$	$f(W_t \cdot H_t)$	$f(W_t, g(H_t))$	$f(W_t \cdot g(H_t))$
0.673	0.536	>1	>1

by a RELU unit [NH10a] before splitting the outputs for μ and Σ. The decoder part contains two feed-forward layers with RELU units. For optimization we use Adam [KB14] with learning rate 10^{-4}. In the experiments we find inference it takes around one hour of coordinate ascent for the linear models; the VAE models converge more slowly (approximately one day), but the result is significantly improved.

In order to quantitatively compare those models, we show the in-matrix prediction performance and sequential predictive performance, described in Section 5.4.2. For in-matrix prediction we crop $1/5$ of the data for testing and use the rest for training. And for sequential prediction we use stock prices from 2008 to 2012 for training and predict stock prices sequentially in 2013.

Linear likelihood models. We show the predictive performance in Table 5.2 as a function of the factorization rank. The VGP-based models have superior performance over the models driven only by Brownian motion. We can also see an improvement in RMSE when we explicitly model the latent trajectory in H. This indicates that the latent motion trend in the market is helpful for prediction. We note that the CKF model is even slightly worse than the linear interpolation method, indicating that the stock market is very hard to track with a linear model. For sequential prediction, the performance gap is not as large as the in-matrix prediction since observations ahead of time and correlations among stocks are informative for a precise prediction.

Deep VAE models. Next we show empirical performance for VAEs. We recall that the linear models $W_t H_t$ is fed into the encoder network in our implementation. (We will compare with other approaches later.) The result is shown in Table 5.3, where we can see that the VAE significantly improves performance of both CKF-drift and VGP-drift models. VGP-VAE is the best among all models by exploiting

both latent jumps and nonlinearity to model the data. Moreover, we observe that VAE models get better results when using a larger number of LSTM latent units S.

VAE network design. We find that the choice of encoder/decoder networks will affect the result. First, we compare three network settings for both the encoder and decoder: feed-forward (ff), recurrent (recur), and convolutional (conv). For recurrent networks we use LSTM for our experiments, and for the decoder convolutional network we use transposed convolution layers (i.e., deconvolution). We tune all networks for various layer widths and depths. Table 5.4 shows the best result we get for each encoder/decoder network pair. We find that using LSTM as encoder and feed-forward as decoder has the best result. This is because LSTM can exploit the sequential properties within data and effectively encode the observations into H. Since we also assume a sequential structure in $q_\theta(H_t|X_t)$, the decoder can be done in a straightforward way through a feed-forward network.

We also note that our decoder can be generalized as a function of W_t and H_t. In this case, we have multiple choices of the functional form for our decoder network. For example, we can first concatenate W_t and H_t and then feed them into a neural network (denoted as $f(W_t, H_t)$), or we can multiply them before feeding into the network (denoted as $f(W_t \cdot H_t)$, the method we've used thus far). The fact that H_t itself has a sequential structure means we can potentially exploit this by first feeding H into an LSTM (denoted as $g(H_t)$) before considering how it interacts with W_t. Table 5.5 shows the quantitative results for those choices. We find that feeding the linear model $W_t H_t$ directly into the neural network has the best performance. This indicates our initial BNP linear approach still has value when combined with a deep model.

Qualitative results. We also present a qualitative evaluation of the VGP-VAE model. Figure 5.4 shows the variance gamma process. In the left plot, we show the posterior mean of the subordinator gamma process $\mathbb{E}[\gamma_{t,m}]$ summed over all stocks, showing amount of "jumpiness" in the stocks learned by our model. We note that this should not be sparse, since in any time frame we expect a subset of stocks to have fundamental changes. However, our model does detect a significant global jump around October in 2008, which corresponds to the most recent stock crush.

In the three right plots, we show the jumps learned from three individual stocks: Abbott Laboratories (Medical & Health), Caterpillar (Energy, Industry), and Coca-Cola (Consumer Staples), in the lower gray subplots. As expected, their learned gamma processes are sparse. From the stock prices,

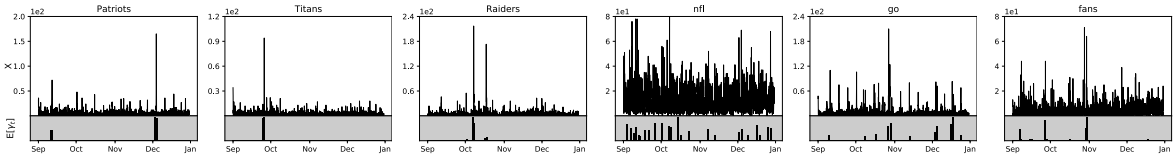


Figure 5.5: Top plots show raw counts for 6 different words. We can see the word frequency is very noisy in the entire period. Lower shaded plots show posterior mean of the gamma process $\mathbb{E}[\gamma_{t,m}]$ showing the jumps in word embedding locations learned by our model. Note that (i) the resolution of the plots are different, the top being hourly and the bottom per-day; (ii) the learned spikes show jumps of latent row vectors in W_t through a variance gamma process (not shown). So our method is essentially different from sparse recovery algorithms, such as soft-thresholding [Don95].

we can see all the three stocks experienced a similar drop during the market crash. The variance gamma process does not detect stable, gradual changes in stock price because these parts can be tracked accurately by H and the VAE. The gamma process is intended to detect change-points, where fundamental changes occur (i.e., as modeled by jumps in the latent embedding space). For example, in July 2011, Caterpillar agreed to pay a Clean Air Act penalty, which seems to have affected its location relative to other stocks in the row space of W_t.

5.5.3 NFL tweets

Another typical example where burst happens naturally is in social media. Here we aim to analyze 377,164 NFL-related tweets (e.g., labeled with an #nfl hashtag) posted during the 2011-2012 season [SDGS13]. All the data were partitioned into $T = 122$ mini-batch, one per day. And for each day, we aggregate all tweets into bag-of-words by hour. Thus $N_t = 24$. We remove all stop words and pick the top $M = 5000$ words by their document level tf-idf rank [SB88]. That is, our observation is a $5,000$ by (24×122) time-series matrix.

Quantitative results. We use the first $4/5$ of data for training and the rest for testing. We set $\lambda = 0.3$, $\sigma^2 = 1$, $\alpha = 0.1$, and the prior $\gamma_{t,m} \sim_{iid} \mathrm{Gam}(1, 1)$. For VAE models we set the LSTM latent dimension to be 40 and the learning rate for Adam to be 10^{-4}. The result is shown in Table 5.6. For CKF models we show the best result when tuning $K \in \{5, 10, 15, 20, 30, 50\}$. Again we see that VAE models have the best predictive performance due to its ability to model bursts through a Bayesian nonparametric jump process and nonlinearity through neural networks.

Qualitative results. In Figure 5.5 we show the counts of various words and their learned gamma process $\mathbb{E}[\gamma_t]$ for the VGP-VAE model. The first three plots shows results for various teams: Patriots, Titans, and Raiders. The appearance of those words is very informative and bursty, and the gamma

Table 5.6: Sequential predictive results for NFL tweets.

Method	Sequential RMSE
Interp	0.481
CKF	0.478± 0.002
CKF-drift	0.475± 0.002
CKF-VAE	**0.470±0.001**
VGP	0.477± 0.002
VGP-drift	0.471± 0.003
VGP-VAE	**0.463±0.002**

process captures the big changes that aren't able to be modeled by changes in H. Clearly we can see only a few spikes in the plot for most teams, and often there are two adjacent spikes, which indicates a very short change that doesn't persist.

The last three plots show results for four less informative words: 'nfl', 'go', and 'fans'. We can still learn some spikes in these words. However, the spikes are less informative and they are at a relatively smaller scale than the spikes learned from first three words.

Sensitivity results. Finally, we show sensitivity analysis for the VGP model in the predictive performance as a function of block window size[3]. The result is shown in Figure 5.6, where we can clearly see that when the window size is very small we are unable to get good predictive results. In this case, the H has difficulty accurately capturing the basic dynamics because W_t is updated so frequently and many local optima exist. On the other hand, when we use very large batch size, we are still have decent predictive results. However, in this case our variance gamma process model is very coarse, so that we do not have interpretable results of change-points.

5.6 Conclusion

In this chapter, we introduce a new method to integrate Bayesian nonparametrics and deep neural networks for time-series data. We treat a collection of data sequences, such as stock prices, as a matrix factorization problem in which there are temporal dynamics in both matrices of the factorization. For the left matrix, we use a variance gamma (jump) process to model change-points in what should otherwise be a fixed latent embedding. Columns of the right matrix follow a Brownian motion. We then extend this linear Gaussian basic model to a neural network and use the variational auto-encoder for approximate posterior inference. Our method benefits from both BNP methods (jump processes

[3]We observe a similar result in the VGP-VAE model.

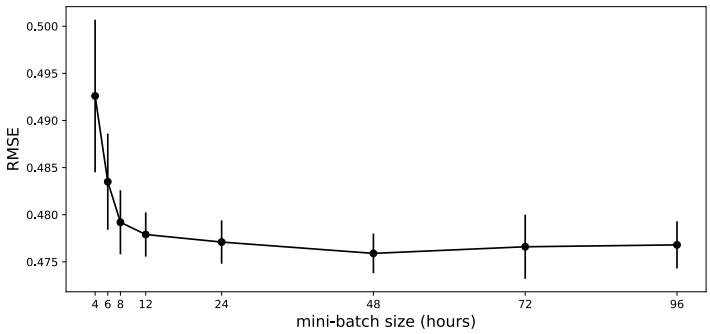


Figure 5.6: Sensitivity result on nfl tweet data set with various time windows. To get both good predictive and interepretable results, the best choice is to choose an intermediate discretization.

for change-point detection and explainable latent representations) and deep neural networks (non-linearity for generalized linear modeling). We empirically evaluated our method on two data sets, demonstrating predictive power and interpretability.

Chapter 6

Fully Supervised Speaker Diarization

In this chapter, we propose a fully supervised speaker diarization approach, named *unbounded interleaved-state recurrent neural networks* (UIS-RNN). Given extracted speaker-discriminative embeddings (*a.k.a.* d-vectors) from input utterances, each individual speaker is modeled by a parameter-sharing RNN, while the RNN states for different speakers interleave in the time domain. This RNN is naturally integrated with a distance-dependent Chinese restaurant process (ddCRP) to accommodate an unknown number of speakers. Our system is fully supervised and is able to learn from examples where time-stamped speaker labels are annotated. We achieved a 7.6% diarization error rate on NIST SRE 2000 CALLHOME, which is better than the state-of-the-art method using spectral clustering. Moreover, our method decodes in an online fashion while most state-of-the-art systems rely on offline clustering.

6.1 Motivations

Aiming to solve the problem of "who spoke when", most existing speaker diarization systems consist of multiple relatively independent components [SGR15, GRSS+17, WDW+18], including but not limited to: (1) A speech segmentation module, which removes the non-speech parts, and divides the input utterance into small segments; (2) An embedding extraction module, where speaker-discriminative embeddings such as speaker factors [CCD+08], i-vectors [DKD+11], or d-vectors [WWPM18] are extracted from the small segments; (3) A clustering module, which determines the number of speakers, and assigns speaker identities to each segment; (4) A resegmentation module, which further refines

the diarization results by enforcing additional constraints [SGR15].

For the embedding extraction module, recent work [GRSS+17, WDW+18, ZHM17] has shown that the diarization performance can be significantly improved by replacing i-vectors [DKD+11] with neural network embeddings, *a.k.a.* d-vectors [WWPM18, HMBS16]. This is largely due to the fact that neural networks can be trained with big datasets, such that the model is sufficiently robust against varying speaker accents and acoustic conditions in different use scenarios.

However, there is still one component that is unsupervised in most modern speaker diarization systems — the clustering module. Examples of clustering algorithms that have been used in diarization systems include Gaussian mixture models [ZHM17, SDDG13], mean shift [SKSD14], agglomerative hierarchical clustering [GRSS+17, SGR14], k-means [WDW+18, DF17], Links [WDW+18, MWD+18], and spectral clustering [WDW+18, NLTH06].

Since both the number of speakers and the segment-wise speaker labels are determined by the clustering module, the quality of the clustering algorithm is critically important to the final diarization performance. However, the fact that most clustering algorithms are unsupervised means that, we will not able to improve this module by learning from examples when the time-stamped speaker labels ground truth are available. In fact, in many domain-specific applications, it is relatively easy to obtain such high quality annotated data.

In this chapter, we replace the unsupervised clustering module by an online generative process that naturally incorporates labelled data for training. We call this method *unbounded interleaved-state recurrent neural network* (UIS-RNN), based on these facts: (1) Each speaker is modeled by an instance of RNN, and these instances share the same parameters; (2) An unbounded number of RNN instances can be generated; (3) The states of different RNN instances, corresponding to different speakers, are interleaved in the time domain. Within a fully supervised framework, our method in addition handles complexities in speaker diarization: it automatically learns the number of speakers within each utterance via a Bayesian non-parametric process, and it carries information through time via the RNN.

