Abstract of "Reliable and scalable variational inference for nonparametric mixtures, topics, and sequences" by Michael C. Hughes, Ph.D., Brown University, May 2016.

We develop new algorithms for training nonparametric clustering models based on the Dirichlet Process (DP), including DP mixture models, hierarchical Dirichlet process (HDP) topic models, and HDP hidden Markov models. This family of models has been widely used because Bayesian nonparametric posterior distributions allow coherent comparison of different numbers of clusters for a given fixed dataset. The nonparametric approach is particularly promising for large-scale or streaming applications, where other model selection techniques like cross-validation are far too expensive. However, existing training algorithms fail to live up to this promise. Both Monte Carlo samplers and variational optimization methods are vulnerable to local optima and sensitive to initialization, especially the initial number of clusters. Our new algorithms can reliably escape local optima and poor initializations to discover interpretable clusters from millions of training examples.

For the basic DP mixture model, we pose a variational optimization problem in which the number of instantiated clusters assigned to data can be adapted during training. The focus of this optimization is an objective function which tightly lower bounds the marginal likelihood and thus can be used for Bayesian model selection. Our algorithm maximizes this objective score via block coordinate ascent interleaved with proposal moves that can add useful clusters to escape local optima while removing redundant or irrelevant clusters. We further introduce an incremental algorithm that can exactly optimize our objective function on large datasets while processing only small batches at each step. Our approach uses cached or memoized sufficient statistics to make exact decisions for proposal acceptance or rejection. This memoized approach has the same runtime cost as previous stochastic methods but allows exact acceptance decisions for cluster proposals and avoids learning rates entirely.

We later extend these algorithms to HDP topic models and HDP hidden Markov models. Previous methods for the HDP have used zero-variance point estimates with problematic model selection properties. Instead, we find sophisticated solutions to the non-conjugacy inherent in the HDP that still yield an optimization objective function usable for Bayesian model selection. We demonstrate promising proposal moves for adapting the number of clusters during memoized training on millions of news articles, hundreds of motion capture sequences, and the human genome.

Finally, we show that introducing an additional sparsity constraint to the variational optimization problem for local cluster assignments leads to speed gains without sacrificing model quality. Looking forward, we anticipate possible memoized variational algorithms with adaptive proposal moves for a broad family of Bayesian nonparametric clustering models and suggest potential theoretical guarantees on approximation quality based on data-driven initializations of proposals.

Reliable and scalable variational inference for nonparametric mixtures, topics, and sequences

by
Michael C. Hughes
B. S., Franklin W. Olin College of Engineering, 2010
Sc. M., Brown University, 2012

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2016

This dissertation by Michael C. Hughes is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____

Erik B. Sudderth, Director

Recommended to the Graduate Council

Date _____

Benjamin Raphael, Reader
Dept. of Computer Science, Brown University

Date _____

Emily B. Fox, Reader
Dept. of Statistics, University of Washington

Approved by the Graduate Council

Date _____

Peter W. Weber
Dean of the Graduate School

Vitae

Michael C. Hughes was born on August 2, 1987, in Englewood, Colorado, USA. His family moved to Billings, Montana in 1994 and he received his entire primary and secondary education from Billings public schools, graduating from Billings Senior High in 2006.

He attended Franklin W. Olin College of Engineering in Needham, Massachusetts, graduating in 2010 with a B.S. in Electrical and Computer Engineering. He immediately enrolled in graduate studies at Brown University's Department of Computer Science. He earned his Master's degree from Brown in 2012 and his Ph.D. in 2016. He gratefully acknowledges funding support from the National Science Foundation Graduate Research Fellowship.

Publications related to this thesis

Michael C. Hughes, William Stephenson, and Erik B. Sudderth. *Scalable Adaptation of State Complexity for Nonparametric Hidden Markov Models.* Neural Information Processing Systems (NIPS), 2015.

Michael C. Hughes, Dae Il Kim, and Erik B. Sudderth. *Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process.* Artificial Intelligence & Statistics (AISTATS), 2015.

Michael C. Hughes and Erik B. Sudderth. *Memoized Online Variational Inference for Dirichlet Process Mixture Models.* Neural Information Processing Systems (NIPS), 2013.

Other relevant publications

Emily B. Fox, Michael C. Hughes, Erik B. Sudderth, and Michael I. Jordan. *Joint Modeling of Multiple Time Series via the Beta Process with Application to Motion Capture Segmentation.* Annals of Applied Statistics, Vol. 8(3), 2014.

Michael C. Hughes, Emily Fox, and Erik B. Sudderth *Effective Split-Merge Monte Carlo Methods for Nonparametric Models of Sequential Data.* Neural Information Processing Systems (NIPS), 2012.

Dedicated to my parents

For my mother, Lorie Hughes, my first teacher. She taught me how to read and write, but also to pursue what is right, even if it means staying long after everyone else leaves.

For my father, Gary Hughes, who was the first scientist I ever met and still the one I look up to most. He taught me to ask questions and work hard, but also not to take myself too seriously. Hook, you're a codfish.

Acknowledgements

First and foremost, my thanks go to Erik Sudderth, who was crazy enough to take me on as an advisee from a small school no one had ever heard of and without a lick of machine learning experience. I could not even pronounce the word "Dirichlet" the first time I walked in his door. As a mentor, he imparted on me the importance of pursuing elegant models and rigorous experiments. I am grateful for his seemingly endless patience in guiding me whenever I floundered along the way. Such patience is a rare quality in advisers with his level of brilliance. Finally, I am thankful for the work-life balance that he modeled during my graduate career. People always say that you will become more of an expert than your adviser on your thesis topic, but I only hope I can be half the mentor he is someday.

Thanks go also to my readers and faculty mentors. First, thanks to Emily Fox, who offered crucial guidance during my early years of graduate school as well as during my final year job search. It is very rare these days – and much appreciated – to find a co-author who will print out a manuscript and mark it up by hand. Next, thanks to Ben Raphael for suggesting the chromatin segmentation problem and for many productive discussions at research meetings and in his seminar course. Along with all the faculty I encountered at Brown, I'd like to give a special word of thanks to James Hays, Stefanie Tellex, Michael Littman, Eugene Charniak, and Michael Black for discussions and coursework that shaped how I approach research problems. Finally, thanks to Tom Ouyang and Marc Stogaitis for a productive summer away at Google back in 2013.

I chose Brown for graduate school because I thought it had an extremely supportive community of fellow students, and I was not disappointed. My thanks go to so many: to Soumya Ghosh who convinced me that Erik's group was the real deal and who was a fellow Broncos fan deep in the heart of the empire; to Dae Il Kim for his contagious enthusiasm for topic models and for always knowing where the party was at from Tahoe to Edinburgh; to Jason Pacheco who was always the first person to ask about anything in machine learning and a genuine friend; to Layla Oesper who kept me sane with homemade calzones and who knows when to say "Calamity Jane"; to Ryan Cabeen for many discussions about research and many hours of frisbee, Jurassic Park, and beer; to Silvia Zuffi who was the most supportive officemate I could ask for; to Anna Ritz who helped me make important life decisions over coffee and brunch; to Zhile Ren, who always helped me think about reaching a bigger audience, to Ben Swanson whose zany banjo antics and graphical model grafitti convinced me to come to Brown in the first place; to Deqing Sun for showing the way in vision

reading group, especially in how to give engaging talks; to Betsy Hilliard, who became a staunch friend and confidant despite the soda explosion; to Scott Wylie, who led countless relaxing diversions involving disc golf, rope swings, and lady ducks; and to Jesse Butterfield, who taught me to bet it all on black and introduced me to important Providence landmarks like the GCB and Hot Club. Finally, thanks to Andy and Kristin Loomis, Eric Sodomka, Joe Politz, Ben Lerner, Mike Bryant, Alex Gillmor, Seth Goldenburg, Connor Gramazio and the rest of INEB reading group for helping me de-stress every Wednesday afternoon.

Thanks also to the crew of students I had the pleasure to mentor the past few years. To Geng Ji, who derived variational algorithms for more models in his first two years than I have in my whole career; to Will Stephenson, who proved the HDP surrogate bound and was the first guinea pig for my Python code; to Sonia Phene, who helped me figure out parallelization; to Leah Weiner, who can do great things will topic models; to Gabe Hope, who bravely combined my code with Gaussian processes; to Jake Soloff, who mastered mixture-of-Gaussian observation models; and to Alexis Cook, who will carefully rederive any equation put in front of her. Many others deserve credit too, including Jincheng Li, Mengrui Ni, Oussama Fadil, and Mert Terzihan. The BNPy project would not be where it is today without them to debug my code and keep my crazy ideas in check.

Even before I made it to graduate school, I had many important teachers who taught me even more important lessons . First, from my pre-college years, I give thanks to K. Stuart Smith, who introduced me to computer science and taught me that good food and good conversation is the best way to spend an evening; to Jacquie MacDonald, who taught calculus in public high school better than anyone; to Heather Oberdeck, who helped me realize how to be an adult without losing the best things about being a kid; to Ms. Linda Horst, who showed me in 8th grade just how far mathematics could take me; and to Ms. Robertson, who convinced me that joining math club in 7th grade was not the end of my social life. Boy, was she right. Next, from my undergraduate days at Olin College, I say thanks to Matthew Jadud, who taught me how to tackle research problems on my own; to Lynn Andrea Stein, whose keen advice about graduate school helped me choose Brown in the first place; to John Geddes, who teaches writing even better than he teaches mathematics; and to Allen Downey, whose books continue to inspire me.

I am grateful to many personal friends for their support during the last few years and throughout my life. There are too many to name, but I will try my best. To Kyle Bjordahl, who has been a partner-in-crime, business co-founder, best man and brother since first grade; to Kayton Parekh and Matt Helgeson, who know what do to when a chicken is on deck and Dutch gold is on the line; to Marc Sweetgall, who still holds the record for most nights spent on my couch; to Mike Roenbeck, who knows the key to graduate school stress is a crunchwrap; to Jessi Murray, who could never resist a good pie fight; to Brian Fahrenbach, who is a fellow lifelong member of the local 316, to Pam Heidt, who was always better than me at dodgeball; to Amanda Pratt, who put the electrical in my ECE degree and was there to watch the sun rise on Half Dome; to Casey Canfield, who is always there when you need her, to Katie Miller and George Sass, who have never been anything but generous; to Ben and Karen Salinas, who do the best karaoke in town; and to Sarah and Stefan

Wolpert, who know how to pack a Yeti 110 for a float down the river.

Thanks to my parents, Lorie and Gary Hughes, to whom I dedicate this thesis. Growing up, I always wanted to go to graduate school because they made it sound so much fun. In this and so many other things, they were right. For my older sister, Sarah, who was always helpful crafting posters for my childhood science projects and who also made sure I didn't turn out to be a complete nerd. For my younger sister, Amy, who has always been there to make me laugh and helped me navigate the grind of graduate school. You're killing me, smalls. Next, many thanks to my newest family members: my father-in-law Kailash Mutha, who has been a constant source of wisdom and advice; to Sarita, who taught me the best way to celebrate is with a limosine; and to Sulochana, who was there for many hikes and who knows the value of a good nap.

Finally, my deepest thanks to my wife and best friend, Heena. She was there at the start to encourage me to apply to graduate school, there in the middle of it to keep me well-rounded and well-fed, and here at the end to help me celebrate. No one has my back like she does. We've had many adventures during these last six years: climbing mountains, rafting rivers, chasing sloths, and falling into sewers. Thanks for all the love and laughter along the way. My future is yours.

Contents

List of Figures

List of Algorithms

Reliable and scalable variational inference for nonparametric mixtures, topics, and sequences

Michael C. Hughes

May 2, 2016

Chapter 1

Introduction

A core task of unsupervised machine learning is the problem of discovering an interpretable set of discrete clusters from a complex dataset. For example, social scientists may wish to find thematic word clouds for concepts like "campaign finance" or "immigration policy" from news articles and then browse articles by concept. Biologists may want to discover groups of regulatory proteins that lead to similar gene transcription behavior. Roboticists may wish to identify common actions like "eat cereal" or "take medicine" from videos of daily human activities. By grouping high-dimensional raw data into clusters, we improve our ability to explore and summarize a dataset.

In this thesis, we consider clustering methods based on an underlying probabilistic model. Specifying such a model requires first identifying the relevant random variables. We will treat both the observed data and the data-specific cluster assignments as random variables. We will also treat the parameters which define a cluster's essential properties as random variables, including the appearance probability of each cluster and the parameters that control each cluster-specific data-generation process. Taking a probabilistic view of these quantities allows coherent reasoning under uncertainty.

The next step of modeling is expressing the relationships between the chosen random variables. We will use the formal language of directed graphical models (Lauritzen, 1996; Jordan, 2004) to encode our assumptions about independence and dependence among random variables. Within directed graphical models, we study hierarchical Bayesian models for clustering, especially those based on the exponential family. The foundations of this hierarchical modeling approach are well-established in textbooks such as Gelman et al. (2013) or Bishop (2006). This approach builds expressive models from a set of composable primitive distributions for binary, discrete, and continuous random variables. Modular composition of these primitives can produce complex joint distributions over our variables of interest while still ensuring that training a model from observed data is tractable.

The rest of this chapter introduces three well-known hierarchical Bayesian clustering models: the mixture model for exchangeable datasets, the topic model for grouped data, and the hidden Markov model for sequential data. After motivating each with a detailed intended application, we discuss the classic model selection problem of deciding how many clusters to use for a given dataset. We will see that there exists a Bayesian nonparametric version of each model which can solve this

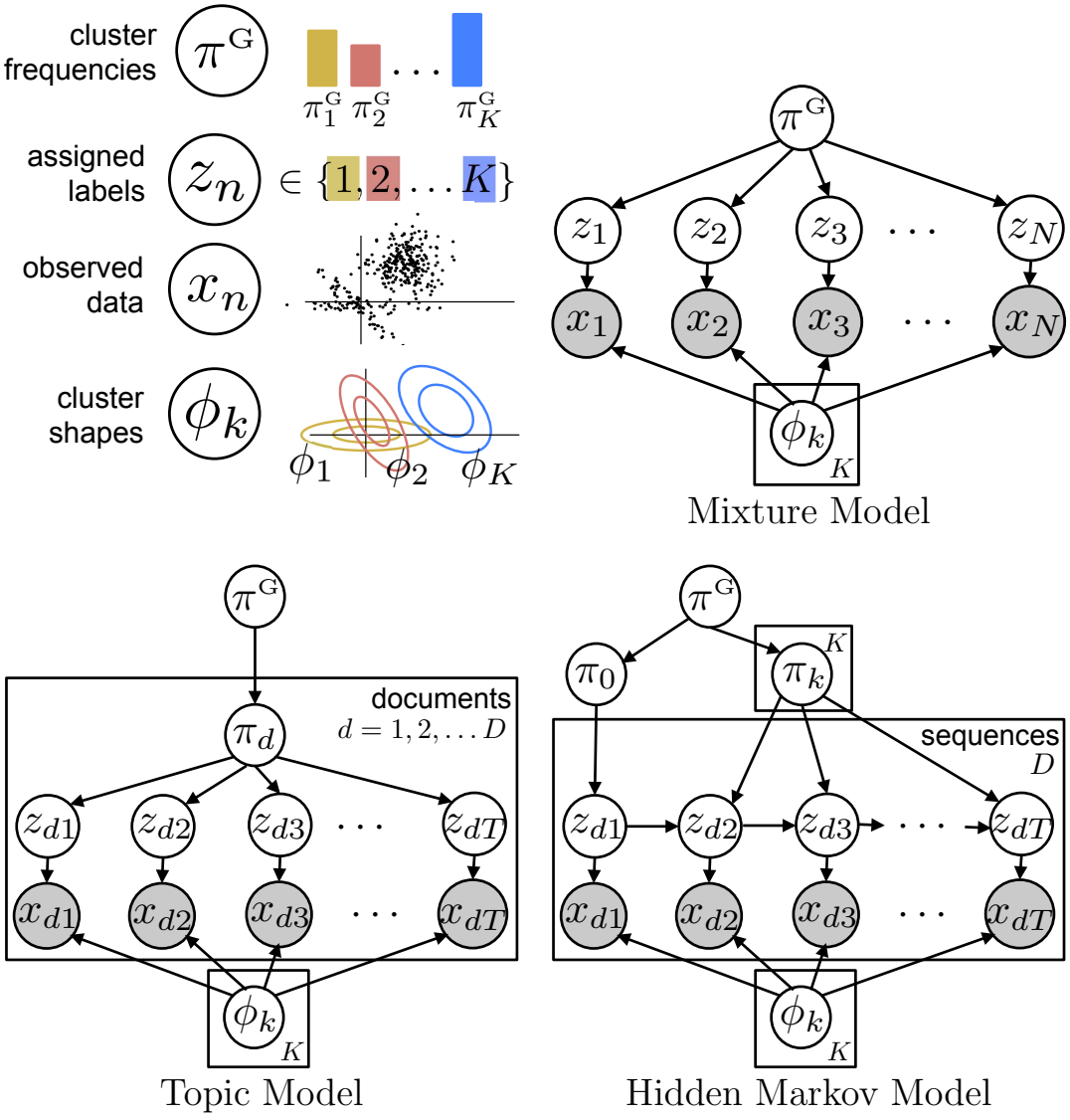

Figure 1.1: Directed graphical representation of mixture, topic, and sequential models. Each model specifies relationships between four random variables: local observed data x_n, local cluster assignments z_n, global cluster frequencies π, and global cluster shape parameters $\{\phi_k\}_{k=1}^K$.

model selection problem elegantly. We will then summarize the core contributions of this thesis: devising efficient algorithms for training Bayesian nonparametric extensions of the mixture model, topic model, and hidden Markov model that simultaneously solve the model selection problem while scaling to large datasets.

1.1 Probabilistic clustering models

This section provides a high-level overview of the generative models discussed in this thesis, which are depicted as directed graphical models in Fig. 1.1. Detailed mathematical description will be provided in Ch. 2 and later.

For now, we will treat each model as having a total of K possible clusters, where the value K is assumed known *a priori*. With this assumption, we can label our clusters with integers $k \in \{1, 2, \dots K\}$. Specifying K in practice leads to issues of *model selection*, which Bayesian nonparametric approaches address by taking an infinite limit $K \to \infty$. We will cover this thoroughly later in Sec. 1.2.

By inspecting the graphical models in Fig. 1.1, we see they are composed entirely of four types of variables: x, z, π, and ϕ. We begin by defining each of these and explaining its role in our family of probabilistic clustering models. We then discuss each model shown in Fig. 1.1 more concretely, explaining the crucial relationships it assumes among these variables and how these matter in applications.

Observed data x. The entire dataset x consists of N total observations. If we assume all may be treated independently, we write $x = \{x_1, x_2, \dots x_N\}$. If instead we have some group structure, such as D documents or D sequences each with T_d observations, then we write $x = \{x_{d1}, x_{d2}, \dots x_{dT_d}\}_{d=1}^{D}$, so that each overall observation with index n has a corresponding d, t index tuple. Generally, each observation x_n may be binary, discrete, or continuous depending on the application. In the illustration of Fig. 1.1, we show an example where each observation x_n is a vector with two real values, specifying a point in the 2-dimensional Cartesian plane.

Assignment labels z. Throughout all the models in this thesis, we assume each model explains the n-th observation x_n by assigning it to exactly one cluster. The random variable $z_n \in \{1, 2, \dots K\}$, which is a discrete integer, indicates the chosen cluster for observation x_n.

Cluster frequencies π. Each model specifies the random variable π^G, which is a length-K vector that is non-negative and sums to one. Each entry π_k^G gives the overall global probability that cluster k is assigned to some observation. We use the superscript G to emphasize that this is a *global* variable not attached to any specific data observation.

Some models specify additional vectors in the set π. Each such *frequency vector* π_j provides an appearance probability for each of the K clusters. Like π^G, each vector π_j must be non-negative and sum to one. Some models have *global* frequency vectors other than π^G, such as the HMM in Fig. 1.1. Other models, such as the topic model, have *local*, document-specific frequency vectors π_d.

Cluster shape parameters ϕ. The set of random variables ϕ contains a parameter vector for every cluster: $\phi = \{\phi_k\}_{k=1}^{K}$. Each vector ϕ_k defines the data-generating distribution of cluster k. For example, if the chosen likelihood model is Gaussian, ϕ_k would define the associated mean and variance parameters. If the chosen likelihood model was Multinomial, ϕ_k would provide the

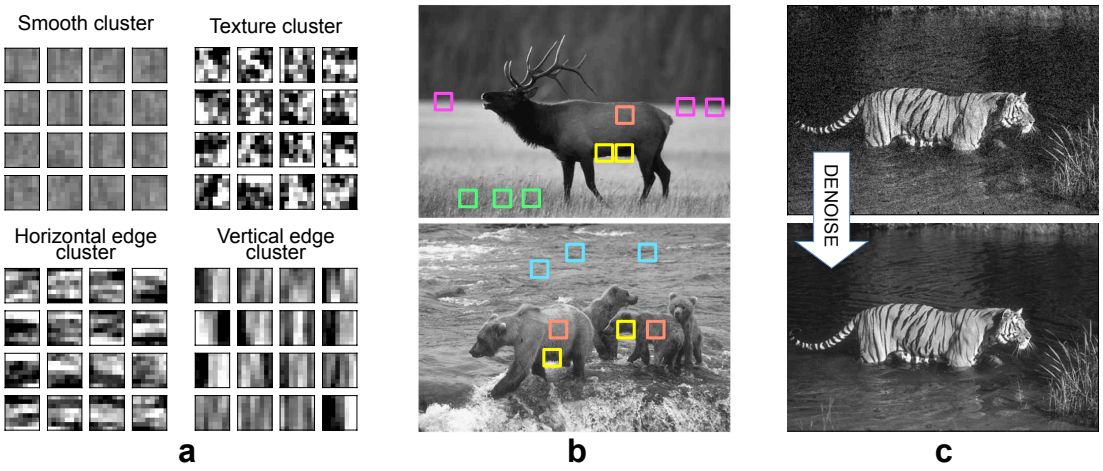

Figure 1.2: Application: discovering expressive models for natural image patches
Full-covariance Gaussian mixture models for 8x8 pixel patches from natural images. *Left (a):* Example patches drawn from four clusters trained on millions of images. Each cluster may represent smooth surfaces, detailed texture patterns, or sharp edges. *Center (b):* Example whole images, along with select patches color-coded by hypothesized cluster. In the top image of an elk, yellow patches have strong horizontal edges while green patches represent textured grass. *Right (c):* Patch-level mixture models can be used within the expected patch log-likelihood (EPLL) framework of Zoran and Weiss (2012) to perform whole-image denoising, as well as deblurring or in-painting (not shown).

probability of each possible discrete outcome. We will sometimes call these variables *observation-model parameters*, because they define how clusters produce observed data. Throughout all the models in Fig. 1.1, the ϕ variables are *global* variables.

1.1.1 Mixture models

The simplest hierarchical Bayesian models we consider are mixture models (Everitt, 1981). In these models, each data observation x_n, such as a text document, an image, or a gene, is assumed to be generated *independently* from one assigned cluster.

Single-membership mixture models have powered many fruitful applications in domains as diverse as medical imaging, computational biology, social science. A particular motivating application for our work is modeling small patches (typically about 8x8 pixels in scale) from natural images. Zoran and Weiss (2011) have shown that using a simple mixture model with full-covariance Gaussian likelihoods, they can discover image patch clusters which are interpretable as smooth surfaces, strong edges, or complex textures, as shown in Fig. 1.2. Furthermore, a trained Gaussian mixture model integrated with clever whole-image reasoning can outperform several baselines for image processing tasks such as image denoising or deblurring (Zoran and Weiss, 2011, 2012).

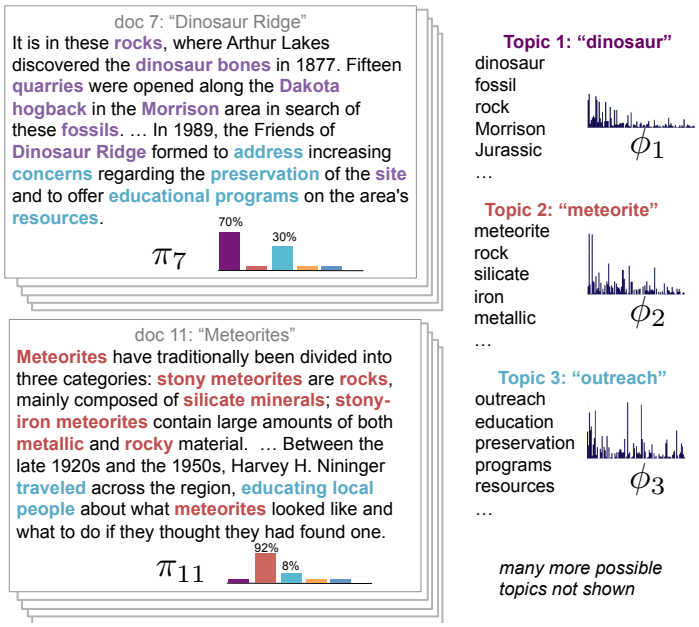

Figure 1.3: Application: discovering common topics from many Wikipedia articles Illustration of insights possible from a topic model trained many Wikipedia articles. To the model, each document is represented as an unordered list of observed words from a fixed vocabulary, sometimes called a "bag of words" representation. The model learns a set of global clusters or topics indexed by k where the data-generating parameters ϕ_k are probability-distributions over the predefined vocabulary. For human consumption, these are often represented by short lists of the top 5 most probable words. The goal of these models is to find a set of topics ϕ which explain the corpus well. For an individual document d, the model can infer the assigned labels z_{dt} of individual words t as well as document-specific probabilities π_d over the possible topics. Each document places probability over a sparse subset of possible topics.

1.1.2 Topic models

In several domains the singular membership assumption of mixture models may be too restrictive. For example, many news articles discuss several thematic topics: a few sentences about campaign finance laws may be followed by discussion of election prospects. Topic models (Blei, 2012) extend mixture models by allowing each group or document to have a specialized mixture of clusters or topics. Such extensions are often called *mixed-membership* clustering models(Airoldi et al., 2009). The most widely-known example is the Latent Dirichlet Allocation topic model (Blei et al., 2003). This model contains a document-specific random variable indicating the percentage of each document which is allocated to each of the possible global clusters or *topics*. Words within the document are then assumed to come from a mixture model with these document-specific cluster weights but common global clusters.