The contributions of our work are summarized as follows:

1. Unbounded interleaved-state RNN, a trainable model for the general problem of segmenting and clustering temporal data by learning from examples.

2. Framework for a fully supervised speaker diarization system.

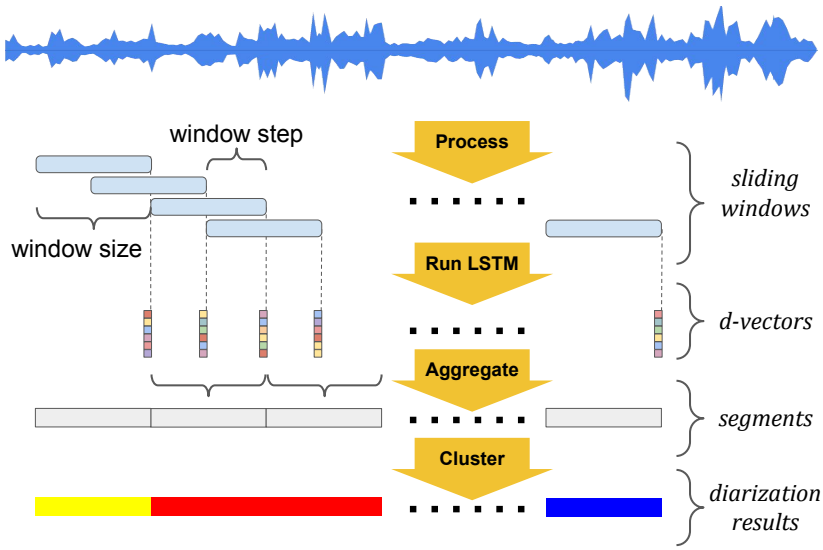


Figure 6.1: The baseline system architecture [WDW+18].

3. New state-of-the-art performance on NIST SRE 2000 CALLHOME benchmark.

4. Online diarization solution with offline quality.

6.2 Baseline system using clustering

Our diarization system is built on top of the recent work by Wang *et al.* [WDW+18]. Specifically, we use exactly the same segmentation module and embedding extraction module as their system, while replacing their clustering module by an unbounded interleaved-state RNN.

As a brief review, in the baseline system [WDW+18], a text-independent speaker recognition network is used to extract embeddings from sliding windows of size 240ms and 50% overlap. A simple voice activity detector (VAD) with only two full-covariance Gaussians is used to remove non-speech parts, and partition the utterance into non-overlapping segments with max length of 400ms. Then we average window-level embeddings to segment-level d-vectors, and feed them into the clustering algorithm to produce final diarization results. The workflow of this baseline system is shown in Fig. 6.1.

The text-independent speaker recognition network for computing embeddings has three LSTM layers and one linear layer. The network is trained with the state-of-the-art generalized end-to-end loss [WWPM18]. We have been retraining this model for better performance, which will be later discussed in Section 6.4.1.

6.3 Unbounded interleaved-state RNN

6.3.1 Overview of approach

Given an utterance, from the embedding extraction module, we get an *observation sequence* of embeddings $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, where each $\mathbf{x}_t \in \mathbb{R}^d$. Each entry in this sequence is a real-valued d-vector corresponding to a segment in the original utterance. In the supervised speaker diarization scenario, we also have the ground truth speaker labels for each segment $\mathbf{Y} = (y_1, y_2, \ldots, y_T)$. Without loss of generality, let $\mathbf{Y}$ be a sequence of positive integers by the order of appearance.

For example, $\mathbf{Y} = (1, 1, 2, 3, 2, 2)$ means this utterance has six segments, from three different speakers, where $y_t = k$ means segment t belongs to speaker k.

UIS-RNN is an online generative process of an entire utterance $(\mathbf{X}, \mathbf{Y})$, where[1]

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{x}_1, y_1) \cdot \prod_{t=2}^{T} p(\mathbf{x}_t, y_t | \mathbf{x}_{[t-1]}, y_{[t-1]}). \tag{6.1}$$

To model speaker changes, we use an augmented representation

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = p(\mathbf{x}_1, y_1) \cdot \prod_{t=2}^{T} p(\mathbf{x}_t, y_t, z_t | \mathbf{x}_{[t-1]}, y_{[t-1]}, z_{[t-1]}), \tag{6.2}$$

where $\mathbf{Z} = (z_2, \ldots, z_T)$, and $z_t = \mathbb{1}(y_t \neq y_{t-1}) \in \{0, 1\}$ is a binary indicator for speaker changes. For example, if $\mathbf{Y} = (1, 1, 2, 3, 2, 2)$, then $\mathbf{Z} = (0, 1, 1, 1, 0)$. Note that $\mathbf{Z}$ is uniquely determined by $\mathbf{Y}$, but $\mathbf{Y}$ cannot be uniquely determined by a given $\mathbf{Z}$, since we don't know which speaker we are changing to. Here we leave z_1 undefined, and factorize each product term in Eq. (6.2) as three parts that separately model sequence generation, speaker assignment, and speaker change:

$$p(\mathbf{x}_t, y_t, z_t | \mathbf{x}_{[t-1]}, y_{[t-1]}, z_{[t-1]}) = \underbrace{p(\mathbf{x}_t | \mathbf{x}_{[t-1]}, y_{[t]})}_{\text{sequence generation}} \cdot \underbrace{p(y_t | z_t, y_{[t-1]})}_{\text{speaker assignment}} \cdot \underbrace{p(z_t | z_{[t-1]})}_{\text{speaker change}}. \tag{6.3}$$

For the first entry of the sequence, we let $y_1 = 1$ and there is no need to model speaker assignment and speaker change. In Section 6.3.2, we introduce these components separately.

[1] We denote an ordered set $(1, 2, \ldots, t)$ as $[t]$.

6.3.2 Details on model components

6.3.2.1 Speaker change

We assume the probability of $z_t \in \{0, 1\}$ follows:

$$p(z_t = 0 | z_{[t-1]}, \boldsymbol{\lambda}) = g_{\boldsymbol{\lambda}}(z_{[t-1]}), \tag{6.4}$$

where $g_{\boldsymbol{\lambda}}(\cdot)$ is a function paramaterized by $\boldsymbol{\lambda}$. Since z_t indicates speaker change at time t, we have

$$p(y_t = y_{t-1} | z_t, y_{[t-1]}) \ = \ 1 - z_t. \tag{6.5}$$

In general, $g_{\boldsymbol{\lambda}}(\cdot)$ could be any function, such as an RNN. But for simplicy, in this work, we make it a constant value $g_{\boldsymbol{\lambda}}(z_{[t-1]}) = p_0 \in [0, 1]$. This means $\{z_t\}_{t \in [2,T]}$ are independent binary variables parameterized by $\boldsymbol{\lambda} = \{p_0\}$:

$$z_t \sim_{iid.} \text{Binary}(p_0). \tag{6.6}$$

6.3.2.2 Speaker assignment process

One of the biggest challenges in speaker diarization is to determine the total number of speakers for each utterance. To model the speaker turn behavior in an utterance, we use a distance dependent Chinese restaurant process (ddCRP) [BF11], a Bayesian non-parametric model that can potentially model an unbounded number of speakers. Specifically, when $z_t = 0$, the speaker remains unchanged. When $z_t = 1$, we let

$$\begin{aligned} p(y_t = k | z_t = 1, y_{[t-1]}) \ &\propto \ N_{k,t-1}, \\ p(y_t = K_{t-1} + 1 | z_t = 1, y_{[t-1]}) \ &\propto \ \alpha. \end{aligned} \tag{6.7}$$

Here $K_{t-1} = \max y_{[t-1]}$ is the total number of unique speakers up to the $(t-1)$-th entry. Since $z_t = 1$ indicates a speaker change, we have $k \in [K_{t-1}] \setminus \{y_{t-1}\}$. In addition, we let $N_{k,t-1}$ be the number of *blocks* for speaker k in $y_{[t-1]}$. A *block* is defined as a maximum-length subsequence of *continuous* segments that belongs to a single speaker. For example, if $y_{[6]} = (1, 1, 2, 3, 2, 2)$, then there are four blocks $(1, 1)|(2)|(3)|(2, 2)$ separated by the vertical bar, with $N_{1,5} = 1, N_{2,5} = 2, N_{3,5} = 1$. The probability of switching back to a previously appeared speaker is proportional to the number of continuous speeches she/he has spoken. There is also a chance to switch to a new speaker, with a

probability proportional to a constant α. The joint distribution of $\mathbf{Y}$ given $\mathbf{Z}$ is

$$p(\mathbf{Y}|\mathbf{Z},\alpha)=\frac{\alpha^{K_T-1}\prod_{k=1}^{K_T}\Gamma(N_{k,T})}{\prod_{t=2}^{T}(\sum_{k\in[K_{t-1}]\setminus\{y_{t-1}\}}N_{k,t-1}+\alpha)^{\mathbb{1}(z_t=1)}}. \tag{6.8}$$

6.3.2.3 Sequence generation

Our basic assumption is that, the observation sequence of speaker embeddings $\mathbf{X}$ is generated by distributions that are parameterized by the output of an RNN. This RNN has multiple instantiations, corresponding to different speakers, and they share the same set of RNN parameters $\boldsymbol{\theta}$. In our work, we use gated recurrent unit (GRU) [CVMG+14] as our RNN model, to memorize long-term dependencies.

At time t, we define $\mathbf{h}_t$ as the state of the GRU corresponding to speaker y_t, and

$$\mathbf{m}_t=f(\mathbf{h}_t|\boldsymbol{\theta}) \tag{6.9}$$

as the output of the entire network, which may contain other layers. Let $t'=\max\{0,\ s<t:\ y_s=y_t\}$ be the last time we saw speaker y_t before t, then:

$$\mathbf{h}_t=\text{GRU}(\mathbf{x}_{t'},\mathbf{h}_{t'}|\boldsymbol{\theta}), \tag{6.10}$$

where we can assume $\mathbf{x}_0=\mathbf{0}$ and $\mathbf{h}_0=\mathbf{0}$, meaning all GRU instances are initialized with the same zero state.

Based on the GRU outputs, we assume the speaker embeddings are modeled by:

$$\mathbf{x}_t|\mathbf{x}_{[t-1]},y_{[t]}\sim\mathcal{N}(\boldsymbol{\mu}_t,\sigma^2\mathbf{I}), \tag{6.11}$$

where $\boldsymbol{\mu}_t=(\sum_{s=1}^{t}\mathbb{1}(y_s=y_t))^{-1}\cdot(\sum_{s=1}^{t}\mathbb{1}(y_s=y_t)\mathbf{m}_s)$ is the averaged GRU output for speaker y_t.

6.3.2.4 Summary of the model

We briefly summarize UIS-RNN in Fig. 6.2, where $\mathbf{Z}$ and $\boldsymbol{\lambda}$ are omitted for a simple demonstration. At the current stage (shown in solid lines) $y_{[6]}=(1,1,2,3,2,2)$. There are four options for y_7: $1,2,3$ (existing speakers), and 4 (a new speaker). The probability for generating a new observation $\mathbf{x}_7$ (shown in dashed lines) depends both on previous label assignment sequence $y_{[6]}$, and previous observation sequence $\mathbf{x}_{[6]}$.

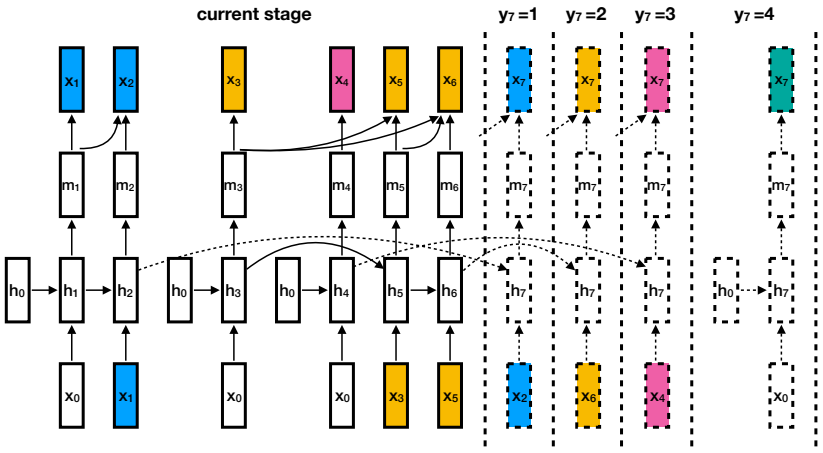


Figure 6.2: Generative process of UIS-RNN. Colors indicate labels for speaker segments. There are four options for y_7 given $\mathbf{x}_{[6]}, y_{[6]}$.