Fig. 1.3 introduces one exciting application of topic models: summarizing many Wikipedia articles. Topic models were popularized for text analysis tasks, but have also been successfully used in genetics (Pritchard et al., 2000; Mimno et al., 2015), action recognition (Wang et al., 2007), and

many other domains.

For the text domain, these models consume the bag-of-words representation of many articles. Our goal is to discover meaningful clusters or *topics* of related words from a finite, predefined vocabulary. Each global *topic* is intuitively a cluster of semantically-related words. More formally, the parameter ϕ_k defining topic k specifies a sparse probability distribution over the fixed vocabulary, with "on-topic" words given high probability. Within a single document, we estimate a mixed-membership probability vector π_d which indicates how often each topic is used in the document.

Overall, estimating this hidden topic structure can lead to improved browsing or exploration of a large corpus. The improved representations π_d for each document could potentially improve downstream tasks like retrieval or recommendation.

1.1.3 Hidden Markov models

When modeling data with assumed sequential structure, we can replace the mixture model's assumption of independence among observations given the global clusters with more structured relationships. For sequential data, hidden Markov models (Rabiner, 1989) specify *pair-wise* relationships between the cluster assignments of neighboring data items in a sequence. This can lead to improved spatial contuinity of learned segmentations, or alternatively to better recovery of transition patterns.

The motion capture segmentation task illustrated in Fig. 1.4 provides one motivating application for the hidden Markov model. Our goal is to discover possible exercise motions such as running, walking or jumping from the motion capture traces gathered from a single human actor. Observed data consists of sensor measurements of the positions and angles of various joints on the human body. These measurements are captured once every 100ms. Given many sequences of these snapshots from several actors, we wish to segment all sequences jointly into continuous blocks of time labelled with distinct exercise motions from a common global set of learned cluster labels. The possible exercise motions are not provided to the algorithm in advance. Instead, they are learned from data.

Fig. 1.5 introduces another motivating application for sequential modeling: a computational biology task called *chromatin state* discovery and segmentation. Within a single chromosome, small functional segments of DNA are wound into bead-like structures called *chromatin*. Gene expression is regulated by external proteins that bind to chromatin. The set of proteins which are present at a given site determines transcription. Given a large dataset measuring the presence or absence of different marker proteins across several DNA sequences, the goal is discovering a small set of *states* or clusters that represent interpretable patterns of spatially co-occurring regulatory proteins. The primary challenge is finding a compact set of states that is small enough to be interpretable, but complex enough to explain biological nuances. Computationally, training effectively from the entire human genome is also a paramount concern.

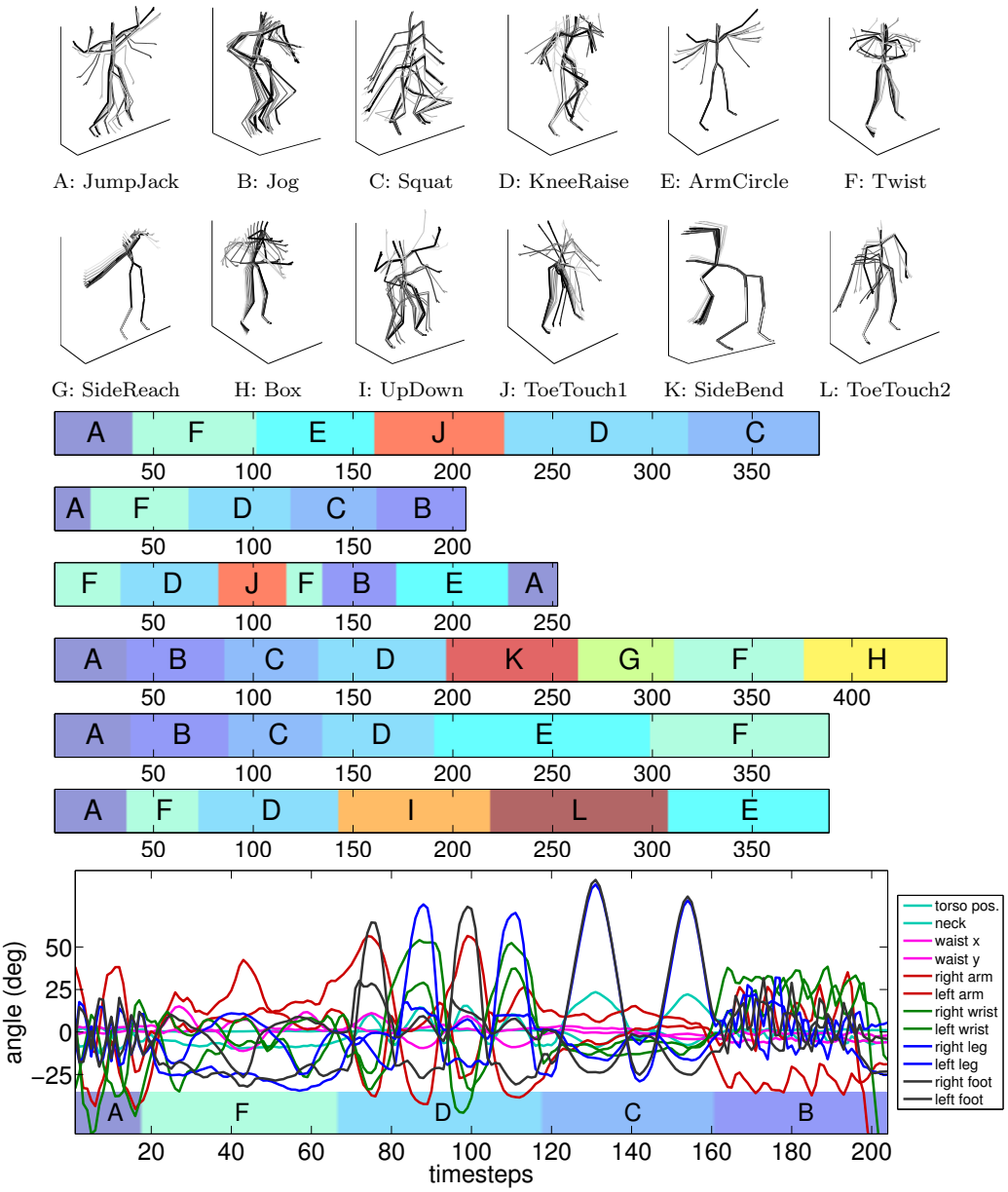

Figure 1.4: Application: segmentation of motion capture sensor traces of human exercises. *Top*: Skeleton visualizations of 12 possible exercise behavior types observed across all sequences. These 12 motions exhaustively cover the motions observed by a human annotator in 6 motion capture sequences. *Middle*: Segmentations z of all 6 observed sequences into the 12 possible exercises. Each sequence was labelled by hand by the same human annotator. *Bottom*: Sequence 2's observed multivariate time series data. Motion capture sensors measure 12 joint angles every 0.1 seconds.

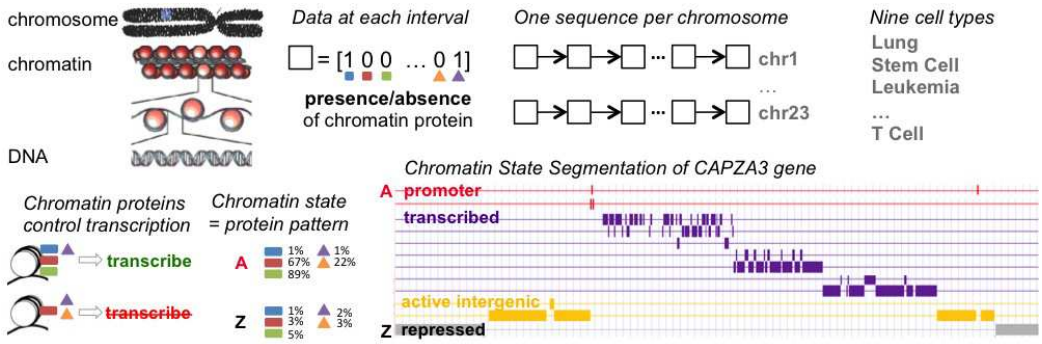

Figure 1.5: Application: segmentation of human genome by regulatory patterns. *Top:* Illustration of chromatin state discovery. Observed data is binary presence/absence of marker proteins that can attach to chromatin DNA to regulate gene expression. For several human cell types, we have one sequence of binary data for each of 23 chromosomes. The goal is to learn several interpretable "states" which define spatially co-occuring patterns of marker proteins. *Lower right:* Ernst and Kellis (2010) used an HMM model to discover 51 biologically plausible states. We show one gene's segmentation under their estimated model for a subset of these states, colored by post-hoc hypothesized role in gene expression.

Other possible clustering models

Single-membership mixture models, mixed-membership topic models, and hidden Markov models for sequences form the three basic model templates we explore in this thesis. Many other useful hierarchical models for clustering are possible but beyond the scope of this thesis. The numerous list of related work includes models with spatial structure for image segmentation (Bouman and Shapiro, 1994; Sudderth and Jordan, 2009), models that apply clusters to relational structures within social networks (Wang and Wong, 1987; Kemp et al., 2006), models which use trees or grammars to establish relationships between data (Finkel et al., 2007; Liang et al., 2007), and models with notions of *multiple-membership*, where each document or group could use any combination of clusters or features without restriction (Griffiths and Ghahramani, 2007). Our hope is that our new approach to training these three basic model templates provides a foundation which could be extended to all related clustering models.

1.2 Bayesian nonparametric models for model selection

Across all clustering models defined above, the conventional model specification requires assuming a fixed number of clusters K. This number determines the flexibility or capacity of the model. Larger values of K can capture finer structure within the observed data at the cost of greater storage and runtime requirements as well as more difficult interpretability of the resulting model. The task of identifying a good (or good enough) value for K is called *model selection*.

The model selection task can often be computationally expensive and difficult in practice. For example, early work on chromatin state segmentation using an HMM required training a model

with $K \approx 100$ states and used ad hoc post-processing to identify a promising set of $K \approx 50$ states (Ernst and Kellis, 2010). Other work assumed a fixed model with $K = 25$ states without rigorous justification beyond the need for a small set of interpretable clusters (Hoffman et al., 2012).

Bayesian nonparametric (BNP) clustering models (Orbanz and Teh, 2010) are a class of hierarchical directed graphical models which extend the conventional mixture, topic, and sequence models via an alternative prior over the global set of cluster random variables π, ϕ. In these models, the *Dirichlet process* (DP) (Ferguson, 1973) prior specifies a generative process which produces a countably infinite set of clusters indexed by $k \in \{1, 2, \ldots k, k+1, \ldots\}$. Each cluster has an associated appearance probability π_k and shape parameter ϕ_k, as shown in Fig. 1.1. We formally define this generative process in Ch. 3. For now, it suffices to realize these models hypothesize an unbounded number of possible clusters, rather than an *a priori* finite set of K possible clusters. Previous work has introduced a Bayesian nonparametric extension of each clustering model we consider. Ferguson (1973) introduced the DP mixture model, which was later connected to an infinite limit of the finite mixture model by Neal (1992). Later, Teh et al. (2006) introduced the hierarchical Dirichlet process (HDP) topic model and the hierarchical Dirichlet process hidden Markov model (HDP-HMM), improving on an earlier "infinite HMM" of Beal et al. (2001).

BNP promise powerful solutions for the model selection task because they inherit the interpretability and extensibility of all hierarchical Bayesian graphical models while avoiding the need to specify the number of clusters in advance. Instead, given a fixed observed dataset, the posterior distribution over the assigned clusters across all N observations gives some probability to *all possible* groupings or segmentations. One possible event is that a single large cluster explains every observation. Another possible event is that each observation comes from its own unique cluster. The total number of possible partition events for a dataset of size N, known as the Bell number, grows exponentially with N. There are 2 possible segmentations of $N = 2$ data items, 15 with $N = 4$, 4140 with $N = 8$ and 10480142147 with $N = 16$. The combinatorics of this space make it impossible to enumerate all possible segmentations. However, BNP models allow coherent reasoning about the posterior probability of events in this combinatorial space. Thus, the cardinality of the number of active clusters can be *learned* from data using these models and need not be specified in advance.

Another advantage of the BNP approach is its natural extension to large-scale or streaming applications. Given a trained model for N observations, the DP provides a coherent mechanism to reuse these existing clusters to explain an additional set of N' observations, possibly using additional clusters for this data as well. The DP prior induces a rich-get-richer property where the *expected* number of active clusters grows logarithmically with the total dataset size. In this large-scale regime, the automatic model selection provided by BNP models is vastly preferred to expensive cross-validation for the number of clusters K. In theory, BNP models could be trained once on a large dataset and deliver a useful clustering which is neither too small (missing crucial structure present in the data) nor too large (containing redundant or irrelevant clusters). However, achieving this automatic model selection in practice is difficult due to limitations of existing training algorithms.

1.3 Existing algorithms: challenges and opportunities

Given a fixed dataset, *training* any clustering model whether finite or Bayesian nonparametric requires deciding the goal of inference. During training, each global variable and local variable can be treated in two possible ways: either we estimate a specific value of the random variable, producing a *point estimate*, or we estimate an *approximate posterior distribution* for the random variable.

For example, a point estimate of the mixture model assignment variable z_n would yield a specific value in the set $\{1, 2, \dots K\}$, where K is the number of clusters in a finite model. In contrast, an approximate posterior distribution $q(z_n)$ for variable z_n could be represented in two ways. First, as vector of probabilities $r_{n1}, \dots r_{nK}$, where r_{nk} gives the probability that cluster k explains data observation n. Second, we could represent the posterior via a finite set of *samples*, $z^{(1)}, z^{(2)}, \dots z^{(S)}$ from this distribution. These samples could be used to easily compute expectations or moments of any functions of the random variable z_n via standard Monte Carlo methods (Gelman et al., 2013).

In this section, we review two prominent approaches in the literature for training clustering models. First, optimization approaches can deliver either point estimates or approximate posterior distributions. These methods set up an optimization problem based on either the joint likelihood $\log p(x, z, \phi, \pi)$ or the marginal likelihood $\log p(x)$. Defining and solving such problems tractably requires calculus of variations, leading to the name *variational optimization methods*. Second, sampling algorithms train models by developing Markov chain Monte Carlo (MCMC) algorithms that deliver samples representing the posterior.

From the outset, we emphasize that our contributions will focus on variational methods that fully instantiate approximate posterior representations for all random variables x, z, π, and ϕ. It is also sometimes possible to integrate away some random variables and obtain an exact representation of the joint density of the remaining variables. Integrating away random variables is sometimes called collapsing or *marginalization*. See Ch. 8 of Bishop (2006) for basic introduction to marginalization. When tractable, this can sometimes lead to improved posterior estimates of remaining variables due to the Rao-Blackwell theorem (Murphy, 2012, Thm. 24.2.1), as advocated by Casella and Robert (1996). Many collapsed MCMC samplers (Griffiths and Steyvers, 2004) and collapsed optimization methods (Teh et al., 2008) for our family of clustering models have been successfully developed. However, these are often difficult to parallelize due to the dependencies induced by marginialization. Because our focus is scalability and simplicity, we will work without marginalization.

1.3.1 Variational optimization algorithms

A wide variety of possible training algorithms exist for the mixture model and its extensions. For finite models, point estimation using a maximum likelihood optimization problem with Gaussian-distributed observations leads in the limit to the well-known k-means algorithm (Lloyd, 1982), which is surveyed extensively by Jain (2010). This approach estimates a single "hard assignment" $z_n \in \{1, 2, \dots K\}$ for each observation n. Kulis and Jordan (2012) developed a variant of the hard-assignment k-means algorithm for the Dirichlet process prior.

In contrast to point estimation of all variables, the expectation-maximization (EM) algorithm (Dempster et al., 1977) for maximum likelihood training estimates an approximate posterior distribution $q(z_n)$ while still producing point estimates for the global variables π, ϕ. These zero-variance point estimation procedures can be applied to Bayesian nonparametric models but will not be useful for model selection, for reasons discussed in Ch. 2. It is also possible to treat global random variables π, ϕ with approximate posterior distributions while using hard point estimates for the local z. This approach is known as maximization-expectation (ME) (Kurihara and Welling, 2009).

Treating all variables π, ϕ, z with approximate posteriors leads to a *variational EM* algorithm, where the optimization problem maximizes an objective that marginalizes rather than conditions on these variables. This approach leads to coherent algorithms for Bayesian nonparametric models because the objective function can be used for model selection, as we discuss in Ch. 2. Marginal likelihood has long been the training objective of choice for Bayesian model selection (Jefferys and Berger, 1992; Rasmussen and Ghahramani., 2001).

As examples of previous work, Blei and Jordan (2006) introduced variational methods for DP mixtures, with later extensions by Kurihara et al. (2007). Variational optimization algorithms for finite hidden Markov models were first developed by Beal (2003) and MacKay (1997). Recently, Johnson and Willsky (2014) developed some more scalable variational training for the HDP-HMM. Variational methods for the finite topic model were first published by Blei et al. (2003) with several extensions to the HDP topic model and other mixed-membership models (Teh et al., 2008; Liang et al., 2007; Wang et al., 2011; Bryant and Sudderth, 2012).

All the approaches described above frame inference as an optimization problem. The objective function is to maximize either a conditional likelihood or a marginal likelihood. This objective takes free parameters for each random variable π, ϕ, z, which define either the point estimate of the variable or its approximate posterior. For mixture models and their extensions, the optimization problem can be solved via block coordinate ascent algorithms, which take as input some initial values of the free parameters and then iteratively update subsets of free parameters while holding others fixed. For all the approaches discussed here, even the simple k-means algorithm, the underlying optimization objective function is non-convex and has many local optima. Depending on the initialization, the final fixed point reached by the optimization may have wildly varying solution quality both in terms of the objective function and in terms of human judgement.

This sensitivity to initialization is especially problematic for variational approaches to Bayesian nonparametric models. Although an infinite number of clusters exists a priori, conventional optimization methods like Blei and Jordan (2006) require instantiating a finite number of clusters at initialization. This finite set is maintained throughout training, because the algorithm contains no steps that add or remove clusters. It may be that some estimated cluster probabilities π_k fall to zero, but these clusters will still be explicitly represented in memory. The true infinite posterior is thus *truncated* to a finite approximate posterior. While some theoretical arguments suggest a large-enough truncation may be satisfactory (Ishwaran and Zarepour, 2002), in practice setting this truncation level too large can lead to slow progress. Selecting an appropriate truncation level thus

reintroduces the very model selection problem that BNP models promised to avoid. Without a very clever initialization, fixed-truncation variational methods for BNP models often reach very poor local optima. The classic advice of performing many random restarts and selecting the best requires too much computation to be practical for large datasets.

1.3.2 MCMC sampling algorithms

Markov chain Monte carlo (MCMC) training algorithms (Andrieu et al., 2003) are widely used for performing approximate posterior inference via sampling. These methods begin by instantiating point-estimate samples of all variables, and then applying carefully constructed Markov transition operators to evolve new samples which gradually approach the true posterior. After many iterations of applying the correct transition operator, the produced samples are guaranteed to asymptotically converge to the true posterior. Samplers have been developed for both finite and infinite versions of the mixture model (Neal, 1992; Escobar and West, 1995; Rasmussen, 1999; Walker, 2007; Jain and Neal, 2004), the topic model (Griffiths and Steyvers, 2004; Yao et al., 2009; Teh et al., 2006), and the hidden Markov model (Scott, 2002; Van Gael et al., 2008; Fox et al., 2011).

In practice, samplers can be slow to make large changes from their initialization. Multiple independent runs using different initializations but applying the same transition operators may thus reach very different regions of the sample space after generous but finite computation budgets. Samplers for hierarchical models are often slow to mix due to the limited range of each conditional transition operator. When holding a large set of variables and only updating a much smaller subset, the number of random samples required to move the whole set of variables to better values may be enormous.

Several improvements have been suggested for mixture models by including transition operators that propose large, joint changes to the whole state space $\{\pi, \phi, z\}$. For example, split and merge proposals (Jain and Neal, 2004) hypothesize splitting an existing data cluster into two smaller subsets or combining two existing clusters into a new larger cluster. These proposals can provide useful improvement over the non-proposal baseline. Recent extensions for DP mixture models, HDP topic models, and Bayesian nonparametric Markov models all report useful gains. Yet these algorithms do not scale well to larger datasets. Furthermore, designing transition operators that maintain the required detailed-balance constraints transition operators becomes increasingly difficult for more complicated models, as seen in the complexity of more recent work in this area (Fox et al., 2014; Chang and Fisher III, 2014).

1.4 Outline of Contributions

This thesis develops new and improved variational optimization algorithms for Bayesian nonparametric models. We design a unified inference framework for a broad class of models including mixtures, topics, and sequences that is both scalable to millions of examples and reliable at escaping poor initialization to recover the ideal set of clusters or states. While not the first effort on

any of these fronts, our contribution lies in our unified approach that pursues elegant optimization problems based on marginal likelihood training objectives and is more reliable than competitors at finding useful, interpretable clusters from uninformed initializations. The outline below specifies our detailed contributions to the field of unsupervised machine learning, which are organized by chapter.

Ch. 2: Background. We first establish the background knowledge for our approach in Ch. 2. This includes careful description of our modeling assumptions as well as some basic optimization algorithms for finite mixture models.

Ch. 3: Scalable inference for DP mixtures. Next, in Ch. 3 we develop a fixed-truncation variational inference algorithm for Dirichlet process mixture models. Building on original work from a NIPS 2013 conference paper (Hughes and Sudderth, 2013), we define a novel optimization problem for mean-field variational inference which makes fewer restrictive assumptions than alternatives. We then develop a block-coordinate ascent optimization algorithms which can be extended to both stochastic and incremental (memoized) settings for processing data at scale. Our memoized algorithm is shown to have the same runtime cost as stochastic but require no nuisance learning rate parameters. We show promising results on several benchmark datasets.

Ch. 4: Reliable inference for DP mixture models via proposal moves. Next, in Ch. 4 we introduce novel proposal moves for variational inference in DP mixture models that can adapt the number of clusters actively represented by the algorithm during training. We show that birth moves can add new useful clusters while merge and delete moves can remove existing clusters which are redundant or irrelevant. We show that these proposal moves are guaranteed to improve our DP mixtures optimization objective, and that they can be deployed on millions of examples via our memoized training algorithm. These proposals were first introduced in Hughes and Sudderth (2013), though significant improvements have been made since.

Ch. 5: Reliable and scalable inference for HDP topic models. In Ch. 5 we develop a new variational optimization problem for the HDP topic model. Due to the challenging non-conjugacy of the HDP generative model, several past authors have used zero-variance point estimates to handle top-level random variables π which define cluster frequencies. Building on an AISTATS 2015 conference paper (Hughes et al., 2015a), we show that these point estimates have terrible model selection properties and suggest instead a tractable surrogate optimization objective that allows proper Bayesian model selection. Our final optimization objective can be optimized via block coordinate ascent algorithms or with stochastic or memoized scalable algorithms. We show that merge and delete proposal moves can produce dramatic improvements in solution quality.

Ch. 6: Reliable and scalable inference for HDP hidden Markov models. In Ch. 6 we extend our earlier results to the HDP hidden Markov model, developing improved optimization objective functions for both the fixed-truncation and adaptive proposal cases. We show results from

several applications, including the motion capture segmentation from Fig. 1.4 and the chromatin state segmentation task of Fig. 1.5. These results extend those published in a NIPS 2015 conference paper (Hughes et al., 2015b).

Ch. 7: Sparse variational posteriors for local assignments. Our final contribution in Ch. 7 is a new framing of the variational optimization problem that allows scaling to large numbers of clusters K as well as large datasets. Specifically, we pursue an new approximate posterior for the discrete cluster assignment variables with sparsity constraints. Standard approaches require estimating a probability for each and every represented cluster. However, at a given single observation only a few clusters of the global total K will have significant explanatory power while the rest will have near zero mass. We suggest an additional constraint: that at most L clusters in this posterior are allowed non-zero mass. We show that this leads to algorithms for both mixtures and topic models where moderate L values are faster to train (2-3x) than the dense baseline $L = K$ but yield comparable heldout predictions.

Ch. 8: Recommendations. We conclude with discussion in Ch. 8 of promising directions for future work, especially connections to semi-supervised learning and probabilistic programming. Our hope is that this thesis inspires optimization algorithms for a broad family of probabilistic clustering models that can simultaneously scale to millions of observed data points and reliably avoid local optima.

1.4.1 Open-source software

As a final contribution of this thesis, all the algorithms presented for mixture models, topic models, and sequential models are available in a public software package called BNPy: Bayesian nonparametrics for Python[1]. We hope this effort will aid reproducibility of our results and allow practioners to use our final algorithms easily on their own data.

[1] https://bitbucket.org/michaelchughes/bnpy-dev/. Accessed May 2016.

Chapter 2

Background

In this chapter, we develop the foundational concepts needed to describe our chosen class of probabilistic clustering models and train these models from observed data. First, we define the probabilistic generative model for all random variables of interest, highlighting a useful two-stage modular decomposition that separates each model into an *allocation* model $p(z, \pi)$ and an observation model $p(x, \phi|z)$. Next, we review key properties of the exponential family of distributions, showing how the sufficient statistics, conjugacy, and information geometry in this family of distributions lead to tractable posterior point estimation or posterior density estimation. Finally, we compare and contrast two possible approaches for optimization-based training of a finite mixture model: posterior point estimation via a k-means-like algorithm and approximate posterior density estimation via variational methods. This background knowledge for the finite mixture model will inform how we approach our novel variational methods for Bayesian nonparametric models in later chapters.

2.1 Hierarchical directed graphical models for clustering

In Fig. 1.1, we illustrate mixture models, topic models, and Hidden markov models as graphical models over related sets of random variables: data x, assignments z, cluster frequency vectors π and cluster shape parameters ϕ. We now establish the fundamental probabilistic relationship between theses random variables. In this chapter, we will assume a finite known set of K clusters. In later chapters, we will show how to easily extend these models to the Bayesian nonparametric case of infinitely many clusters.