6.3.3 MLE Estimation

Given a training set $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N)$ containing N utterances together with their labels $(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N)$, we maximize the following log joint likelihood:

$$\max_{\boldsymbol{\theta}, \alpha, \sigma^2, \boldsymbol{\lambda}} \quad \sum_{n=1}^{N} \ln p(\mathbf{X}_n, \mathbf{Y}_n, \mathbf{Z}_n | \, \boldsymbol{\theta}, \alpha, \sigma^2, \boldsymbol{\lambda}). \tag{6.12}$$

Here we include all hyper-parameters, and each term in Eq. (6.12) can be factorized exactly as Eq. (6.2).

The estimation of $\boldsymbol{\lambda}$ depends on how $g_{\boldsymbol{\lambda}}(\cdot)$ is defined. When we simply have $g_{\boldsymbol{\lambda}}(z_{[t-1]}) = p_0$, we have a closed-form solution:

$$p_0^* = \frac{\sum_{n=1}^{N} \sum_{t=2}^{T_n} \mathbb{1}(y_{n,t} = y_{n,t-1})}{\sum_{n=1}^{N} T_n - N}, \tag{6.13}$$

where T_n denotes the sequence length of the nth utterance.

For $\boldsymbol{\theta}$ and σ^2, there is no closed-form update. We use stochastic gradient ascent by randomly selecting a subset $B^{(\tau)} \subset [N]$ of $|B^{(\tau)}| = b$ utterances. For $\boldsymbol{\theta}$, we update:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} + \frac{N \rho^{(\tau)}}{b} \sum_{n \in B^{(\tau)}} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{X}_n | \, \mathbf{Y}_n, \mathbf{Z}_n, \boldsymbol{\theta}, -), \tag{6.14}$$

since $\boldsymbol{\theta}$ is independent of $(\mathbf{Y}_n, \mathbf{Z}_n)$. Eq. (6.15) also applies to σ^2 by replacing $\boldsymbol{\theta}$ with σ^2. For α, we

Algorithm 7 Online greedy MAP decoding for UIS-RNN.

Data: $\mathbf{X}^{test} = (\mathbf{x}_1^{test}, \mathbf{x}_2^{test}, \ldots, \mathbf{x}_T^{test})$
Result: $\mathbf{Y}^* = (y_1^*, y_2^*, \ldots, y_T^*)$
initialize $\mathbf{x}_0 = \mathbf{0}, \mathbf{h_0} = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
$(y_t^*, z_t^*) = \arg\max_{(y_t, z_t)} \big(\ln p(z_t)$ Eq. (6.6)
$+ \ln p(y_t | z_t, y_{[t-1]}^*)$ Eq. (6.5, 6.7)
$+ \ln p(\mathbf{x}_t | \mathbf{x}_{[t-1]}, y_{[t-1]}^*, y_t) \big)$ Eq. (6.11)
update $N_{k,t-1}$ and GRU hidden states
end for

update

$$\alpha^{(\tau)} = \alpha^{(\tau-1)} + \frac{N\rho^{(\tau)}}{b} \sum_{n \in B^{(\tau)}} \nabla_\alpha \ln p(\mathbf{Y}_n | \ \mathbf{Z}_n, \alpha, -), \tag{6.15}$$

where $p(\mathbf{Y}_n | \ \mathbf{Z}_n, \alpha, -)$ is given in Eq. (6.8). In our experiments, we run multiple iterations with a constant step size $\rho^{(\tau)} = \rho$ until convergence.

6.3.4 MAP Decoding

Since we can decode each testing utterance in parallel, here we assume we are given a testing utterance $\mathbf{X}^{test} = (\mathbf{x}_1, \mathbf{x}_2 \ \ldots, \mathbf{x}_T)$ without labels. The ideal goal is to find

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} \ \ln p(\mathbf{X}^{test}, \mathbf{Y}). \tag{6.16}$$

However, this requires an exhaustive search over the entire combinatorial label space with complexity $\mathcal{O}(T!)$, which is impractical. Instead, we use an online decoding approach which sequentially performs a greedy search, as shown in Alg. 7. This will significantly reduce computational complexity to $\mathcal{O}(T^2)$. We observe that in most cases the maximum number of speakers per-utterance is bounded by a constant C. In that case, the complexity will further reduce to $\mathcal{O}(T)$. In practice, we apply a beam search [MCF+77] on the decoding algorithm, and adjust the number of look-ahead entries to achieve better decoding results.

Table 6.1: Speaker verification EER of the three speaker recognition models. en-ALL represents all English locales. The EER=3.55% for d-vector V1 on en-US phone data is the same as the number reported in Table 3 of [WWPM18].

Model	EER (%) on en-US phone data	EER (%) on en-ALL phone + farfield data
d-vector V1	3.55	6.14
d-vector V2	3.06	2.03
d-vector V3	3.03	1.91

6.4 Experiments

6.4.1 Speaker recognition model

We have been retraining the speaker recognition network with more data and minor tricks (see next few paragraphs) to improve its performance. Let's call the text-independent speaker recognition model in [WDW+18, WWPM18, JZW+18] as "d-vector V1". This model is trained with 36M utterances from 18K US English speakers, which are all mobile phone data based on anonymized voice query logs.

To train a new version of the model, which we call "d-vector V2" [WMW+18], we added: (1) non-US English speakers; (2) data from far-field devices; (3) public datasets including LibriSpeech [PCPK15], VoxCeleb [NCZ17], and VoxCeleb2 [CNZ18]. The non-public part contains 34M utterances from 138K speakers, while the public part is added to the training process using the MultiReader approach [WWPM18].

Another minor but important trick is that, the speaker recognizer model used in [WDW+18] and [WWPM18] are trained on windows of size 1600ms, which causes performance degradation when we run inference on smaller windows. For example, in the diarization system, the window size is only 240ms. Thus we have retrained a new model "d-vector V3" by using variable-length windows, where the window size is drawn from a uniform distribution within $[240\text{ms}, 1600\text{ms}]$ during training.

The speaker verification Equal Error Rate (EER) of the three models on two testing sets are shown in Table 6.1. On speaker verification tasks, adding more training data has significantly improved the performance, while using variable-length windows for training also slightly further improved EER.

6.4.2 UIS-RNN setup

For the speaker change, as we have stated in Section 6.3.2.1, we assume $\{z_t\}_{t\in[2,T]}$ follow independent identical binary distributions for simplicity.

Our sequence generation model is composed of one layer of 512 GRU cells with a tanh activation, followed by two fully-connected layers each with 512 nodes and a ReLU [NH10b] activation. The two

fully-connected layers corresponds to Eq. (6.9).

For decoding, we use beam search of width 10.

6.4.3 Evaluation protocols

Our evaluation setup is exactly the same as [WDW+18], which is based on the pyannote.metrics library [Bre17]. We follow these common conventions of other works:

- We evaluate on single channel audio.
- We exclude overlapped speech from evaluation.
- We tolerate errors less than 250ms in segment boundaries.
- We report the confusion error, which is usually directly referred to as Diarization Error Rate (DER) in the literature.

6.4.4 Datasets

For the evaluation, we use 2000 NIST Speaker Recognition Evaluation (LDC2001S97), Disk-8, which is usually directly referred to as "CALLHOME" in literature. It contains 500 utterances distributed across six languages: Arabic, English, German, Japanese, Mandarin, and Spanish. Each utterance contains 2 to 7 speakers.

Since our approach is supervised, we perform a 5-fold cross validation on this dataset. We randomly partition the dataset into five subsets, and each time leave one subset for evaluation, and train UIS-RNN on the other four subsets. Then we combine the evaluation on five subsets and report the averaged DER.

Besides, we also tried to use two off-domain datasets for training UIS-RNN: (1) 2000 NIST Speaker Recognition Evaluation, Disk-6, which is often referred to as "Switchboard"; (2) ICSI Meeting Corpus [JBE+03]. We first tried to train UIS-RNN purely on off-domain datasets, and evaluate on CALLHOME; we then tried to add the off-domain datasets to the training partition of each of the 5-fold.

6.4.5 Results

We report the diarization performance results on 2000 NIST SRE Disk-8 in Table 6.2. For each version of the speaker recognition model, we compare UIS-RNN with two baseline approaches: k-means and

Table 6.2: DER on NIST SRE 2000 CALLHOME, with comparison to other systems in literature. VB is short for Variational Bayesian resegmentation [SGR15]. The DER=12.0% for d-vector V1 and spectral clustering is the same as the number reported in Table 2 of [WDW+18].

d-vector	Method	Training data	DER (%)
V1	k-means	—	17.4
	spectral	—	12.0
	UIS-RNN	5-fold	**11.7**
	UIS-RNN	Disk-6 + ICSI	**11.7**
	UIS-RNN	5-fold + Disk-6 + ICSI	**10.6**
V2	k-means	—	19.1
	spectral	—	11.6
	UIS-RNN	5-fold	**10.9**
	UIS-RNN	Disk-6 + ICSI	**10.8**
	UIS-RNN	5-fold + Disk-6 + ICSI	**9.6**
V3	k-means	—	12.3
	spectral	—	8.8
	UIS-RNN	5-fold	**8.5**
	UIS-RNN	Disk-6 + ICSI	**8.2**
	UIS-RNN	5-fold + Disk-6 + ICSI	**7.6**
Castaldo *et al.* [CCD+08]			13.7
Shum *et al.* [SDDG13]			14.5
Senoussaoui *et al.* [SKSD14]			12.1
Sell *et al.* [SGR15] (+VB)			13.7 (11.5)
Garcia-Romero *et al.* [GRSS+17] (+VB)			12.8 (9.9)

spectral offline clustering. For k-means and spectral clustering, the number of speakers is adaptively determined as in [WDW+18]. For UIS-RNN, we show results for three types of evaluation settings: (1) in-domain training (5-fold); (2) off-domain training (Disk-6 + ICSI); and (3) in-domain plus off-domain training.

From the table, we see that the biggest improvement in DER actually comes from upgrading the speaker recognition model from V2 to V3. This is because in V3, we have the window size consistent between training time and diarization inference time, which was a big issue in V1 and V2.

UIS-RNN performs noticeably better than spectral offline clustering, when using the same speaker recognition model. It is also important to note that UIS-RNN inference produces speaker labels in an **online** fashion. As discussed in [WDW+18], online unsupervised clustering algorithms usually perform significantly worse than offline clustering algorithms such as spectral clustering.

Also, adding more data to train UIS-RNN also improved DER, which is consistent with our expectation – UIS-RNN benefits from learning from more examples. Specifically, while large scale off-domain training already produces great results in practice (Disk-6 + ICSI), the availability of in-domain data can further improve the performance (5-fold + Disk-6 + ICSI).

6.5 Conclusions

In this chapter, we presented a speaker diarization system where the commonly used clustering module is replaced by a trainable unbounded interleaved-state RNN. Since all components of this system can be learned in a supervised manner, it is preferred over unsupervised systems in scenarios where training data with high quality time-stamped speaker labels are available. On the NIST SRE 2000 CALLHOME benchmark, using exactly the same speaker embeddings, this new approach, which is an online algorithm, outperforms the state-of-the-art spectral offline clustering algorithm.

Besides, the proposed UIS-RNN is a generic solution to the sequential clustering problem, with other potential applications such as face clustering in videos. One interesting future work direction is to directly use accoustic features instead of pre-trained embeddings as the observation sequence for UIS-RNN, such that the entire speaker diarization system becomes an end-to-end model.

Part IV

Inference for Bayesian Nonparametric Models

Chapter 7

Stochastic Variational Inference for the HDP-HMM

In this chapter, we derive a variational inference algorithm for the HDP-HMM based on the two-level stick breaking construction. This construction has previously been applied to the hierarchical Dirichlet processes (HDP) for mixed membership models, allowing for efficient handling of the coupled weight parameters. However, the same algorithm is not directly applicable to HDP-based infinite hidden Markov models (HDP-HMM) because of extra sequential dependencies in the Markov chain. In this chapter we provide a solution to this problem by deriving a variational inference algorithm for the HDP-HMM, as well as its stochastic extension, for which all parameter updates are in closed form. We apply our algorithm to sequential text analysis and audio signal analysis, comparing our results with the beam-sampled iHMM, the parametric HMM, and other variational inference approximations.