Across all the graphical models in Fig. 1.1, we observe that each can be decomposed into two smaller models which generate subsets of the random variables π, ϕ, z, x. First, an *allocation* model that allocates or assigns cluster labels to every observation in a dataset. This process generates both the frequency vectors π and the cluster labels z. Second, the *observation* model generates the shape parameters ϕ, and then produces the data x given fixed cluster labels z. Mathematically, this means

that our chosen family of clustering models has a joint probability density tha factorizes as:

$$p(x, z, \pi, \phi) = \underbrace{p(\pi)p(z|\pi)}_{\text{allocation model}} \cdot \underbrace{p(\phi)p(x|z, \phi)}_{\text{observation model}} . \tag{2.1}$$

This factorization is true for both the finite-capacity models of this chapter as well as the Bayesian nonparametric models of later chapters. For a review of how graphical models imply factorizations in joint density functions, see Jordan (2004) or Bishop (2006).

The differences between the mixture model, topic model, and hidden Markov model are entirely in the form of the allocation model. The structure in Fig. 1.1 shows that all three models have the same overall factorization assumptions for $p(\phi)p(x|z, \phi)$, and thus the same observation model.

The conditional independence implied by our decomposition into allocation and observation models leads to several useful properties. For example, when the assignments z are known or observed, then the posterior for π is independent of the posterior for ϕ. This conditional independence leads to simplified update steps for all our inference algorithms. The separation of models into allocation and observation pieces promotes code reuse and easy composibility, because individual allocation models can be used interchangably with different observation models.

Below, we present the formal assumptions behind both allocation models and observation models, in the context of the models from Fig. 1.1. First, we emphasize a crucial distinction between global and local random variables.

2.1.1 Global vs. local random variables

Each random variable in our joint set x, z, π, ϕ can be identified as either *global* or *local*. This terminology was introduced by Hoffman et al. (2013). We define the set of local variables as those specific to some specific subset of the observed data. Global variables are by definition not local or specific to any particular data.

Examples of local variables. In general, any random variable indexed by n (indicating a specific data atom) or by d (indicating a specific document or sequence) is a local random variable. For example, each x_n is a local random variable, as are the corresponding cluster label variables $z_n \in \{1, 2, \dots K\}$. The document-specific frequencies π_d in the topic model are also local variables.

Examples of global variables. Each model has global cluster shape parameters $\phi = \{\phi_k\}_{k=1}^K$. Additionally, all models have a global frequency vector π^G, which indicates the appearance probability of each cluster k via the value of π_k^G. Both π^G and ϕ define the complete set of global parameters for the mixture and topic models. The HMM in Fig. 1.1 has some addition frequency vectors $\{\pi_j\}_{j=0}^K$ which are global random variables. These are treated thoroughly in Ch. 6.

The distinction between global and local variables matters for several reasons. Essentially, a model is fully-defined by concrete instantiations of each of its global variables. This set of concrete global variables are all that we need to represent a trained model in memory or on disk. Given

this set, we can apply a model to new, never-before-seen data x', or even generate sample data by running the model's generative process forward.

2.1.2 Allocation model: Generating cluster assignments

An allocation model defines a generative model for two sets of random variables: a collection of frequency vectors π and a set of assignment variables z. There is always one assignment variable $z_n \in \{1, 2, 3, \dots K\}$ for each of the N data atoms. However, the size of the set π varies greatly depending on the chosen allocation model. Within the set π, every model includes a top-level frequency vector π^G, which we call the *global* frequency vector. For mixture models, this is the only member of π. More complicated models include additional frequency vectors. For example, the topic model has one for every document: $\{\pi_d\}_{d=1}^D$. The HMM has one frequency vector for every cluster: $\{\pi_k\}_{k=1}^K$. The inclusion of additional frequency vectors in π gives these more complex models increased expressiveness and flexibility.

Following the factorization in Eq. (2.1), the generative process first generates π, and then conditionally generates z given π. Below, we will first define the complete generative process for the mixture model. We will later sketch the process for topic models and HMMs, defering some details to later chapters dedicated to these more complex allocation models.

Allocation process for the mixture model

For the finite mixture model, π^G is a non-negative vector of length K that sums-to-one. We denote the space of such frequency vectors as $\pi^G \in \Delta_K$, where Δ_K refers to the K-dimensional simplex. The natural probability distribution to produce the random variable π^G is the *Dirichlet* distribution (Bishop, 2006, Appendix B). This generative model is formalized as:

$$\pi^G \sim \mathrm{Dir}_K(\pi^G | \frac{\gamma}{K}, \dots \frac{\gamma}{K}) \tag{2.2}$$

The log density $\log p(\pi^G)$ of the vector π^G is

$$\log \mathrm{Dir}(\pi^G|\gamma) = \log \Gamma(\gamma) - \sum_{k=1}^K \log \Gamma(\frac{\gamma}{K}) + \sum_{k=1}^K (\frac{\gamma}{K} - 1) \log \pi_k^G \tag{2.3}$$

Under this Dirichlet distribution, the expected value of this vector is the uniform distribution over K choices:

$$\mathbb{E}[\pi^G] = [\frac{1}{K} \dots \frac{1}{K}]. \tag{2.4}$$

The scalar *concentration* parameter $\gamma > 0$ determines the variance around this mean. Large values $\gamma >> K$ correspond to very low variance, while small values $\gamma << K$ encourage very sparse outcomes for π^G with many entries close to zero.

Given the global frequency vector π^G, the mixture model assumes that each data atom indexed by n has an assigned label $z_n \in \{1, 2, \dots K\}$ drawn independently from the same categorical distribution:

$$z_n \sim \mathrm{Cat}_K(\pi_1^G, \dots \pi_K^G) \tag{2.5}$$

Under this categorical distribution, we have $p(z_n = k) \triangleq \pi_k^G$. We can write the log density as

$$\log p(z_n|\pi^G) = \sum_{k=1}^{K} \delta_k(z_n) \log \pi_k^G, \qquad \delta_k(z_n) = \begin{cases} 1 & \text{if } z_n = k \\ 0 & \text{if } z_n \neq k \end{cases} \tag{2.6}$$

where $\delta_k(z_n)$ is an indicator function which is one if $z_n = k$ and zero otherwise.

Allocation models for topics, sequences, and beyond.

Finite topic model. The topic model in Fig. 1.1 extends the mixture model by allowing each document d its own custom frequency vector π_d, which are tied hierarchically via the global vector π^G. Generatively, we still draw π^G as in Eq. (2.2). Then, we have each document's frequencies drawn independently according to:

$$\pi_d|\pi^G \sim \text{Dir}_K(\alpha\pi_1^G \dots \alpha\pi_K^G) \tag{2.7}$$

Then, each assignment in document d is drawn independently

$$z_{dt} \sim \text{Cat}_K(\pi_{d1}, \dots \pi_{dK}) \tag{2.8}$$

Finite hidden Markov model. The HMM in Fig. 1.1 extends the mixture model by having a custom transition vectors π_k for each cluster k, and by introducing chain-graph dependencies among the assignments z_d made within each sequence d.

Generatively, we still draw π^G as in Eq. (2.2). Then, for each cluster $k \in 1, 2, \dots K$ as well as a special cluster indexed by 0 for the start of each sequence, we have:

$$\pi_j|\pi^G \sim \text{Dir}_K(\alpha\pi_1^G \dots \alpha\pi_K^G) \tag{2.9}$$

Then, the assignments z evolve according to a standard first-order Markov process, with the assigned label at timestep $t > 1$ dependent on the previous timestep $t - 1$:

$$z_{d1} \sim \text{Cat}_K(\pi_{01}, \dots \pi_{0K}), \quad z_{dt}|z_{dt-1} \sim \text{Cat}_K(\pi_{z_{dt-1}1}, \dots \pi_{z_{dt-1}K}) \tag{2.10}$$

We assume that the first timestep in a sequence is always drawn from the same starting distribution with frequencies π_0.

Towards a general specification of allocation models. We can imagine a much broader family of allocation models which include mixture models, topic models, sequence models, and many more. Generally, the collection π can be seen as many frequency vectors $\{\pi_j\}_{j=1}^J$, whose graphical model is a tree-like dependency structure, with each child generated by a Dirichlet distribution with mean equal to its parent and user-supplied variance. Similarly, the collection of assignment random variables z are generated by a graphical model where each group (document, sequence, etc.) has its own tree. Within a given tree, each individual node z_n is generated from Categorical distribution using a specific node π_j from π. The chosen node in π is specified by the parent of z_n in the z graph. Further discussion of this idea is given in Ch. 8.

2.1.3 Observation model: Generating data from assigned clusters

The observation models we consider assume that the cluster assignments z for every data atom are known, and must define a generative model for the pair of random variables ϕ, x. This pair can be decomposed into K individual cluster shape parameters $\{\phi_k\}_{k=1}^K$ as well as the N observed data atoms $\{x_n\}_{n=1}^N$. We then assume the following factorized generative model:

$$\log p(x, \phi|z) = \sum_{k=1}^{K} \log p(\phi_k) + \sum_{k=1}^{K} \sum_{n=1}^{N} \delta_k(z_n) \log p(x_n|\phi_k) \tag{2.11}$$

We will refer to the two probability density functions in the equation above as the *likelihood* L and the *prior* P, where

$$\mathrm{L}(x_n|\phi_k) \triangleq p(x_n|\phi_k) \tag{2.12}$$

$$\mathrm{P}(\phi_k) \triangleq p(\phi_k) \tag{2.13}$$

We will assume each of these distributions belongs to the *exponential family*, and further that the prior P is *conjugate* to the likelihood L. This assumption allows us to perform efficient training under many different observed data scenarios, including binary data, discrete count data, and real-valued data. The next section formalizes this exponential family assumption and its consequences.

2.2 Distributions from the exponential family

By definition, a density belongs to the exponential family if we can write its log density in the form:

$$\log p(x_n|\phi_k) = \phi_k^T s(x_n) - c(\phi) + h(x_n) \tag{2.14}$$

The vector $s(x_n) \in \mathbb{R}^A$ is some *sufficient statistic* vector. It contains all functions of the data atom $x_n \in \mathcal{X}$ necessary for computing the chosen probability density. For example, if data atom x_n is a binary variable ($\mathcal{X} = \{0, 1\}$) and the likelihood density L is Bernoulli, then the sufficient statistic is simply the one-dimensional vector $s(x_n) = [x_n]$. If each data atom x_n is a scalar real value and the likelihood density L is a Gaussian with unknown mean and unknown variance, then vector $s(x_n)$ will be two-dimensional: $s(x_n) = [x_n x_n^2]$. Using these two scalars, both the same mean and sample variance can be computred.

The vector $\phi_k \in \Phi \subset \mathbb{R}^A$ is a *natural parameter* vector. Each possible instantiation of $\phi_k \in \Phi$ defines a single valid density function over the random variable x_n.

The reference measure function $h(x_n)$ contains all terms in the log density that are constant with respect to the parameter ϕ_k. It rarely plays a crucial role in learning tasks, and is needed solely to make sure that the density $\mathcal{L}(x_n|\phi_k)$ integrates to one over the domain of $x_n \in \mathcal{X}$. It is frequently either zero (e.g. $h(x_n) = 0$) or some other constant (e.g. $h(x_n) = \log \sqrt{2\pi}$).

The function $c(\phi)$ is known as a *cumulant* function. This function plays a crucial role, much more so than $h(x_n)$. The function $c(\phi)$ maps every natural parameter to a real scalar: $\Phi \to \mathbb{R}$. Specifically, the cumulant function is defined as an integral over all possible data items $x \in \mathcal{X}$. The

natural parameter space Φ is the set of all ϕ_k vectors for which this integral converges to a finite value.

$$c(\phi_k) \triangleq \log \int_{x \in \mathcal{X}} \exp\left[\phi_k^T s(x_n) + h(x_n)\right], \qquad \Phi = \{\phi_k \in \mathbb{R}^A : c(\phi_k) < +\infty\} \tag{2.15}$$

Only when $\phi_k \in \Phi$ does the function $p(x_n|\phi_k)$ define a valid probability density function that integrates to one over its domain.

2.2.1 Examples of exponential family likelihoods

We now review several examples of exponential family likelihood distributions, contrasting the common way of writing each distribution with the natural parameterization which fits the pattern of Eq. (2.14).

Example: Categorical likelihood

Consider a random variable x_n representing a single word from a finite discrete vocabulary of V possible word. Formally, the value of $x_n \in \{1, 2, \dots V\}$ identifies a particular index of this vocabulary. The exponential family density which explains this data is called the *categorical* distribution.

Categorical: Common form. The common parameterization of the Categorical likelihood for discrete data x_n is:

$$\log p(x_n|\mu_k) = \sum_{v=1}^{V} \delta_v(x_n) \log \mu_{kv} \tag{2.16}$$

where the parameter vector $\mu_k \in \Delta_V$ is a non-negative vector of size V that sums to one. The entry μ_{kv} defines how probable each symbol v.