7.1 Motivations

The hierarchical Dirichlet process (HDP) [TJBB06b] is a Bayesian nonparametric prior for generating multiple random measures on the same countably infinite collection of atoms. This property of the HDP makes it a natural tool for modeling groups of data that share hidden components with different mixing proportions. The most well-known application of the HDP is for handling exchangeable data through mixed membership modeling [ABEF14], as well as nonexchangeable data with hidden

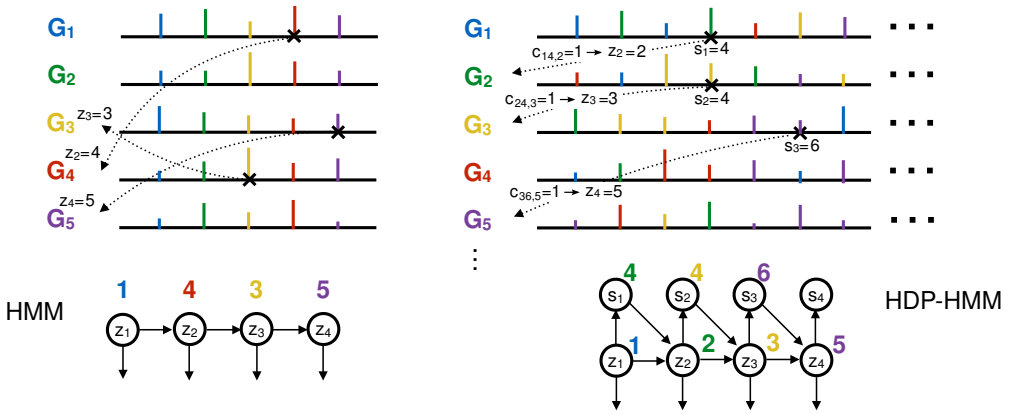


Figure 7.1: Comparison between the transition structure of the HMM and the stick-breaking construction of HDP-HMM. Each G_i represents a transition distribution from state i, and the color of a stick corresponds to one state. Left: In the HMM the state and the column index are one-to-one. Right: For the stick-breaking construction of the HDP-HMM, multiple sticks in each row may point to the same state (same color) and there is no one-to-one mapping between column and row. For example, a transition from state 3 to 5 takes place if $s = 5$ or $s = 6$. By holding the sequence $\mathbf{s}_d$ fixed and changing a DP indicator c_{km} for a selected stick (i.e., changing the color of a stick), the state transitions for *all* subsequent states may change. This presents an inference challenge for the HDP-HMM not faced by [WPB11a].

Markov models (HDP-HMM) [FSJW08, TJBB06b].

The hierarchical structure of the HDP makes inference a significant problem. For example, various sampling strategies have been developed for these models to improve efficiency: For the mixed membership model, a Chinese restaurant franchise [TJBB06b] sampling method was proposed, while for the HDP-HMM a beam sampling strategy was introduced to enable forward-backward sampling via slice sampling [Nea03] over a dynamic, truncated stick-breaking construction [VGSTG08].

Variational inference provides another promising strategy for inference in Bayesian hierarchical models by restricting the posterior to a simpler form that is able to be deterministically optimized [JGJS99]. Moreover, stochastic variational inference (SVI) allows for efficient inference over large datasets, and has been applied successfully on locally exchangeable data [HBB10, HBWP13a, WPB11a] and nonexchangeable data [FXLF14, JW14a]. For the HDP mixed membership model, batch and stochastic variational inference algorithms have been derived using a two-level stick breaking construction [WPB11a]. However, these algorithms are not immediately transferable to the HDP-HMM because of sequential dependencies in the local variables. Alternative, fully conjugate nonparametric priors for the HMM have also been proposed [PC09a].

Previous work has focused on SVI for hidden Markov models [FXLF14], but is not directly applicable to the HDP-HMM. A recent SVI approach to the HDP-HMM is based on a point estimate strategy

to avoid non-conjugacy [LPJK07b, JW14a]. In this chapter we address posterior inference for the HDP-HMM over all variables by deriving batch and stochastic variational algorithms using the fully conjugate representation of [WPB11a] rather than the representation by [JW14a], with which we compare.

In Section 7.2, we present the construction of the HDP-HMM we use for inference, and derive batch and stochastic variational inference algorithms in Section 7.3. We then apply our model to both artificial data and real data in Section 7.4, including a sequential text dataset and a large-scale audio dataset. Empirical results demonstrate the effectiveness of our method when compared with the beam-sampled iHMM [VGSTG08], the HDP-HMM with simpler direct assignment variational approximations [LPJK07b, JW14a] and its split-merge variation [BS12], as well as the parametric batch and stochastic HMM [Bea03, FXLF14].

7.2 The HDP-HMM

The HDP-HMM uses the hierarchical Dirichlet process for Bayesian nonparametric inference over the number of states in a hidden Markov model. To review, let G be a top-level Dirichlet process, written $G \sim \mathrm{DP}(a_0\mu)$, with $a_0 > 0$ and μ a non-atomic probability measure. Since G is a.s. discrete, we can write $G = \sum_k \eta_{0k}\delta_{\theta_k}$. The HDP uses G as the base distribution of a possibly infinite number of second-level Dirichlet processes, written $G_k \sim_{iid} \mathrm{DP}(\tau_0 G)$.

In the context of the HDP-HMM, G_k is the transition distribution for state k. To generate a sequence of data, one generates a sequence of parameters $(\theta'_1, \ldots, \theta'_n)$ by first sampling θ'_1 from an initial-state distribution G_0 followed by the rest of the sequence. To this end, we introduce the state index z_i, which equals k if $\theta'_i = \theta_k$. The next parameter in the Markov chain is then generated from the DP indexed by z_i, $\theta'_{i+1} \sim G_{z_i}$. The observed sequence $(x_1, \ldots, x_n)$ is generated using these parameters, where $x_i \sim p(x|\theta'_i)$. In this chapter, we will focus on the discrete HMM.

7.2.1 Stick-breaking construction

Our inference method is based on the stick-breaking construction for the HDP [Set94a], which we briefly review. To generate the top-level DP, we let

$$G = \sum_{k=1}^{\infty} \underbrace{\zeta_k \prod_{j=1}^{k-1}(1-\zeta_j)}_{\equiv\, \boldsymbol{\eta_{0k}}} \delta_{\theta_k},$$

$$\zeta_k \overset{iid}{\sim} \text{Beta}(1, a_0), \quad \theta_k \overset{iid}{\sim} \text{Dir}(b_0 \mathbf{1}). \tag{7.1}$$

The infinite number of second-level DP's are then drawn

$$G_k = \sum_{m=1}^{\infty} \underbrace{\varepsilon_{km} \prod_{j=1}^{m-1} (1 - \varepsilon_{kj})}_{\equiv \boldsymbol{\eta_{km}}} \delta_{\phi_{km}},$$
$$\varepsilon_{km} \overset{iid}{\sim} \text{Beta}(1, \tau_0), \quad \phi_{km} \overset{iid}{\sim} G. \tag{7.2}$$

Since G is discrete almost surely, there is a mapping from ϕ_{km} to θ_i, and many ϕ_{km} will map to the same θ_i. This introduces additional complexity during inference that makes learning parameters more complicated than for the parametric HMM [Bea03].

For inference we introduce the indicator vector c_{km}, which indexes the top-level atom picked for ϕ_{km}. Therefore $c_{km,k'} = 1$ if $\phi_{km} = \theta_{k'}$ and $c_{km} \sim \text{Mult}(\eta_0)$.[1] It turns out that for the HDP mixed membership model these indicator variables are especially important for closed-form updates in variational inference [WPB11a]. [WPB11a] draw from G_k by first drawing a stick indicator s and then mapping to the top-level atom associated with the chosen stick as indicated by c_{ks}.

We will use this auxiliary variable for the HDP-HMM as well. First, for the dth observed sequence, we sample $s_{d,i} | \{z_{d,i-1} = k\} \sim \text{Disc}(\eta_k)$ and then set $z_{d,i} = k'$ if $c_{k s_{d,i}, k'} = 1$ to index the next state (see Figure 7.1). This two-step process of first selecting the stick and then mapping to the top level atom works easily for the HDP in the mixed membership setting, but this algorithm is not directly applicable to the HDP-HMM because of the non-exchangeability of the sequence. That is, given a sequence of stick indicators $\mathbf{s}_d$, if we change the value of c_{km} we may affect the path of the entire Markov chain $\mathbf{z}_d$. This problem of indicators pointing to indicators constitutes the challenge of variational inference for the HDP-HMM, which we illustrate in Figure 7.1.

In the next section we derive a variational inference algorithm that works with the marginal distribution (integrating out $\mathbf{s}_d$) to perform forward-backward on $\mathbf{z}_d$, and then reintroduces $\mathbf{s}_d$ as an auxiliary variable to perform local variational inference [Bis06].

[1] We will work only with c_{km} and ignore ϕ_{km} from now on.

Algorithm 8 An outline of VI for the HDP-HMM

Iterate the below updates to the variational distributions

$q(\mathbf{z}_d)$: Forward-backward (appendix) with Eq. (7.6) approx.

$q(\theta_k)$: See appendix for discrete HMM case.

$q(\mathbf{s}_d|\mathbf{z}_d)$: See Eq. (7.9). This is used in following updates.

$q(c_{km})$: See Eq. (7.10) and appendix for expectations.

$q(\varepsilon_{km})$ and $q(\zeta_i)$: See appendix.

7.3 Variational inference for the HDP-HMM

The variational objective function is formed by integrating over the model variables in the log joint likelihood using an approximation to the posterior distribution. Since we model each sequence as independent, the joint likelihood of the HDP-HMM can be factorized as

$$\begin{aligned} p(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\zeta}, \varepsilon, \mathbf{c}, \mathbf{z}) &= p(\boldsymbol{\theta})p(\boldsymbol{\zeta})p(\varepsilon)p(\mathbf{c}|\boldsymbol{\zeta}) \\ &\quad \times \textstyle\prod_d p(\mathbf{z}_d|\varepsilon, \mathbf{c}) \prod_i p(x_{di}|\boldsymbol{\theta}, z_{di}), \end{aligned} \tag{7.3}$$

with additional factorizations on all variables in $p(\boldsymbol{\zeta})$, $p(\varepsilon)$, $p(\mathbf{c}|\boldsymbol{\zeta})$ and $p(\boldsymbol{\theta})$. We approximate the posterior of these random variables with the distribution

$$\begin{aligned} q(\boldsymbol{\theta}, \boldsymbol{\zeta}, \varepsilon, \mathbf{c}, \mathbf{z}) &= q(\boldsymbol{\theta})q(\boldsymbol{\zeta})q(\varepsilon)q(\mathbf{c})q(\mathbf{z}) \\ &= \prod_k q(\theta_k)q(\zeta_k) \prod_{k,m} q(\varepsilon_{km})q(c_{km}) \prod_d q(\mathbf{z}_d). \end{aligned} \tag{7.4}$$

We set each variational distribution to be in the same family as the prior (see below for the explicit form). The goal is then to maximize the objective function

$$\mathcal{L} = \mathbb{E}_q[\ln p(\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\zeta}, \varepsilon, \mathbf{c}, \mathbf{z})] - \mathbb{E}_q[\ln q]$$

over the parameters of q to minimize the KL-divergence between q and the posterior [JGJS99].

In the following batch inference algorithm, we work directly with $\mathcal{L}$ to update $q(\mathbf{z}_d)$ and $q(\theta_k)$. Using $\mathcal{L}$ for the remaining q distributions is more difficult, and so we introducing the latent variables $\mathbf{s}_d$ and variational distributions $q(\mathbf{s}_d|\mathbf{z}_d)$ to locally lower bound $\mathcal{L}$. This allows for closed-form updates of $q(\boldsymbol{\zeta})$, $q(\varepsilon)$ and $q(\mathbf{c})$. We focus on the novel aspects of our inference algorithm in the following subsections. The parts of our algorithm that overlap with other HMM inference algorithms are given in the appendix. We sketch one batch iteration in Algorithm 1.

7.3.1 The state transition matrix

The first issue we address is the state transition matrix, which we use to update $q(\mathbf{z}_d)$ and $q(\theta_k)$. Let $A_{kk'}$ be the probability of transitioning from state k to k'.

The challenge here is in the term

$$\mathbb{E}_q \ln A_{kk'} = \mathbb{E}_q \ln \sum_m c_{km,k'} \eta_{km} \tag{7.5}$$

where $\eta_{km} = \varepsilon_{km} \prod_j (1 - \varepsilon_{kj})$. We recall that this is the sum over all sticks that have been assigned to atom $\theta_{k'}$ for the stick-breaking construction of atom θ_k (i.e., state k). We must account for the possible assignment of each stick to $\theta_{k'}$ since the distribution on c_{km} is discrete almost surely. This expectation is not tractable, and so we form a lower bound and an approximation as follows,

$$\begin{aligned} \mathbb{E}_q \ln \textstyle\sum_m c_{km,k'} \eta_{km} &\geq \mathbb{E}_q \ln \sum_m c_{km,k'} e^{\mathbb{E}_q \ln \eta_{km}} \\ &\approx \ln \sum_m \mathbb{E}_q[c_{km,k'}] e^{\mathbb{E}_q \ln \eta_{km}}, \end{aligned} \tag{7.6}$$

We observe that, since we only need the expectation $\mathbb{E}_q \ln \sum_m c_{km,k'} \eta_{km}$ for forward-backward, we could also have sampled c_{km} and η_{km} from their variational q distributions and formed an unbiased approximation. This resulted in a somewhat slower algorithm and we did not empirically observe any difference in performance. We report results with the approximation in Eq. (7.6) in this chapter. We empirically observed that $q(c_{km})$ was nearly deterministic in general, so the approximation to the bound was good.

Making this approximation, we run forward-backward to find $q(\mathbf{z}_d)$ and then use this to update $q(\theta_k)$. These are found as in the parametric HMM (see the appendix).