Categorical: Natural exponential family form. The sufficient statistic vector $s(x_n)$ and natural parameter are both $V-1$ dimensional vectors:

$$\begin{aligned} s(x_n) &= [\delta_1(x_n)\delta_2(x_n) \dots \delta_{V-1}(x_n)] \\ \phi_k &= [\phi_{k1} \ \phi_{k2} \dots \phi_{kV-1}] \end{aligned} \tag{2.17}$$

Each real value ϕ_{kv} can be interpreted as the logarithm of an odds-ratio comparing the probability of the v-th word in the vocabulary to the probability of the last word at index V: $\phi_{kv} = \log \frac{\mu_{kv}}{p(x_n=V)}$. A value near zero indicates these two words are roughly equivalent in probability. A large positive value indicates v is more likely, while a large negative value indicates V is more likely. Generally, the larger that ϕ_{kv} is relative to all other entries in ϕ_k, the more probable that we will choose word v.

The cumulant function $c(\phi_k)$ and the reference measure $h(x_n)$ are:

$$c(\phi_k) = \log\left(1 + \sum_{v=1}^{V-1} e^{\phi_{kv}}\right) \tag{2.18}$$
$$h(x_n) = 0$$

Example: Multinomial likelihood

Common form. A Multinomial density is specified by two parameters: the fixed total count C and a probability vector μ_k. The random variable x_n then has domain $\mathcal{X} = \{x_n \in \mathbb{Z}^V_{\geq 0} : \sum_v x_{nv} = C\}$. That is, this density places probabilty on each possible vector x_n of V non-negative integers that sums to C.

$$\log p(x_n|\mu_k, C) = \log \frac{C!}{\prod_{v=1}^V x_{nv}!} + \sum_{v=1}^V x_{nv} \log \mu_{kv} \tag{2.19}$$

The parameter vector $\mu_k \in \Delta_K$ defines the probability of each possible word type v.

Natural exponential family form. For a Multinomial density over atoms with fixed count C, sufficient statistic vector $s(x_n)$ and natural parameter ϕ_k are both $V-1$ dimensional vectors:

$$s(x_n) = [x_{n1}\ x_{n2} \ldots x_{nV-1}] \tag{2.20}$$
$$\phi_k = [\phi_{k1}\ \phi_{k2} \ldots \phi_{kV-1}]$$

As with the categorical distribution, each real value ϕ_{kv} can be interpreted as the logarithm of the odds ratio of choosing word v to choosing word V. The larger that ϕ_{kv} is relative to other vocabulary words, the more probable that we will choose word v.

The cumulant function $c(\phi_k)$ and the reference measure $h(x_n)$ are then:

$$c(\phi_k|C) = C \log\left(1 + \sum_{v=1}^{V-1} e^{\phi_{kv}}\right) \tag{2.21}$$
$$h(x_n|C) = \log \frac{C!}{\prod_{v=1}^V x_{nv}!}$$

Example: Gaussian univariate likelihood with known variance

Common form. Suppose we have a fixed variance $v > 0$. Then the unknown mean Gaussian density has one parameter, mean m_k. The domain of the random variable x_n is all real numbers: $\mathcal{X} = \mathbb{R}$, and the density of this real random variable is:

$$\log p(x_n|m_k, v) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log v - \frac{1}{2}v^{-1}(x_n - m_k)^2 \tag{2.22}$$

Natural exponential family form. The sufficient statistic vector is one-dimensional, as is the natural parameter $\phi_k \in \mathbb{R}$:

$$s(x_n) = [x_{n1}] \tag{2.23}$$
$$\phi_k = [\phi_{k1}]$$

The cumulant function $c(\phi_k)$ and reference measure $h(x_n)$ are defined as implicit functions of the fixed variance v:

$$c(\phi_k) = \frac{1}{2} v \phi_{k1}^2 \tag{2.24}$$
$$h(x_n) = -\frac{1}{2} v^{-1} x_n^2 - \frac{1}{2} \log v - \frac{1}{2} \log 2\pi$$

Example: Gaussian univariate likelihood with known zero mean

Common form. Suppose we have a fixed mean $m = 0$. Then the unknown-variance form of the Gaussian density has one parameter for each cluster k: the variance v_k. The domain of the random variable x_n is all real numbers: $\mathcal{X} = \mathbb{R}$, and the density of this real random variable is:

$$\log p(x_n | v_k, m = 0) = -\frac{1}{2} \log 2\pi - \frac{1}{2} \log v_k - \frac{1}{2} v_k^{-1} x_n^2 \tag{2.25}$$

Natural exponential family form. The sufficient statistic vector is one-dimensional, as is the natural parameter $\phi_k \in \mathbb{R}$:

$$s(x_n) = [x_{n1}^2] \tag{2.26}$$
$$\phi_k = [\phi_{k1}]$$

The cumulant function $c(\phi_k)$ and reference measure $h(x_n)$ are defined as:

$$c(\phi_k) = -\frac{1}{2} \log(-\phi_k) \tag{2.27}$$
$$h(x_n) = -\frac{1}{2} \log 2\pi$$

2.2.2 Properties of the cumulant function

The cumulant function $c(\phi_k)$ of any exponential family distribution satisfies several useful properties. These facts are discussed in Proposition 2.1.1 of (Sudderth, 2006) and also (Wainwright and Jordan, 2008), but have been long known in the statistics community.

Property 1: All cumulant functions are convex functions. Convexity implies that the Hessian matrix for this function $\nabla^2_{\phi_k} c(\phi_k)$ must be a positive semi-definite $A \times A$ matrix. Additionally, convexity implies that for any two natural parameters $\phi_\ell, \phi_m \in \Phi$ and a scalar $\xi \in [0, 1]$, we have:

$$c(\xi \phi_\ell + (1 - \xi)\phi_m) \leq \xi c(\phi_\ell) + (1 - \xi) c(\phi_m) \tag{2.28}$$

The natural parameter space Φ is also a convex space, meaning the interpolation of any two natural parameters is also a valid natural parameter.

Property 2: Derivatives of the cumulant function correspond to moments of the sufficient statistic. That is:

$$\nabla_{\phi_k} c(\phi_k) = \mathbb{E}_{p(x_n|\phi_k)}\left[s(x_n)\right] \tag{2.29}$$
$$\nabla^2_{\phi_k} c(\phi_k) = \mathbb{E}_{p(x_n|\phi_k)}\left[s(x_n)s(x_n)^T\right] - \mathbb{E}_{p(x_n|\phi_k)}\left[s(x_n)\right]\mathbb{E}_{p(x_n|\phi_k)}\left[s(x_n)^T\right]$$

This second result suggests that the Hessian of the cumulant function is interpretable as the expected covariance matrix of the sufficient statistic vector, which agrees with our positive semi-definite result above.

2.2.3 Mean parameterization and Bregman divergences

The natural parameterization of exponential families from Eq. (2.14) is often not the most convenient for practical analysis. For example, from a given the parameter $\phi_k \in \Phi$ it may be difficult to reason intuitively about what data x_n may be likely under the density $\mathrm{L}(x_n|\phi_k)$, because $x_n \in \mathcal{X}$ lives in a different space than Φ.

Here, we introduce the well-known alternative to using the natural parameter ϕ_k: the *mean* parameterization, using mean vector $\mu_k \in \mathcal{M} \subset \mathbb{R}^A$. Every exponential family density admits a mean parameterization (Wainwright and Jordan, 2008). This is due to the convexity of the function $c(\phi_k)$, which implies (by Legendre duality) a one-to-one invertible transformation between the natural space Φ and the mean space $\mathcal{M}$. We often find the mean parameterization convenient because by definition, every sufficient statistic $s(x_n)$ lives in the mean space $\mathcal{M}$ or its closure (Wainwright and Jordan, 2008).

Using the mean parameterization, we can derive a useful "distance" function between any statistic $s(x_n)$ and any mean parameter μ_k. This pseudo-distance (which is not always a proper metric) is called a Bregman divergence (Banerjee et al., 2005). Practical algorithms can work in either the mean parameter space or the natural parameter space, allowing greater computational flexibility. The subsections below introduce the crucial formal ideas behind the mean parameterization.

Formal definition of the mean parameter

Given a specific exponential family likelihood and its corresponding natural parameter $\phi_k \in \Phi$, there are two complementary formal definitions of the mean parameter μ_k. First, we can define this vector as the expected value of the sufficient statistic given fixed natural parameter ϕ_k:

$$\mu_k \triangleq \mathbb{E}_{p(x_n|\phi_k)}[s(x_n)] \tag{2.30}$$

This suggests that sufficient statistic vectors $s(x_n)$ and mean parameters μ_k live in the same mean parameter space $\mathcal{M}$. This makes the parameter μ_k often easier to interpret, because it occupies the same space as the essential statistics of observed data.

The second definition of the mean parameter μ_k is perhaps more computationally intuitive: we define μ_k via a one-to-one, invertible transformation from the natural parameter space Φ to the

mean space $\mathcal{M}$. Thus, each natural parameter ϕ_k has exactly one corresponding μ_k, and vice-versa (Wainwright and Jordan, 2008; Banerjee et al., 2005). First, we can define the natural-to-mean transformation in terms of the gradient of the cumulant function.

$$\mu_k = \mu(\phi_k) \triangleq \nabla_{\phi_k} c(\phi_k) \tag{2.31}$$

The reverse mean-to-natural transformation is then

$$\phi_k = \phi(\mu_k) \triangleq \nabla_{\mu_k} \gamma(\mu_k) \tag{2.32}$$

where the *conjugate* cumulant function $\gamma(\cdot)$ is the Legendre dual of the cumulant $c(\cdot)$. Formally, this *conjugate* cumulant function $\gamma(\mu_k)$ maps input vectors from the mean parameter space to the real line: $\gamma(\mu_k) : \mathcal{M} \to \mathbb{R}$. The function is also convex and defined as:

$$\gamma(\mu_k) \triangleq \mu_k^T \phi(\mu_k) - c(\phi(\mu_k)) \tag{2.33}$$

Every exponential family density with cumulant function $c(\phi_k)$ has a corresponding conjugate cumulant function $\gamma(\mu_k)$.

The space $\mathcal{M}$ is an *open set*. For example, for the Bernoulli density the mean space is the interval $(0, 1)$, while for a univariate zero-mean Gaussian the mean space is the open interval of valid variances, $(0, +\infty)$. For multivariate zero-mean Gaussians, the mean space is the open set of all possible covariance matrices; that is, all $D \times D$ symmetric, positive-definite matrices.

Let $\mathcal{M}^c$ denote the *closure* of the mean parameter space $\mathcal{M}$. That is, for our Bernoulli example where $\mathcal{M} \triangleq (0, 1)$, then the closure is $\mathcal{M}^c = [0, 1]$. This closure space is important because the sufficient statistic function always lies in the closure: $s(x_n) \in \mathcal{M}^c \subset \mathbb{R}^A$. For example, the sufficient statistic for a Bernoulli data atom is equal to either 0 or 1, which bookend the interval $(0, 1)$. The sufficient statistic for a multivariate zero-mean Gaussian, which is $x_n x_n^T$, is a rank-one matrix that is *positive semi-definite*, thus closing the set of positive-definite matrices.

Given a boundary parameter m that lies in the closure $\mathcal{M}^c$ but not the interior $\mathcal{M}$, we can compute functions like $\gamma(\mu)$ by taking the limit as we move from the interior toward the boundary position m:

$$\gamma(m) = \lim_{||\epsilon|| \to 0^+} \gamma(m + \epsilon), \qquad m \in \mathcal{M}^c \text{ and } m + \epsilon \in \mathcal{M} \tag{2.34}$$

Any other function that takes input from the interior of the mean parameter space can also be computed this way. See Theorem 3.3 of Wainwright and Jordan (2008) for more details.

Bregman divergences

For any convex function $\gamma(\cdot) : \mathcal{M} \to \mathbb{R}$ with dual function $c(\cdot) : \Phi \to \mathbb{R}$, we can compute a non-negative "distance" between two elements of the mean parameter space $\mathcal{M}$ via the *Bregman divergence* function.

$$D_\gamma(\mu_a, \mu_b) \triangleq \gamma(\mu_a) - \gamma(\mu_b) - (\mu_a - \mu_b)^T \phi(\mu_b) \qquad \mu_a, \mu_b \in \mathcal{M} \tag{2.35}$$

The journal article of Banerjee et al. (2005) provides essential coverage of the useful properties of Bregman divergence functions. For our purposes, we emphasize that the output of this function is a non-negative scalar: $D_\gamma(\mu_a, \mu_b) \geq 0$ for all $\mu_a, \mu_b \in \mathcal{M}$, with equality occuring only if $\mu_a = \mu_b$. From the definition above, it is clear this function is assymetric, with $D_\gamma(\mu_a, \mu_b) \neq D_\gamma(\mu_b, \mu_a)$ in general.

The Bregman divergence function is useful because it provides a "distance" between a sufficient statistic vector $s(x_n) \in \mathcal{M}^c$ and a mean parameter $\mu_k \in \mathcal{M}$. While not a proper distance metric because of its assymmetry, computing a pseudo-distance can still be useful, as we show in later distance-biased initialization algorithms. We emphasize that computing the Bregman divergence for sufficient vectors which lie on the closure of the open set $\mathcal{M}$ requires applying the same limiting arguments from Eq. (2.34).