7.3.2 A local lower bound using $q(\mathbf{s}_d|\mathbf{z}_d)$

Updating the remaining q distributions on ε, $\mathbf{c}$ and $\boldsymbol{\zeta}$ is difficult because our approximation of the expected log state transition probabilities in Eq. (7.6) does not yield tractable variational parameter updates for these variables. We address this with a local lower bounding using the sequence $\mathbf{s}_d$. We recall that this latent sequence interacts with $\mathbf{z}_d$ and c_{km} as follows: The pair $(z_{d,i-1} = k, s_{di} = m)$ indicates that the next state z_{di} can be found by choosing the mth stick of the kth DP, and setting $z_{di} = k'$ if $c_{km,k'} = 1$.

In the variational HMM, the state transition from $z_{d,i-1}$ to z_{di} is captured by the marginal $q(z_{d,i-1}, z_{di})$,

which is calculated using the output of the forward-backward algorithm. For the variational HDP-HMM, we instead model this transition via the triple $(z_{di}, s_{d,i+1}, c_{z_{di},s_{d,i+1}})$. Introducing these variables creates the local lower bound

$$\mathbb{E}_q \mathbb{1}(z_{d,i-1} = k, z_{di} = k') \ln \sum_m c_{km,k'} \eta_{km} \tag{7.7}$$

$$\geq \mathbb{E}_q \sum_m c_{km,k'} \mathbb{1}(z_{d,i-1} = k, s_{di} = m) \ln \eta_{km}.$$

We then define the joint variational distribution

$$q(\mathbf{s}_d, \mathbf{z}_d) = q(\mathbf{s}_d|\mathbf{z}_d) q(\mathbf{z}_d) = q(\mathbf{z}_d) \textstyle\prod_i q(s_{di}|\mathbf{z}_d),$$

where $q(\mathbf{z}_d)$ is already calculated using forward-backward.

The RHS of Eq. (7.7) involves three separate expectations because of the factorization of q. The expectation of η is discussed in the appendix. One novelty introduced by our construction of the HDP-HMM is the term

$$\begin{aligned} q(z_{d,i-1} = k, s_{di} = m) &= \mathbb{E}_q \mathbb{1}(z_{d,i-1} = k, s_{di} = m) \\ &\equiv \xi_{di}(k, m), \end{aligned} \tag{7.8}$$

which is the variational marginal probability of picking the mth stick from the kth DP in step i of sequence d. This value serves a similar purpose as the marginal state transition probability in the parametric HMM, only it is not a distribution between states, but between a state and a stick that must be mapped to a top-level DP state. We find ξ_{di} by calculating

$$\xi_{di}(k, m) \propto \exp\{\mathbb{E}_q \ln p(\mathbf{x}_d, z_{d,i-1} = k, s_{di} = m, -)\},$$

and normalizing ("$-$" indicates all other variables). Similar to the variational HMM, this requires the forward α_d and backward β_d calculations found when updating $q(\mathbf{z}_d)$ (see the appendix). As a result, we update the marginal

$$\xi_{di}(k, m) \propto \alpha_{d,i-1}(k) \exp\{\mathbb{E}_q \ln \eta_{km}\} \times \tag{7.9}$$

$$\textstyle\prod_{k'} [\exp\{\mathbb{E}_q[\ln \theta_{k',x_{di}}]\} \beta_{di}(k')]^{\varphi_{km,k'}}.$$

φ_{km} is the variational multinomial parameter for c_{km} derived later. The difference between this term and the corresponding term for the HMM is the product over the assignment of stick m (which is

Table 7.1: Methods compared with in our experiments. Top half are batch methods and bottom half are stochastic methods.

Method	Notation	Reference
Batch variational HMM	HMM	[Bea03]
Beam sampling HDP-HMM	Beam	[VGSTG08]
Direct assignment variational HDP-HMM	HDPHMM-p	[LPJK07b]
Direct assignment variational HDP-HMM + mean-field assumption	HDPHMM-p-mf	[LPJK07b]
Two-level stick breaking for variational HDP-HMM	HDPHMM-sb	Our method
Stochastic variational HMM	oHMM	[FXLF14]
Direct assignment stochastic variational HDP-HMM	oHDPHMM-p	[JW14a]
Direct assignment stochastic variational HDP-HMM + split-merge	oHDPHMM-sm	[BS12]
Two-level stick breaking for stochastic variational HDP-HMM	oHDPHMM-sb	Our method

known *a priori* in that model). We obtain $\xi_{di}(k,m)$ by normalizing this matrix.

7.3.3 Mapping atoms between DP levels

For c_{km}, being the indicator of the atom associated with the mth stick in the kth DP, we let $q(c_{km,k'} = 1) \equiv \varphi_{km,k'}$ where

$$\varphi_{km,k'} \propto \exp\{\mathbb{E}_q \ln \eta_{0,k'} + \textstyle\sum_{d,i} \xi_{di}(k,m) \mathbb{E}_q \ln \theta_{k',x_{di}}\}. \tag{7.10}$$

We give the expectations in the appendix. A similar calculation appears in the HDP mixed membership model [WPB11a].

7.3.4 Stochastic variational inference

For computationally intensive scenarios in which we have a large collection of sequences over which to learn q, the proposed inference algorithm can be scaled with stochastic variational inference [HBWP13a]. SVI works in this context by subsampling a set of sequences $\mathbf{x}_d$, where $d \in B_t \subset \{1, \ldots, D\}$ at iteration t. It then optimizes the q distributions for these sequences and takes a weighted step in the direction of the natural gradient of the global variational parameters.[2]

Since stochastic inference for $q(\theta_k)$ and $q(\varepsilon_{km})$ are common calculations, we discuss them in the appendix. The stochastic update for $q(c_{km})$ requires the following new SVI derivation. First, we restrict the scaled variational objective function to B_t and terms involving φ_{km},

$$\mathcal{L}^{(t)}_{c_{km}} = \sum_{k'} \varphi_{km,k'} \mathbb{E}_q \ln \eta_{0,k'} - \varphi_{km,k'} \ln \varphi_{km,k'} + \varphi_{km,k'} \tfrac{D}{|B_t|} \sum_{d \in B_t, i} \xi_{di}(k,m) \mathbb{E}_q \ln \theta_{k',x_{di}}. \tag{7.11}$$

[2] We note that our algorithm will scale with the number of sequences, not the length of the sequence as in [FXLF14]. We compare with SVI for the parametric HMM in this many short sequences setting, but still reference [FXLF14] in this case.

The natural parameter of $q(c_{km})$ is $\ln \varphi_{km}$ (taking element-wise logarithm), and so the natural gradient update is

$$\ln \varphi_{km} \leftarrow \ln \varphi_{km} + \rho_t M_{km}^{-1} \nabla_{\ln \varphi_{km}} \mathcal{L}_{c_{km}}^{(t)}, \tag{7.12}$$

$$\begin{aligned} M_{km} &= \mathbb{E}_q \left[\frac{d \ln q(c_{km})}{d \ln \varphi_{km}} \frac{d \ln q(c_{km})}{d \ln \varphi_{km}^T} \right] \\ &= \operatorname{diag}(\varphi_{km}). \end{aligned} \tag{7.13}$$

Next, we observe from Eq. (7.10) that we can write the update of this multinomial distribution in the form

$$q(c_{km,k'} = 1) = \varphi_{km,k'} \propto \exp\{\lambda_{km,k'}\}. \tag{7.14}$$

Swapping in this representation, we find that the natural gradient step over the scale term λ followed by the restriction to the simplex gives the update

$$\varphi_{km,k'} \propto \exp\{\lambda_{km,k'}^{(t)}\}, \tag{7.15}$$

where $\lambda_{km,k'}^{(t)}$ is the typical weighted average

$$\begin{aligned} \lambda_{km,k'}^{(t)} &= (1 - \rho_t)\, \lambda_{km,k'}^{(t-1)} + \rho_t\, \lambda'_{km,k'} \\ \lambda'_{km,k'} &= \mathbb{E}_q \ln \eta_{0,k'} + \frac{D}{|B_t|} \sum_{d \in B_t, i} \xi_{di}(k,m) \mathbb{E}_q \ln \theta_{k',x_{di}} \end{aligned} \tag{7.16}$$

7.4 Experiments

We perform experiments on artificial data, batch inference experiments using the "Alice" dataset and large-scale experiments using discretized audio sequences. We list the methods we compare with in Table 7.1.

7.4.1 Artificial data

In this subsection we demonstrate the effectiveness of our variational HDP-HMM on artificial data. We generate discrete training data of length 1,000 from two four-state HMMs with transition matrices A_{pos}, A_{neg} and emission matrix B set to,

$$A_{pos} = \begin{bmatrix} .99 & .01 & 0 & 0 \\ 0 & .99 & .01 & 0 \\ 0 & 0 & .99 & .01 \\ .01 & 0 & 0 & .99 \end{bmatrix}, \quad A_{neg} = \begin{bmatrix} .01 & .99 & 0 & 0 \\ 0 & .01 & .99 & 0 \\ 0 & 0 & .01 & .99 \\ .99 & 0 & 0 & .01 \end{bmatrix}, \quad B = \frac{1}{3} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

For each scenario we ran 20 experiments with a newly generated sequence and average the results. For the variational HDP-HMM, we truncate the posterior to 10 states and set $a_0 = 1,\ \tau_0 = 1,\ b_0 = 1$. For beam sampling we use the same setting, and we randomly assign each observation to an initial state between 1 and 10. We initialize the variational parameter (see appendix) to $\hat{\theta}_k/100 \sim \text{Dir}(10 \times \mathbf{1})$. For larger state truncations the results were the same, but converged over a longer timescale.

We analyze the convergence of both methods using their respective approximations of the log marginal likelihood on the top of Figure 7.2. For this small-scale problem, the variational method converges in less than 2.5 seconds (∼0.03s per iteration) while beam sampling converges in a longer time (∼0.02s per iteration). Since beam sampling requires multiple samples after the burn-in phase, it requires significantly more computation time. However, the constrained posterior q distribution is restrictive for variational inference, while beam sampling learns a slightly better model in terms of the log marginal likelihood.

We also compare the accuracy of the number of posterior states recovered by both methods. For

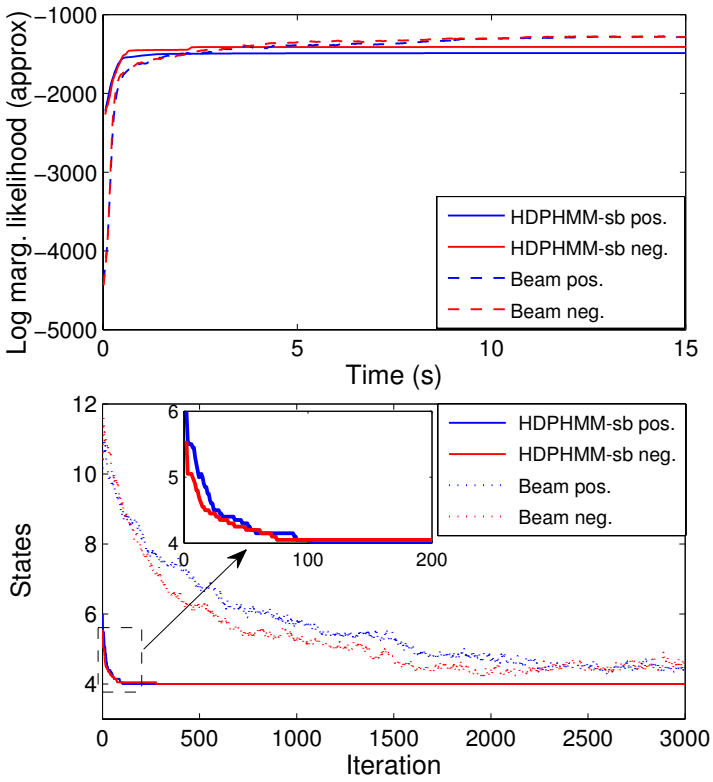


Figure 7.2: Top: Log likelihood for our variational HDP-HMM and beam sampling. Bottom: Number of posterior states inferred by variational HDP-HMM and beam sampling. Results are averaged over 20 random experiments. "pos" indicates A_{pos} and "neg" indicates A_{neg} was used.