Aside from its utility as a pseudo-distance, we can also write our density function over random variable x_n in terms of the Bregman divergence function. First, we take definition of D_γ in Eq. (2.35) and replace the term $\gamma(\mu_k)$ with its definition in Eq. (2.33). After canceling out the terms $\mu_k^T \phi(\mu_k)$ with opposite signs, we have:

$$D_\gamma(s(x_n), \mu_k) = \gamma(s(x_n)) + c(\phi(\mu_k)) - s(x_n)^T \phi(\mu_k) \tag{2.36}$$

Using Eq. (2.36), we can write the log probability density function for any exponential family distribution (Eq. (2.14)) in terms of the Bregman divergence:

$$\log p(s(x_n)|\phi_k) = -D_\gamma(s(x_n), \mu(\phi_k)) + \gamma(s(x_n)) + h(x_n) \tag{2.37}$$

This highlights the expressive power of the mean parameter transformation and the corresponding Bregman divergence function.

Example: Bregman divergence for Multinomial likelihood

When x_n is a random variable in the domain of all non-negative integer vectors that sum to C, and we assume x_n has a multinomial likelihood, then we have sufficient statistic $s(x_n) = x_n$ and the corresponding Bregman divergence is:

$$D(x_n, \mu_k) = \sum_{v=1}^{V} x_{nv} \log \frac{x_{nv}}{\mu_{kv}} \tag{2.38}$$

where the mean vector μ_k must be non-negative real vector that sums to C: $\sum_{v=1}^{V} \mu_{kv} = C$.

Example: Bregman divergence for fixed-variance 1D Gaussian

When x_n is a random variable in the domain of scalar reals, and we assume x_n has a Gaussian likelihood with fixed variance v and mean parameter μ_k, then we have sufficient statistic $s(x_n) = x_n$ and the corresponding Bregman divergence is:

$$D(x_n, \mu_k) = \frac{1}{2} v^{-1} (x_n - \mu_k)^2 \tag{2.39}$$

where the mean vector μ_k is any real scalar. We can recognise this divergence function as a *Mahalonobis distance* for univariate inputs. This Bregman divergence happens to be symmetric, but symmetry does not hold in general.

Example: Bregman divergence for zero-mean, unknown-variance 1D Gaussian

When x_n is a random variable in the domain of scalar reals, and we assume x_n has a zero-mean Gaussian likelihood with unknown variance $\mu_k > 0$, then we have sufficient statistic $s(x_n) = x_n^2$ and the corresponding Bregman divergence is:

$$D(s(x_n), \mu_k) = -\frac{1}{2}\log\frac{x_n^2}{\mu_k} + \frac{1}{2}x_n^2(\mu_k)^{-1} - \frac{1}{2} \tag{2.40}$$

Here, the mean vector μ_k must be a positive real. Note that the possible edge-case input $x_n = 0$ technically leads to an infinite divergence because of the $\log(x_n)$ in the first term, but all other inputs lie in the proper open set $\mathcal{M}$.

2.2.4 Conjugate Priors

The sections above define the likelihood density $p(x_n|\phi_k) = \mathrm{L}(x_n|\phi_k)$ and its useful properties. We now turn our attention to the prior density $p(\phi_k)$, which is the other required choice to full specify an observation model. While any density function over the space Φ would be a valid choice for $p(\phi_k)$, we constrain our attention to a specific exponential density P known as the *conjugate prior* to the chosen likelihood L. We make this choice because (as we will show in later sections) it leads to a posterior over ϕ_k that comes from the same density family P, with updated parameters. Conjugate priors lead to closed-form expressions for the updated parameters, making algorithms simpler and the mathematical formulas easier to understand. For a textbook introduction, see Sec. 2.4.2 of Bishop (2006).

Log probability density function of the conjugate prior

We will write the log probability density function of the prior distribution P as:

$$\log p(\phi_k|\bar{\nu}, \bar{\tau}) = \bar{\tau}^T\phi_k - \bar{\nu}c(\phi_k) - c^P(\bar{\nu}, \bar{\tau}) \tag{2.41}$$

We can identify this density function as a member of the exponential family, with natural parameter equal to the concatenated vector $[\bar{\tau}\ \ \bar{\nu}]$ and sufficient statistic equal to the concatenated vector $[\phi_k\ \ -c(\phi_k)]$.

The scalar hyperparameter $\bar{\nu} > 0$ defines the effective sample size of the prior. Larger values of $\bar{\nu}$ imply a highly concentrated, low-variance distribution. Small values indicate a high-variance distribution.

The vector hyperparameter $\bar{\tau} \in \mathcal{M}$ is understood as the aggregated mean. Dividing this value by the effective sample size gives the prior density's expected value for the mean parameter: $\mathbb{E}_{p(\phi|\bar{\nu},\bar{\tau})}[\mu(\phi)] = \frac{\bar{\tau}}{\bar{\nu}}$.

The prior cumulant function $c^P(\bar{\nu}, \bar{\tau})$ is defined as:

$$c^P(\bar{\nu}, \bar{\tau}) = \log \int_{\phi_k \in \Phi} e^{\bar{\tau}^T \phi_k - \bar{\nu} c(\phi_k)} d\phi_k \tag{2.42}$$

By definition as an exponential family cumulant function, it is a convex function which can be differentiated to compute the expected values of ϕ_k and $c(\phi_k)$.

Example: Dirichlet prior for Multinomial likelihood

Common form. Traditionally, assuming random variable $\eta_k \in \Delta^V$ is Dirichlet distributed with parameter $\lambda \in \mathbb{R}_+^V$ leads to the log density function:

$$\log p(\eta_k|\lambda) \triangleq c_{\text{Dir}}(\lambda) + \sum_{v=1}^{V} (\lambda_v - 1) \log \eta_k \tag{2.43}$$

where the normalizing constant is defined as:

$$c_{\text{Dir}}(\lambda_1, \lambda_2, \ldots \lambda_V) = \log \Gamma(\textstyle\sum_{v=1}^V \lambda_v) - \sum_{v=1}^V \log \Gamma(\lambda_v) \tag{2.44}$$

Natural conjugate form. Instead, we have a density parameterized by a scalar $\bar{\nu} > 0$ and a vector $\bar{\tau} \in \mathbb{R}_+^{V-1}$, which is constrained so that $\sum_v \bar{\tau}_v \leq \bar{\nu}$.

There exists a one-to-one invertible mapping between the common parameter vector $\lambda \in \mathbb{R}_+^V$ and these natural parameters:

$$\lambda_v(\bar{\tau}, \bar{\nu}) = \begin{cases} \bar{\tau}_v & \text{if } v < V \\ \bar{\nu} - \sum_{w=1}^{V-1} \bar{\tau}_w & \text{if } v = V \end{cases} \tag{2.45}$$

Now, the essential quantities for the natural parameterization of Eq. (2.41) for a Dirichlet density are the natural likelihood parameter $\phi_k \in \mathbb{R}^{V-1}$, the cumulant function $c(\phi_k)$, and the prior cumulant function c^P:

$$\begin{aligned} \phi_k &= [\phi_{k1} \;\; \phi_{k2} \ldots \phi_{kV-1}] \\ c(\phi_k) &= \log(1 + \sum_{v=1}^{V-1} e^{\phi_{kv}}) \\ c^P(\bar{\tau}, \bar{\nu}) &= \left[\sum_{v=1}^{V-1} \log \Gamma(\bar{\tau}_v) \right] + \log \Gamma(\bar{\nu} - \textstyle\sum_{v=1}^{V-1} \bar{\tau}_v) - \log \Gamma(\bar{\nu}) \end{aligned} \tag{2.46}$$

As with the natural form of the Multinomial likelihood, we interpret ϕ_k as a vector containing logarithms of odds ratios. The larger ϕ_{kv} is relative to other entries in ϕ_k, the more likely word v will be.

Under this prior density, we have the useful expectations which follow from moment-generating properties of the prior cumulant function:

$$\begin{aligned} \mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}[\phi_{kv}] &= \nabla_{\bar{\tau}} c^P(\bar{\tau}, \bar{\nu}) = \psi(\bar{\tau}_v) - \psi(\bar{\nu} - \textstyle\sum_{w=1}^{V-1} \bar{\tau}_w) \\ \mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}[c(\phi_k)] &= \nabla_{\bar{\nu}} c^P(\bar{\tau}, \bar{\nu}) = \psi(\bar{\nu} - \textstyle\sum_{w=1}^{V-1} \bar{\tau}_w) - \psi(\bar{\nu}) \end{aligned} \tag{2.47}$$

We can also show that under this conjugate prior, we have

$$\mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}\left[\mu_v(\phi_k)\right] = \frac{\bar{\tau}_v}{\bar{\nu}} \tag{2.48}$$

where $\mu_v(\phi_k)$ is the mean parameter at vocabulary word v. Thus, we interpret the non-negative value $\bar{\tau}_v$ as a pseudocount at word v, and $\bar{\nu}$ as the total pseudocount across all words.

Example: Gaussian prior for fixed-variance Gaussian likelihood

Natural parameterization. Let scalar $\bar{\nu} > 0$ define the effective-sample-size pseudocount and scalar $\bar{\tau}$ define the aggregate mean. Then we have the prior cumulant function defined as an implicit function of fixed variance parameter v:

$$c^P(\bar{\nu}, \bar{\tau}) = \frac{1}{2}v^{-1}\bar{\nu}^{-1}\bar{\tau}^2 - \frac{1}{2}\log[v\bar{\nu}] + \frac{1}{2}\log[2\pi] \tag{2.49}$$

Under this prior, we have the expectations:

$$\mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}\left[\phi_{kv}\right] = \nabla_{\bar{\tau}} c^P(\bar{\tau}, \bar{\nu}) = v^{-1}\bar{\nu}^{-1}\bar{\tau} \tag{2.50}$$

$$\mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}\left[c(\phi_k)\right] = \nabla_{\bar{\nu}} c^P(\bar{\tau}, \bar{\nu}) = -\frac{1}{2}v^{-1}\bar{\nu}^{-2}\bar{\tau}^2 + \frac{1}{2}\bar{\nu}^{-1}$$

as well as the expectations of the mean parameter:

$$\mathbb{E}_{p(\phi_k|\bar{\tau},\bar{\nu})}\left[\mu(\phi_k)\right] = \bar{\nu}^{-1}\bar{\tau} \tag{2.51}$$

We can thus interpret the value $\bar{\tau}$ as determining the expectation of the mean parameter, while the value $\bar{\nu}$ sets the variance.

2.3 Learning observation model parameters from data

The previous sections have described the generative model for data x and cluster shape parameters ϕ which conditions on known cluster assignments z. We now review the two fundamental approaches for global parameter inference of ϕ: point estimation and approximate posterior estimation. First, we emphasize the simplifying role that *sufficient statistics* of the data x and assignments z can play in any estimation procedure for ϕ. We then address maximum likelihood point estimation, maximum-a-posterior (MAP) point estimation, and proper posterior estimation. For alternative presentation of these concepts, see Ch. 9 of the textbook by (Murphy, 2012).

2.3.1 Sufficient statistics

Any procedure for estimating ϕ from data x requires considering the aggregate likelihood for the whole dataset: $p(x|z,\phi)$. Under our assumed exponential family likelihood, we can write the aggregate log likelihood for the whole dataset as

$$\begin{aligned}\log p(x|z,\phi) &= \sum_{k=1}^{K}\sum_{n=1}^{N}\delta_k(z_n)\log p(x_n|\phi_k) \\ &= \sum_{k=1}^{K}\sum_{n=1}^{N}\delta_k(z_n)\Big[\phi_k^T s(x_n) - c(\phi_k) + h(x_n)\Big] \\ &= \sum_{k=1}^{K}\Big(\phi_k^T S_k(x,z) - N_k(z)c(\phi_k)\Big) + \sum_{n=1}^{N}h(x_n)\end{aligned} \tag{2.52}$$

As a function of ϕ_k, this expression has been greatly simplified. The sum of reference measures $h(x_n)$ is independent of ϕ_k and thus can be ignored during any estimation of ϕ_k. The remaining sum over clusters shows that each cluster k may be treated independently.

The quantities $S(x,z)$ and $N(z)$ are *sufficient statistics*, defined as

$$S_k(x,z) \triangleq \sum_{n=1}^{N}\delta_k(z_n)s(x_n) \tag{2.53}$$

$$N_k(z) \triangleq \sum_{n=1}^{N}\delta_k(z_n) \tag{2.54}$$

We can interpret $N_k(z)$ as the total number of observations assigned to cluster k. Similarly, we can interpret $S_k \in \mathcal{R}^A$ as the aggregate data statistic across all observations assigned to cluster k. Importantly, both statistics have dimension independent of the number of data atoms N. Evaluation of the likelihood thus need not scale with the number of data atoms N after the statistics have been computed.

2.3.2 Maximum Likelihood (ML) Point Estimation

We now consider the following optimization problem given data x:

$$\phi_k^* = \arg\max_{\phi_k} \log p(x|z,\phi) \tag{2.55}$$

This can be interpreted as maximizing the likelihood of data x given the point estimates $\{\phi_k\}_{k=1}^K$.