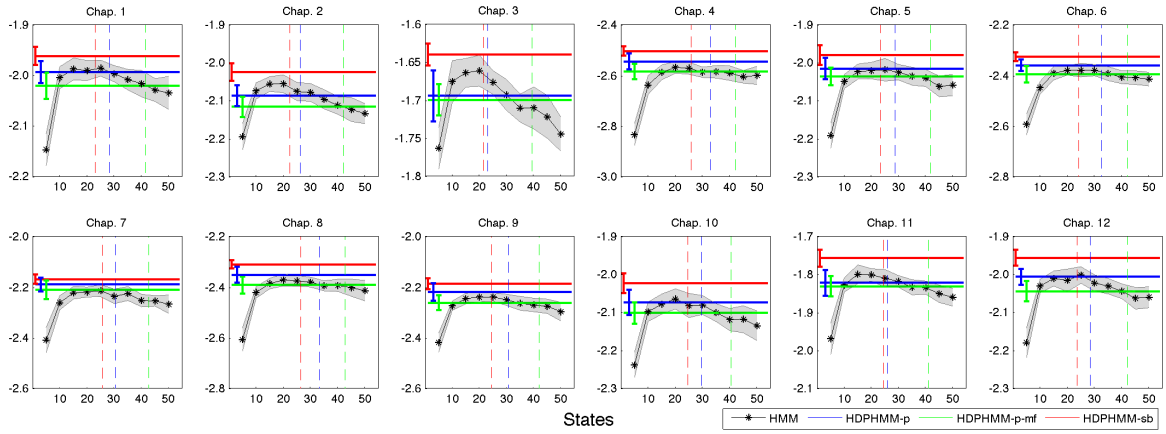


Figure 7.3: Posterior variational bound for each chapter ($\times 10^4$). The black plots represents results for HMM with various number of states. The red, blue, green bars separately show the average result for the HDP-HMM, HDP-HMM with direct assignment, and HDP-HMM with direct assignment and fully-factorized mean-field assumption in variational posterior, all with a truncation level of 50. The standard deviation is shown on the left hand side, for each HDP-HMM model. The dashed lines are the number of states used in the posterior of according models averaged over multiple runs.

variational inference we count the minimum number of occupied states k to cover 99.5% of the data. For beam sampling we record the number of states used per iteration. On the bottom of Figure 7.2, we can see that it takes more than 2,000 iterations (∼40 seconds) for beam sampling to find the true number of states. For variational inference, this requires less than 100 iterations (∼3 seconds).

7.4.2 Alice's Adventures in Wonderland

We also compare our HDP-HMM algorithm with the parametric HMM and the direct assignment approach to the HDP-HMM. We recall that the direct assignment learns a point estimate of the top-level truncated DP and represents each second level with a finite Dirichlet distribution.

We consider a sequential text analysis problem in which we collect 12 chapters in "Alice's Adventures in Wonderland" and filter out all the symbols other than the 26 characters and whitespace. We use these symbols as codewords giving a codebook of size of 27. The entire text sequence was encoded as a sequence of these codewords. For each chapter, we truncate the entire sequence into small chunks of size 200. This gave 663 sequences in total and 55 sequences per chapter on average. We pick 537 sequences for training and hold out the remaining 126 for testing, modeling each chapter separately. For the HMM we use variational inference [Bea03] and tried various number of states K ranging from 5 to 50 with the transition Dirichlet parameter set to $5/K$. For the HDP-HMM we

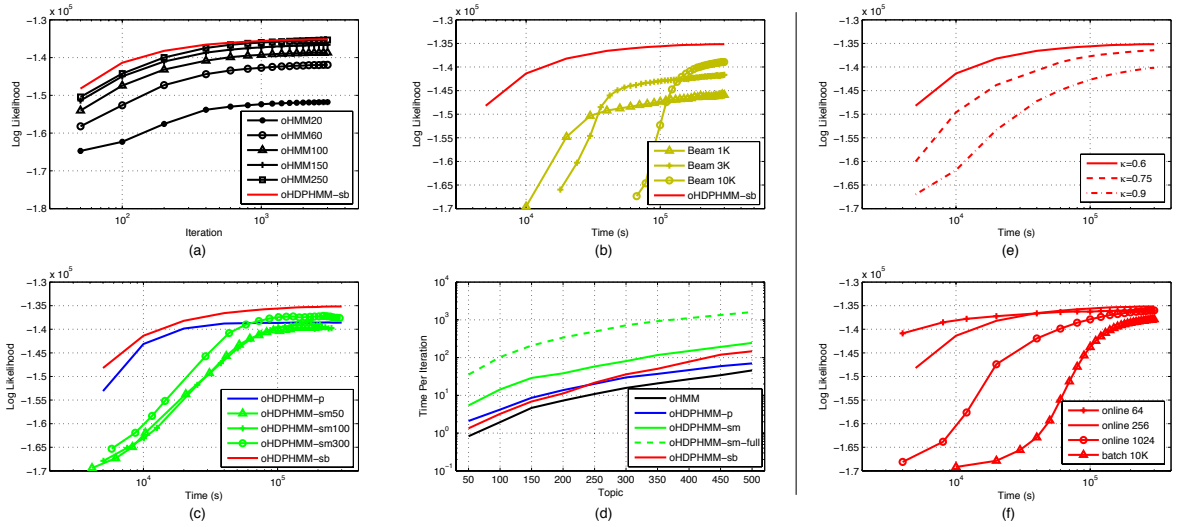


Figure 7.4: (a). Comparison between stochastic HMM and stochastic HDP-HMM. (b). Comparison between stochastic HDP-HMM and beam sampling. (c). Comparison between the two-level stick-breaking stochastic HDP-HMM with the direct assignment stochastic HDP-HMM with and without split-merge updates. (d). Comparison of time cost per iteration among various methods. (e). Comparison among our stochastic HDP-HMM with various learning rate. (f). Comparison among our stochastic HDP-HMMs with various mini-batch sizes and batch HDP-HMMs trained with 10k sequences.

truncate the posterior to 50 states and set the first-level scale parameter $a_0 = 5$ and second-level scale parameter $\tau_0 = 3$. For all models we set the emission Dirichlet scale parameter $b_0 = 1/27$. We ran 20 trials for each experiment. For the performance criterion, we use the predictive variational bound on the test data (using sampling in Eq. (7.5) rather than lower bound).

In Figure 7.3 we show the predictive variational bounds for the HMM (black lines) as a function of state number, and for our HDP-HMM (red lines). For all experiments we show both the mean and standard deviation. From the figure we can see that when the model grows too large, the HMM may overfit the training data, resulting in a drop in predictive performance. For different chapters, the best performance in the HMM varies, which makes model selection more time consuming. For every chapter, our HDP-HMM out performs HMM in predictive performance and learns roughly the ideal number of states according to the HMM. We mark the average number of occupied states in our HDP-HMM posterior with a red vertical line. The number of states varies from 21.4 (Chap. 3) to 26.4 (Chap. 8), which shows flexibility when the data complexity varies.

We also show the result for the HDP-HMM with direct assignment (blue lines). In general, our algorithm converges to a better solution that uses slightly fewer states, indicating the benefit of our representation. In addition to the direct assignment model, we also made a mean-field assumption

in which $q(\mathbf{z}_d) = \prod_i q(z_{di})$ (green lines). This factorization is required in the split-merge stochastic model we compare with [BS12] in the next section. We see that a mean-field assumption on $q(\mathbf{z}_d)$ significantly overestimates the number of states.

7.4.3 Million Song dataset

We also conduct large-scale experiments on discretized audio sequences extracted from the Million Song dataset. We first extract audio features from 371K songs and learn a codebook of size 256 using K-means. We split all the sequences into small chunks of length 50 and learn a single HMM on all sequences.

We compare with the beam-sampled iHMM [VGSTG08], for which we initialized all experiments by randomly assigning observations to one of 500 hidden states, and trained with the same parameter setting as the variational HDP-HMM. For the stochastically-learned parametric HMM models [FXLF14] we set the transition Dirichlet parameter to $20/K$ and the emission Dirichlet parameter to $b_0 = 0.1$. For the HDP-HMM models, including ours and [JW14a], we truncated to 500 states and set $a_0 = 20, \tau_0 = 3, b_0 = 0.1$.

In addition, we compare with the stochastic HDP-HMM using a direct assignment and a split-merge strategy during online learning [BS12]. The split-merge method, originally introduced for stochastic variational HDP, can adaptively create (split) new states or merge old states during each learning iteration in a data-driven manner. In practice we can start with a few states and let the algorithm gradually learn more states. We adapt this method to the HDP-HMM and compare with our method.

For all experiments we ran the algorithms for 3×10^5 seconds ($\sim$3.5 days), during which stochastic HDP-HMM can process around 1 million sequences. We also held out 634 sequences for testing and use the predictive log marginal likelihood on this test set as a performance measure. We use $\rho_t = (100 + t)^{-0.6}$ as the learning rate and $|B_t| = 256$ as mini-batch size.

In Figure 7.4(a) we show the comparison between the parametric stochastic HMM and the non-parametric stochastic HDP-HMM. The predictive log marginal likelihood for stochastic HMM will stop increasing at around 200 to 250 states. The performance for stochastic HDP-HMM is roughly equal to the best of the stochastic HMM.

In Figure 7.4(b) we also compare the stochastic HDP-HMM with beam sampling. Since there is no stochastic solution for beam sampling, we performed batch inference with various amounts of data and present experiments as a function of time in order to compare the efficiency of the algorithms. As

shown, stochastic HDP-HMM outperforms beam sampling because it uses more data. When beam sampling is trained with a limited amount of data (for example, 1K sequences) it will converge more quickly, but the performance will suffer. On the other hand, using too much data for beam sampling will be computationally inefficient. For instance, if we use 10K sequences for training, beam sampling can only draw 70 samples in three days. Beam sampling also did not efficiently infer the number of states. In our experiments, beam sampling will use more than 500 states, while the HDP-HMM occupies around 250 states in its posterior.

In Figure 7.4(c) we compare our stochastic HDP-HMM with the direct assignment method and the split-merge methods. The predictive likelihood for our method outperforms the direct assignment method. For the split-merge method we do three trials by starting with $\{50, 100, 300\}$ states. All three of these cases converged to around 270 states. However, split-merge is restricted to using the fully factorized mean-field assumption over $q(\mathbf{z}_d)$ as discussed previously. Also, we cannot try all split-merge candidates during each online iteration, otherwise split-merge will be computationally prohibitive.[3] Considering all these factors, the split-merge method performs slightly better than the direct assignment method without split-merge, but still worse than our method.

In Figure 7.4(d) we show the time per iteration as a function of state number. HMM is the fastest. Direct assignment also performs fast for large numbers of states since the point estimate of the top-level DP significantly reduces the complexity during learning. Our two-level stick-breaking method is slightly slower than the direct assignment method because of the additional of the posterior complexity. The split-merge method is clearly the slowest, even when we reduce the computation by not checking all possible split-merge moves (green solid). When checking all split-merge options, the algorithm is much slower (green dashed line).

In Figure 7.4(e) we compare our stochastic HDP-HMM as a function of learning rate. For batch size $|B_t| = 256$, we set $\rho_t = (100 + t)^{-\kappa}$ with $\kappa \in \{0.6, 0.75, 0.9\}$. When $\kappa = 0.6$, the learning rate decays more slowly, which gives the best result in our experiments. In Figure 7.4(f) we compare our stochastic HDP-HMM with various batch sizes $|B_t| \in \{64, 256, 1024\}$ and $\kappa = 0.6$, as well as the batch HDP-HMM trained with 10K sequences. Similar to beam sampling, batch HDP-HMM is inefficient in processing large amounts of data. For stochastic algorithms, choosing small mini-batch sizes will result in fast convergence. On the other hand, choosing larger mini-batch sizes can give better performance,

[3]The split-merge strategy requires more computation in addition to the direct assignment method by applying a "restricted iteration" for the split part, and a checking over $K(K-1)/2$ potential candidates (K is the number of states) for the merge part. For details, see [BS12].

but with slower convergence speed.

7.5 Conclusion

We have presented a scalable variational inference algorithm for the HDP-HMM. Using a two-level stick-breaking construction, we were able to infer approximate posteriors of all model variables with closed form updates. We compared our algorithm with beam sampling of the HDP-HMM, the parametric HMM, and the direct assignment methods, showing that our inference algorithm is competitive in batch inference settings, and often better in large scale settings using stochastic variational inference. We observe that this algorithm can be applied more generally to extend other latent Markov modeling frameworks to the nonparametric setting [ZP15a].

7.6 Appendix

Updating $q(\mathbf{z}_d)$ and $q(\theta_k)$: For the forward-backward algorithm we define

$$\tilde{p}(x_{di}|\theta_k) = \exp\{\mathbb{E}_q \ln p(x_{di}|\theta_k)\}, \tag{7.17}$$

$$\tilde{A}_{kk'} = \exp\{\mathbb{E}_q \ln A_{kk'}\}. \tag{7.18}$$

For discrete HMMs we let $q(\theta_k) = \text{Dir}(\hat{\theta}_k)$. In this case we have the same variational expectation as for the parametric model, $\mathbb{E}_q \ln p(x_{di}|\theta_k) = \psi(\hat{\theta}_{k,x_{di}}) - \psi(\sum_j \hat{\theta}_{k,j})$. For Eq. (7.18) we use the approximation in Sec. 7.3.1. We recall that for the variational forward-backward algorithm, $\alpha_{di}(k)$ is the variational joint probability of $z_{di} = k$ and the sequence $\mathbf{x}_d$ up to step i, and $\beta_{di}(k)$ is the variational probability of the sequence $\mathbf{x}_d$ after step i conditioned on $z_{di} = k$ [Bea03]. We then iterate forward over α_{di} and backward over β_{di} as follows,

$$\alpha_{di}(k) = \tilde{p}(x_{di}|\theta_k) \sum_{j=1}^{\infty} \alpha_{d,i-1}(j)\tilde{A}_{jk}, \tag{7.19}$$

$$\beta_{di}(k) = \sum_{j=1}^{\infty} \tilde{A}_{kj}\tilde{p}(x_{d,i+1}|\theta_j)\beta_{d,i+1}(j). \tag{7.20}$$

Having these values, we can make the following update to the marginal of z_{di} for $q(\mathbf{z}_d)$,

$$\gamma_{di}(k) = \frac{\alpha_{di}(k)\beta_{di}(k)}{\sum_j \alpha_{di}(j)\beta_{di}(j)}. \tag{7.21}$$

The variational marginal on the state transition $(z_{di}, z_{d,i+1})$ used for the parametric HMM is not used by our algorithm. Given each γ_{di}, we can update $\hat{\theta}_{k,v}$ in $q(\theta_k) = \text{Dir}(\hat{\theta}_k)$ exactly as in the parametric HMM,

$$\hat{\theta}_{k,v} = b_0 + \sum_{d,i} \gamma_{di}(k) \mathbb{1}(x_{di} = v). \tag{7.22}$$

Updating $q(c)$, $q(\varepsilon)$ and $q(\zeta)$: The update of $q(c)$ uses $\mathbb{E}_q \ln \eta_{0,k'} = \mathbb{E}_q \ln \zeta_{k'} + \sum_{j<k'} \mathbb{E}_q \ln(1-\zeta_j)$. Using the variational distribution $q(\zeta_k) = \text{Beta}(c_k, d_k)$, these expectations are

$$\begin{aligned} \mathbb{E}_q \ln \zeta_k &= \psi(c_k) - \psi(c_k + d_k), \\ \mathbb{E}_q \ln(1-\zeta_j) &= \psi(d_j) - \psi(c_j + d_j). \end{aligned} \tag{7.23}$$

Also, $\mathbb{E}_q \ln \eta_{km} = \mathbb{E}_q \ln \varepsilon_{km} + \sum_j \mathbb{E}_q \ln(1-\varepsilon_{kj})$ used elsewhere is similarly calculated as above, only using $q(\varepsilon_{km}) = \text{Beta}(a_{km}, b_{km})$.

To update $q(\varepsilon_{km}) = \text{Beta}(a_{km}, b_{km})$, we have

$$a_{km} = 1 + \sum_{d,i} \xi_{di}(k,m), \tag{7.24}$$

$$b_{km} = \tau_0 + \sum_{d,i} \sum_{m'>m} \xi_{di}(k,m'). \tag{7.25}$$

As is evident, given the allocations ξ_{di} (defined in Eq.(7.8)), this is simply the expected counts used for updating exchangeable stick-breaking mixture models [BJ06].

Finally, we have the top-level stick-breaking construction update $q(\zeta_k) = \text{Beta}(c_k, d_k)$. We have

$$c_k = 1 + \sum_{k',m} \varphi_{k'm,k}, \tag{7.26}$$

$$d_k = a_0 + \sum_{k',m} \sum_{j>k} \varphi_{k'm,j}. \tag{7.27}$$

where $\varphi_{k'm,k} = \mathbb{E}_q c_{k'm,k}$.

Stochastic inference: For the HDP-HMM, the global q distributions (whose parameter updates are linked to the data size) are on c_{km}, θ_k and ε_{km}. Though ζ_k is also a global parameter, it is conditionally independent of the data given $\mathbf{c}$, and so stochastic inference isn't necessary for this q distribution.

The stochastic updates for $q(\theta_k)$ and $q(\varepsilon_{km})$ are the same as those used by similar models [HBWP13a]. First, form the scaled closed form updates restricted to B_t, denoted $\hat{\theta}'_{k,v}$, a'_{km} and b'_{km}, and then average

with the previous variational parameters,

$$q(\theta_k) : \quad \hat{\theta}_{k,v}^{(t)} = (1-\rho_t)\hat{\theta}_{k,v}^{(t-1)} + \rho_t \hat{\theta}'_{k,v}, \tag{7.28}$$

$$q(\varepsilon_{km}) : \quad a_{km}^{(t)} = (1-\rho_t)a_{km}^{(t-1)} + \rho_t a'_{km},$$

$$b_{km}^{(t)} = (1-\rho_t)b_{km}^{(t-1)} + \rho_t b'_{km}. \tag{7.29}$$

Bibliography

[ABEF14] E.M. Airoldi, D. Blei, E.A. Erosheva, and S.E. Fienberg, editors. *Handbook of Mixed Membership Models and Their Applications*. Chapman and Hall/CRC Handbooks of Modern Statistical Methods, 2014.

[ABFX08] E. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

[ADH10] 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, 2010.

[AEB06] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[AHS13] A. Ahmed, L. Hong, and A. Smola. Nested chinese restaurant franchise processes: Applications to user tracking and document modeling. In *International Conference on Machine Learning*, 2013.

[Ald85] David J Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIIIâĂŤ1983*, pages 1–198. Springer, 1985.

[AM07] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[Bea03] M.J. Beal. Variational algorithms for approximate bayesian inference. *Ph. D. Thesis, University College London*, 2003.

[BF11] David M Blei and Peter I Frazier. Distance dependent chinese restaurant processes. *Journal of Machine Learning Research*, 12(Aug):2461–2488, 2011.

[BFT+13] S. Bacallado, S. Favaro, L. Trippa, et al. Bayesian nonparametric analysis of reversible Markov chains. *The Annals of Statistics*, 41(2):870–896, 2013.

[BGJ10a] D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Proceedings of the National Academy of Sciences*, 57(2):1–30, 2010.

[BGJ10b] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BJ06] D. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

[BJP+12] Tamara Broderick, Michael I Jordan, Jim Pitman, et al. Beta processes, stick-breaking and power laws. *Bayesian analysis*, 7(2):439–476, 2012.

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association (JASA)*, 112(518):859–877, 2017.

[BL06] D. Blei and J. Lafferty. Dynamic topic models. In *International Conference on Machine Learning*, 2006.

[BL07] D. Blei and J. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[BMEWL11] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *International Society for Music Information Retrieval*, 2011.

[BMPJ15] Tamara Broderick, Lester Mackey, John Paisley, and Michael I Jordan. Combinatorial clustering and the beta negative Binomial process. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):290–306, 2015.

[BNJ03a] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BNJ03b] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of machine Learning research (JMLR)*, 3(Jan):993–1022, 2003.

[Bot98] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9), 1998.

[BPJ+13] Tamara Broderick, Jim Pitman, Michael I Jordan, et al. Feature allocations, probability functions, and paintboxes. *Bayesian Analysis*, 8(4):801–836, 2013.

[Bre17] Hervé Bredin. `pyannote.metrics`: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems. *hypothesis*, 100(60):90, 2017.

[BS12] M. Bryant and E. B. Sudderth. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems*, 2012.

[BW+01] Gary Bishop, Greg Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.

[BWJ14] Tamara Broderick, Ashia C Wilson, and Michael I Jordan. Posteriors, conjugacy, and exponential families for completely random measures. *arXiv preprint arXiv:1410.6843*, 2014.

[Car12] François Caron. Bayesian nonparametric models for bipartite graphs. In *Advances in Neural Information Processing Systems*, pages 2051–2059, 2012.

[CB09] J. Chang and D. Blei. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[CCB16] Diana Cai, Trevor Campbell, and Tamara Broderick. Edge-exchangeable graphs and sparsity. In *Advances in Neural Information Processing Systems*, pages 4249–4257, 2016.

[CCB+18] Trevor Campbell, Diana Cai, Tamara Broderick, et al. Exchangeable trait allocations. *Electronic Journal of Statistics*, 12(2):2290–2322, 2018.

[CCD+08] Fabio Castaldo, Daniele Colibro, Emanuele Dalmasso, Pietro Laface, and Claudio Vair. Stream-based speaker segmentation using speaker factors and eigenvoices. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4133–4136. IEEE, 2008.

[CD17] Harry Crane and Walter Dempsey. Edge exchangeable models for interaction networks. *Journal of the American Statistical Association (JASA)*, 2017.

[CF17] François Caron and Emily B Fox. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(5):1295–1366, 2017.

[Çın11] Erhan Çınlar. *Probability and stochastics*, volume 261. Springer Science & Business Media, 2011.

[CNZ18] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[CR17] François Caron and Judith Rousseau. On sparsity and power-law properties of graphs based on exchangeable point processes. *arXiv preprint arXiv:1708.03120*, 2017.

[CRBT13] Changyou Chen, Vinayak Rao, Wray Buntine, and YW Teh. Dependent normalized random measures. In *International Conference on Machine Learning*, 2013.

[CVMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[DF80] P. Diaconis and D. Freedman. De Finetti's theorem for Markov chains. *The Annals of Probability*, 8(1):115–130, 1980.

[DF17] Dimitrios Dimitriadis and Petr Fousek. Developing on-line speaker diarization system. pages 2739–2743, 2017.

[DJ09] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[DKD+11] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[DMVGT09] Finale Doshi, Kurt Miller, Jurgen Van Gael, and Yee Whye Teh. Variational inference for the indian buffet process. In *Artificial Intelligence and Statistics*, pages 137–144, 2009.

[Don95] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[DR06] P. Diaconis and S. W. Rolles. Bayesian analysis for reversible Markov chains. *The Annals of Statistics*, pages 1270–1292, 2006.

[FFRW13] Nicholas Foti, Joseph Futoma, Daniel Rockmore, and Sinead Williamson. A unifying representation for a class of dependent random measures. In *Artificial Intelligence and Statistics*, pages 20–28, 2013.

[FSJW08] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. An HDP-HMM for systems with state persistence. In *International Conference on Machine Learning*, 2008.

[FSJW11] 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.

[FXLF14] N. Foti, J. Xu, D. Laird, and E.B. Fox. Stochastic variational inference for hidden Markov models. In *Advances in Neural Information Processing Systems*, 2014.

[GCWG15] Hong Ge, Yutian Chen, Moquan Wan, and Zoubin Ghahramani. Distributed inference for Dirichlet process mixture models. In *International Conference on Machine Learning (ICML)*, pages 2276–2284, 2015.

[GG06] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems*, pages 475–482, 2006.

[GG11] Thomas L Griffiths and Zoubin Ghahramani. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research (JMLR)*, 12(Apr):1185–1224, 2011.

[GH96a] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, 1996.

[GH96b] Zoubin Ghahramani and Geoffrey E Hinton. Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, 1996.

[GHFZ13] Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28, 2013.

[GJTB04] Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems*, pages 17–24, 2004.

[GO09] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[GP14] San Gultekin and John Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *International Conference on Data Mining (ICDM)*, pages 140–149. IEEE, 2014.

[GRSS+17] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree. Speaker diarization using deep neural network embeddings. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4930–4934. IEEE, 2017.

[GSG07] Z. Ghahramani, P. Sollich, and T.L. Griffiths. Bayesian nonparametric latent feature models. In *Bayesian Statistics 8*. Oxford University Press, 2007.

[HB15] Matthew D Hoffman and David M Blei. Structured stochastic variational inference. In *Artificial Intelligence and Statistics*, 2015.

[HBB10] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, 2010.

[HBC10] M. Hoffman, D. Blei, and P. Cook. Bayesian nonparametric matrix factorization for recorded music. In *International Conference on Machine Learning*, 2010.

[HBWP13a] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1005–1031, 2013.

[HBWP13b] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research (JMLR)*, 14(1):1303–1347, 2013.

[HKG14] Creighton Heaukulani, David Knowles, and Zoubin Ghahramani. Beta diffusion trees. In *International Conference on Machine Learning*, pages 1809–1817, 2014.

[HMBS16] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119. IEEE, 2016.

[Hoo79] D. N. Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute of Advanced Study, Princeton, 1979.

[HR13] C. Heaukulani and D. M. Roy. The combinatorial structure of Beta negative Binomial processes. *arXiv preprint arXiv:1401.0062*, 2013.

[HR15] Creighton Heaukulani and Daniel M Roy. Gibbs-type Indian buffet processes. *arXiv preprint arXiv:1512.02543*, 2015.

[HR+16] Creighton Heaukulani, Daniel M Roy, et al. The combinatorial structure of beta negative Binomial processes. *Bernoulli*, 22(4):2301–2324, 2016.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[IZ02] H. Ishwaran and M. Zarepour. Dirichlet prior sieves in finite normal mixtures. *Statistica Sinica*, 12:941–963, 2002.

[J+17] Lancelot F James et al. Bayesian Poisson calculus for latent feature modeling via generalized Indian buffet process priors. *The Annals of Statistics*, 45(5):2016–2045, 2017.

[JBE+03] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. The icsi meeting corpus. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages I–I. IEEE, 2003.

[JGJS99] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, editors. *An introduction to variational methods for graphical models*. MIT Press, Cambridge, 1999.

[JW14a] M. J. Johnson and A. Willsky. Stochastic variational inference for Bayesian time series models. In *International Conference on Machine Learning*, 2014.

[JW14b] M. J. Johnson and A. S. Wilsky. Stochastic variational inference for Bayesian time series models. In *International Conference on Machine Learning (ICML)*, 2014.

[JZW+18] Ye Jia, Yu Zhang, Ron J Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Conference on Neural Information Processing Systems (NIPS)*, 2018.

[Kal06] Olav Kallenberg. *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KGP14] D. Knowles, Z. Ghahramani, and K. Palla. A reversible infinite HMM using normalised random measures. In *International Conference on Machine Learning (ICML)*, pages 1998–2006, 2014.

[Kin67] John Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[Kin78] John FC Kingman. The representation of partition structures. *Journal of the London Mathematical Society*, 2(2):374–380, 1978.

[KKKO12] J. Kim, D. Kim, S. Kim, and A. Oh. Modeling topic hierarchies with the recursive Chinese restaurant process. In *International Conference on Information and Knowledge Management*, 2012.

[KSS15] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LB06] John D Lafferty and David M Blei. Correlated topic models. In *Advances in Neural Information Processing Systems*, pages 147–154, 2006.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[LBK09] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *International conference on Knowledge discovery and data mining (KDD)*, pages 497–506. ACM, 2009.

[LHE13] D. Liang, M. Hoffman, and D. Ellis. Beta process sparse nonnegative matrix factorization for music. In *International Society for Music Information Retrieval*, 2013.

[LJC16] Juho Lee, Lancelot F James, and Seungjin Choi. Finite-dimensional BFRY priors and variational Bayesian inference for power law models. In *Advances in Neural Information Processing Systems*, pages 3162–3170, 2016.

[LOGR12] James Lloyd, Peter Orbanz, Zoubin Ghahramani, and Daniel M Roy. Random function priors for exchangeable arrays with applications to graphs and relational data. In *Advances in Neural Information Processing Systems*, pages 998–1006, 2012.

[LP05] F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*, 2005.

[LPJK07a] P. Liang, S. Petrov, M. I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, 2007.

[LPJK07b] P. Liang, S. Petrov, M. I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

[LTO13] X. Liu, M. Tanaka, and M. Okutomi. Single-image noise level estimation for blind denoising. *IEEE Transactions on Image Processing*, 22(12):5226–5237, 2013.

[LZZC12] L. Li, X. Zhang, M. Zhou, and L. Carin. Nested dictionary learning for hierarchical organization of imagery and text. In *Uncertainty in Artificial Intelligence*, 2012.

[MCC98] Dilip B Madan, Peter P Carr, and Eric C Chang. The variance gamma process and option pricing. *Review of Finance*, 2(1):79–105, 1998.

[MCF+77] Mark F. Medress, Franklin S Cooper, Jim W. Forgie, CC Green, Dennis H. Klatt, Michael H. O'Malley, Edward P Neuburg, Allen Newell, DR Reddy, B Ritea, et al. Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, 9(3):307–316, 1977.

[MJG09] Kurt Miller, Michael I Jordan, and Thomas L Griffiths. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems*, pages 1276–1284, 2009.

[MWD+18] Philip Andrew Mansfield, Quan Wang, Carlton Downey, Li Wan, and Ignacio Lopez Moreno. Links: A high-dimensional online clustering method. *arXiv preprint arXiv:1801.10123*, 2018.

[MWJ10] L. Mackey, D. Weiss, and M. Jordan. Mixed membership matrix factorization. In *International Conference on Machine Learning*, 2010.

[NCZ17] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *arXiv preprint arXiv:1706.08612*, 2017.

[Nea03] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31:705–741, 2003.

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[NH10a] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning (ICML)*, pages 807–814, 2010.

[NH10b] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML)*, pages 807–814, 2010.

[NLTH06] Huazhong Ning, Ming Liu, Hao Tang, and Thomas S Huang. A spectral clustering approach to speaker diarization. In *INTERSPEECH*, 2006.

[Nor98] J. R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[OR15a] P. Orbanz and D. Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):437–461, 2015.

[OR15b] Peter Orbanz and Daniel M Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2015.

[OW11] Peter Orbanz and Sinead Williamson. Unit–rate poisson representations of completely random measures. *Electronic Journal of Statistics*, pages 1–12, 2011.

[Pau14] M. J. Paul. Mixed membership markov models for unsupervised conversation modeling. In *Empirical Methods in Natural Language Processing*, 2014.

[PBJ12] John Paisley, David Blei, and Michael Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, pages 1367–1374, 2012.

[PC09a] J. Paisley and L. Carin. Hidden markov models with stick breaking priors. *IEEE Transactions on Signal Processing*, 57:3905–3917, 2009.

[PC09b] John Paisley and Lawrence Carin. Nonparametric factor analysis with beta process priors. In *International Conference on Machine Learning*, pages 777–784. ACM, 2009.

[PCB11] John William Paisley, Lawrence Carin, and David M Blei. Variational inference for stick-breaking beta process priors. In *International Conference on Machine Learning*, pages 889–896. Citeseer, 2011.

[PCPK15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[Pit95] Jim Pitman. Exchangeable and partially exchangeable random partitions. *Probability theory and related fields*, 102(2):145–158, 1995.

[Pit06] Jim Pitman. *Combinatorial Stochastic Processes: Ecole d'Eté de Probabilités de Saint-Flour XXXII-2002*. Springer, 2006.

[PJ16] John Paisley and Michael I Jordan. A constructive definition of the beta process. *arXiv preprint arXiv:1604.00685*, 2016.

[PSRD00] J. K. Pritchard, M. Stephens, N. A. Rosenberg, and P. Donnelly. Association mapping in structured populations. *American Journal of Human Genetics*, 67:170–181, 2000.

[PWB+12] John Paisley, Chong Wang, David M Blei, et al. The discrete infinite logistic normal distribution. *Bayesian Analysis*, 7(4):997–1034, 2012.

[PWBJ15a] J. Paisley, C. Wang, D. Blei, and M. Jordan. Nested hierarchical dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):256–270, 2015.

[PWBJ15b] John Paisley, Chong Wang, David M Blei, and Michael I Jordan. Nested hierarchical Dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):256–270, 2015.

[PZW+10] John W Paisley, Aimee K Zaas, Christopher W Woods, Geoffrey S Ginsburg, and Lawrence Carin. A stick-breaking construction of the beta process. In *International Conference on Machine Learning*, pages 847–854, 2010.

[RB18] Rajesh Ranganath and David M Blei. Correlated random measures. *Journal of the American Statistical Association (JASA)*, 113(521):417–430, 2018.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RPW+02] N. A. Rosenberg, J. K. Pritchard, J. L. Weber, H. M. Cann, K. K. Kidd, L. A. Zhivotovsky, and M. W. Feldman. Genetic structure of human populations. *science*, 298(5602):2381–2385, 2002.

[RT+08] Daniel M Roy, Yee Whye Teh, et al. The mondrian process. In *NIPS*, pages 1377–1384, 2008.

[RTB16] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.

[RTCB15] Rajesh Ranganath, Linpeng Tang, Laurent Charlin, and David Blei. Deep exponential families. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 762–771, 2015.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[Sch12] Mark J Schervish. *Theory of statistics*. Springer Science & Business Media, 2012.

[SDDG13] Stephen H Shum, Najim Dehak, Réda Dehak, and James R Glass. Unsupervised methods for speaker diarization: An integrated and iterative approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(10):2015–2028, 2013.

[SDGS13] Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A Smith. Predicting the NFL using Twitter. In *The European conference on machine learning & principles and practice of knowledge discovery in databases (ECML/PKDD)*, 2013.

[Set94a] J. Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[Set94b] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[SGR14] Gregory Sell and Daniel Garcia-Romero. Speaker diarization with plda i-vector scoring and unsupervised calibration. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 413–417. IEEE, 2014.

[SGR15] Gregory Sell and Daniel Garcia-Romero. Diarization resegmentation in the factor analysis subspace. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4794–4798. IEEE, 2015.

[SKG15] A. Shah, D. A. Knowles, and Z. Ghahramani. An empirical study of stochastic variational algorithms for the Beta Bernoulli process. In *International Conference on Machine Learning (ICML)*, 2015.

[SKSD14] Mohammed Senoussaoui, Patrick Kenny, Themos Stafylakis, and Pierre Dumouchel. A study of the cosine distance-based mean shift for telephone speech diarization. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(1):217–227, 2014.

[SP15] S. Sertoglu and J. Paisley. Scalable Bayesian nonparametric dictionary learning. In *European Signal Processing Conference (EUSIPCO)*, 2015.

[SWZ16] Aaron Schein, Hanna Wallach, and Mingyuan Zhou. Poisson-gamma dynamical systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5005–5013, 2016.

[TG09] Yee W Teh and Dilan Gorur. Indian buffet processes with power-law behavior. In *Advances in Neural Information Processing Systems*, pages 1838–1846, 2009.

[TJ07] Romain Thibaux and Michael I Jordan. Hierarchical beta processes and the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, volume 2, pages 564–571, 2007.

[TJBB05] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 1385–1392, 2005.

[TJBB06a] 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, 2006.

[TJBB06b] 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, 2006.

[TRB17] Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.

[VGSTG08] J. Van-Gael, Y. Saatci, Y. W. Teh, and Z. Ghahramani. Beam sampling for the infinite hidden Markov model. In *International Conference on Machine Learning*, 2008.

[VR15] Victor Veitch and Daniel M Roy. The class of random graphs arising from exchangeable random measures. *arXiv preprint arXiv:1512.03099*, 2015.

[WB09] C. Wang and D. Blei. Variational inference for the nested chinese restaurant process. In *Advances in Neural Information Processing Systems*, 2009.

[WB11] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *Knowledge Discovery and Data Mining*, 2011.

[WB12] C. Wang and D. Blei. Truncation-free online variational inference for bayesian nonparametric models. In *Advances in Neural Information Processing Systems*, 2012.

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[WDW+18] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno. Speaker diarization with lstm. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5239–5243. IEEE, 2018.

[WJ+08] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[WMW+18] Quan Wang, Hannah Muckenhirn, Kevin Wilson, Prashant Sridhar, Zelin Wu, John Hershey, Rif A. Saurous, Ron J. Weiss, Ye Jia, and Ignacio Lopez Moreno. Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking. *arXiv preprint arXiv:1810.04826*, 2018.

[WPB11a] C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical Dirichlet process. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[WPB11b] Chong Wang, John Paisley, and David Blei. Online variational inference for the hierarchical Dirichlet process. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 752–760, 2011.

[WWHB10] Sinead Williamson, Chong Wang, Katherine Heller, and David Blei. The IBP compound Dirichlet process and its application to focused topic modeling. 2010.

[WWPM18] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883. IEEE, 2018.

[ZCC15] Mingyuan Zhou, Yulai Cong, and Bo Chen. The Poisson Gamma belief network. In *Advances in Neural Information Processing Systems*, pages 3043–3051, 2015.

[ZCP+09] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric Bayesian dictionary learning for sparse image representations. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2009.

[ZGP16a] A. Zhang, S. Gultekin, and J. Paisley. Stochastic variational inference for the HDP-HMM. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[ZGP16b] Aonan Zhang, San Gultekin, and John Paisley. Stochastic variational inference for the hdp-hmm. In *Artificial Intelligence and Statistics*, pages 800–808, 2016.

[ZHDC12] Mingyuan Zhou, Lauren A Hannah, David B Dunson, and Lawrence Carin. Beta-negative Binomial process and Poisson factor analysis. *Journal of Machine Learning Research (JMLR)*, 2012.

[ZHM17] Zbyněk Zajíc, Marek Hrúz, and Luděk Müller. Speaker diarization using convolutional neural network for statistics accumulation refinement. In *INTERSPEECH*, 2017.

[Zho18] Mingyuan Zhou. Parsimonious Bayesian deep networks. *arXiv preprint arXiv:1805.08719*, 2018.

[ZP15a] A. Zhang and J. Paisley. Markov mixed membership models. In *International Conference on Machine Learning*, 2015.

[ZP15b] Aonan Zhang and John Paisley. Markov mixed membership models. In *International Conference on Machine Learning*, pages 475–483, 2015.

[ZP16] Aonan Zhang and John Paisley. Markov latent feature models. In *International Conference on Machine Learning*, pages 1129–1137, 2016.

[ZPS15] M. Zhou, O. Padilla, and J. Scott. Priors for random count matrices derived from a family of negative binomial processes. *Journal of the American Statistical Association*, (to appear), 2015.

[ZYS+11] Mingyuan Zhou, Hongxia Yang, Guillermo Sapiro, David Dunson, and Lawrence Carin. Dependent hierarchical beta process for image interpolation and denoising. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 883–891, 2011.