This maximization objective function can be simplified by expanding the form of the likelihood $\mathrm{L}(x_n|\phi_k)$ and dropping terms constant with respect to the vector ϕ_k:

$$\phi_k^* = \arg\max_{\phi_k} \sum_{k=1}^{K}\phi_k^T S_k(x,z) - N_k(z)c(\phi_k) \tag{2.56}$$

Because the cumulant function $c(\phi_k)$ is convex, we know that $\phi_k^T S_k(x,z) - N_k(z)c(\phi_k)$ is a linear function minus a convex function and therefore concave. This implies our maximization problem

has a unique solution, which we find by taking derivatives, setting to zero, and solving for ϕ_k^*:

$$\begin{aligned} 0 &= \nabla_{\phi_k} \left[\phi_k^T S_k(x,z) - N_k(z) c(\phi_k) \right] \\ 0 &= S_k(x,z) - N_k(z) \mu(\phi_k) \end{aligned} \tag{2.57}$$

Remember that the function $\mu(\phi_k)$ is defined as the gradient of the cumulant function, and that it is an invertible function whose inverse μ^{-1} is defined as the function $\phi(\cdot)$ in Eq. (2.32) from the mean parameter space $\mathcal{M}$ to the natural space Φ. Thus, our maximum likelihood point estimate is:

$$\phi_k^* = \phi\left(\frac{S_k(x,z)}{N_k(z)}\right). \tag{2.58}$$

Mean parameter interpretation. By replacing the log density function $\log p(x_n|\phi_k)$ with the equivalent density function parameterized by a Bregman divergence in Eq. (2.37) and dropping constant terms, we have an equivalent optimization problem in terms of the mean parameter μ_k^*:

$$\mu_k^* = \min_{\mu_k \in \mathcal{M}} \sum_{n=1}^{N} D_\gamma(s(x_n), \mu_k) \tag{2.59}$$

This optimization problem over the mean parameters has a unique solution via Theorem 1 of Agarwal and Daumé III (2010).

$$\mu_k^* = \frac{S_k(x,z)}{N_k(z)}. \tag{2.60}$$

The optimal mean parameter is thus always the weighted *sample mean* across all observed sufficient statistics, no matter what exponential family density is used for the observation model likelihood (Gaussian, Multinomial, Bernoulli, etc.). The optimal natural parameter ϕ_k^* is found by mapping the optimal mean μ_k^* into the natural parameter space: $\phi_k^* = \phi(\mu_k^*)$. Substituing μ_k^* from Eq. (2.60) into $\phi(\mu*_k)$, we find we sensibly recover the formula for ϕ_k^* from Eq. (2.58).

2.3.3 Maximum a posteriori (MAP) point estimation

We consider now the problem of point estimation of the parameter ϕ_k for cluster k under the joint distribution of the likelihood L and prior P.

$$\phi_k^* = \arg\max_{\phi_k} \log p(x|z,\phi) + \log(\phi) \tag{2.61}$$

This is equivalent to:

$$\phi_k^* = \arg\max_{\phi_k} \sum_{k=1}^{K} \phi_k^T \Big(S_k(x,z) + \bar{\tau} \Big) - (N_k(z) + \bar{\nu}) c(\phi_k) \tag{2.62}$$

Mean parameter interpretation. By rewriting the densities in terms of Bregman divergences and simplifying, we find the equivalent problem in mean space reduces to:

$$\mu_k^* = \min_{\mu_k \in \mathcal{M}} \left[\bar{\nu} D_\gamma(\tfrac{\bar{\tau}}{\bar{\nu}}, \mu_k) + \textstyle\sum_{n=1}^N D_\gamma(s(x_n), \mu_k) \right] \tag{2.63}$$

This MAP optimization problem has a unique solution via Theorem 2 of Agarwal and Daumé III (2010).

$$\mu_k^* = \frac{S_k(x,z) + \bar{\tau}}{N_k(z) + \bar{\nu}} \tag{2.64}$$

We can interpret μ_k^* as a weighted sample mean, though now the sample includes the observed weighted dataset $\{x_n\}_{n=1}^N$ as well as the prior "pseudo-dataset", which has an effective size of $\bar{\nu}$ atoms and an aggregate sufficient statistic value of $\bar{\tau}$.

Again, we can find the corresponding natural parameter estimate via the mean-to-natural transformation function: $\phi_k^* = \phi(\mu_k^*)$. Thus, we have the MAP estimate:

$$\phi_k^* = \phi\left(\frac{S_k(x,z) + \bar{\tau}}{N_k(z) + \bar{\nu}} \right) \tag{2.65}$$

We emphasize that this is the MAP estimate under the joint density $\log p(x, \phi | z)$. MAP estimates are well-known to depend strongly on the choice of basis (MacKay, 1998). If some other parameterization of the prior is used other than the natural conjugate form $\log p(\phi | \bar{\tau}, \bar{\nu})$, the corresponding MAP estimate will be different.

2.3.4 Posterior estimation

Consider now the problem of determining the posterior distribution $p(\phi | x, z)$. We can accomplish this by inspecting the form of the joint distribution:

$$\begin{aligned} \log p(\phi, x | z) &= \log p(\phi) + \log p(x | z, \phi) \\ &= \text{const} + \sum_{k=1}^K \phi_k^T (S_k(x,z) + \bar{\tau}) - c(\phi_k)(N_k(z) + \bar{\nu}) \end{aligned} \tag{2.66}$$

After gather all terms constant with respect to ϕ_k, we find the remaining terms have the same functional form as the prior P over ϕ_k. Our chosen *conjugacy* between the prior P and likelihood L guarantees that the posterior also belongs to the P density family, with updated parameters:

$$\begin{aligned} p(\phi_k | x, z) &= \mathrm{P}(\phi_k | \hat{\tau}_k, \hat{\nu}_k) \\ \hat{\tau}_k &= S_k(x,z) + \bar{\tau} \\ \hat{\nu}_k &= N_k(x,z) + \bar{\nu} \end{aligned} \tag{2.67}$$

The posterior distribution is specified completely by the two parameters $\hat{\tau}_k$ and $\hat{\nu}_k$.

Under our assumed conjugate exponential family observation model, the algorithmic cost of estimating the full posterior is no greater than the cost of MAP point estimates found earlier.

We further emphasize that this posterior is exact. If both data x and cluster assignments z are fully observed, we can find the posterior $p(\phi_k|x,z)$ in closed form, with the sufficient statistics $N_k(z), S_k(x,z)$ representing all we need to know from x, z.

Next, we examine algorithms for the case where only the dataset x is observed, and assignments z and shape parameters ϕ, and frequencies π^G must be learned from data.

2.4 Point estimation algorithms for finite mixture models

Here, we consider the simplest possible probabilistic clustering model in Fig. 1.1, a finite mixture model with K clusters. The allocation generative process is specified by a single global frequency vector π^G, which we will in this section treat as the *mean parameter* transformation of a random variable $\varphi \in \mathbb{R}^{K-1}$ which has a (natural-form) Dirichlet distribution with effective sample-size γ and expected mean $[\frac{1}{K} \dots \frac{1}{K}]$:

$$\log p(\varphi|\gamma) = \text{const} + \sum_{k=1}^{K-1} \frac{\gamma}{K}\varphi_k - \gamma \log(1 + \sum_{\ell=1}^{K-1} e^{\varphi_\ell}) \tag{2.68}$$

$$\pi_k(\varphi) \triangleq \frac{e^{\varphi_k}}{1 + \sum_{\ell=1}^{K-1} e^{\varphi_\ell}} \tag{2.69}$$

Under this parameterization, we have

$$\mathbb{E}[\pi_k(\varphi)] = \frac{1}{K} \tag{2.70}$$

Given the fixed value for the frequency vector $\pi(\varphi)$, we generate each data atom assignment z_n independently:

$$z_n|\pi(\varphi) \sim \text{Cat}_K(\pi_1, \pi_2, \dots \pi_K) \tag{2.71}$$

Then, given the observed assignments z, we have our standard conjugate-exponential-family observation model:

$$\phi_k \sim \text{P}(\phi_k|\bar{\tau}, \bar{\nu}), \qquad x_n|z_n, \phi \sim \text{L}(x_n|\phi_{z_n}) \tag{2.72}$$

Our first goal will be an algorithm for point estimation for each of the hidden global random variables φ, ϕ and local assignments z. We will later extend this to a full variational method for approximate posteriors.

Algorithm 2.1 Bregman k-means for point-estimation of finite mixture model

Input:

$\{x_n\}_{n=1}^N$: dataset with N exchangeable observations
$\gamma > 0$: hyperparameters defining the allocation-model prior $p(\varphi|\gamma)$
$\bar{\tau}, \bar{\nu}$: hyperparameters defining the prior $\mathrm{P}(\phi_k|\bar{\tau}, \bar{\nu})$
$\{\mu_k\}_{k=1}^K$: Initial point estimates of observation-model mean parameters, $\mu_k \in \mathcal{M}$
π : Initial point estimate of allocation-model frequency vector

Output:

$\{z_n\}_{n=1}^N$: Point estimates of hard assignments
$\{\mu_k\}_{k=1}^K$: Point estimates of observation-model mean parameters
π : Point estimates of allocation-model frequency vector

function BregmanKMeans$((x, \bar{\tau}, \bar{\nu}, \gamma, \{\mu_k\}_{k=1}^K, \pi))$
  while not converged **do**
    for $n \in 1, 2, \ldots N$ **do** ▷ Local parameter update step
      for $k \in 1, 2, \ldots K$ **do**
        $W_{nk} = \log \pi_k - D(s(x_n), \mu_k)$ ▷ Log posterior weight of cluster k
      $z_n = \arg\max_{k \in \{1,2,\ldots K\}} W_{nk}$
    for $k \in 1, 2, \ldots K$ **do** ▷ Summary step
      $S_k(x, z) = \sum_{n=1}^N \delta_k(z_n) s(x_n)$
      $N_k(z) = \sum_{n=1}^N \delta_k(z_n)$
    for $k \in 1, 2, \ldots K$ **do** ▷ Global parameter update step
      $\mu_k = \frac{S_k(x,z) + \bar{\tau}}{N_k(z) + \bar{\nu}}$
      $\pi_k = \frac{N_k(z) + \frac{\gamma}{K}}{N + \gamma}$
  return z, μ

Block-coordinate ascent point estimation algorithm for mixture models. Sometimes called hard-assignment expectation-maximization (EM).

2.4.1 Bregman k-means point estimation for finite mixture model

Our goal is finding point estimates of φ, ϕ, z. We set up a maximum a-posteriori optimization problem:

$$\begin{aligned} \max_{\pi,\mu,z} \quad & \mathcal{J}(\pi, \mu, z, x) \\ \text{subject to } & \pi \in \Delta_K \\ & \mu_k \in \mathcal{M}, && \text{for } k = 1, 2, \dots K \\ & z_n \in \{1, 2, \dots K\}, && \text{for } n = 1, 2, \dots N \end{aligned} \tag{2.73}$$

where the objective function we wish to maximize is the joint log probability density over all the variables of interest:

$$\mathcal{J}(x, \pi, \mu, z) = \log p(\varphi(\pi)) + \sum_{k=1}^{K} \log p(\phi_k(\mu)) + \sum_{k=1}^{K} \sum_{n=1}^{K} \delta_k(z_n) \Big[\log \pi_k + \log p(x_n | \phi_k(\mu)) \Big]. \tag{2.74}$$

Crucially, we emphasize that the probability density functions used here are over the natural-form random variables φ and ϕ. However, the variables we instantiate throughout the algorithm are the corresponding mean parameters π (for frequency vectors) and μ (for cluster shapes).

An iterative optimization algorithm is given in Alg. 2.1, based on a similar algorithm from Banerjee et al. (2005). This algorithm operates in the mean-parameter space, but could equivalently be written in the natural-parameter space using appropriate transformation functionrs $\phi(\cdot) : \mathcal{M} \to \Phi$ from Eq. (2.32) and $\mu(\cdot) : \Phi \to \mathcal{M}$ in Eq. (2.31). We can interpret this algorithm as an extension of the popular k-means algorithm (Lloyd, 1982; Jain, 2010) to proper mixture models with conjugate-exponential-family observation models. Sensibly, each iteration of this method has cost that is *linear* in the desired number of clusters K and the total number of atoms to cluster N.

Local step: point estimates of assignments z.

Given fixed means $\{\mu_k\}_{k=1}^K$ for each cluster as well as the frequency vector π, we wish to maximize our objective $\mathcal{J}(x, \pi, \mu, z)$ with respect to z. Rewriting the objective as a function of z_n, we have:

$$\begin{aligned} \mathcal{J}_n(x_n, z_n, \mu, \pi) &= \sum_{k=1}^{K} \delta_k(z_n) W_{nk}(\mu, \pi) \\ W_{nk}(\mu, \pi) &\triangleq \log p(x_n | \phi(\mu_k)) + \log \pi_k \\ &= -D(s(x_n), \mu_k) + \log \pi_k \\ &= -c(\phi(\mu_k)) + s(x_n)^T \phi(\mu_k) + \log \pi_k \end{aligned} \tag{2.75}$$

Here, we can interpret W_{nk} as the log posterior weight for assigning cluster k to data atom n. Larger values indicate that cluster k provides a better explanation for atom n.

Note the last definition of W_{nk}, in terms of the cumulant $c(\phi_k)$ and the natural parameter, uses the Bregman divergence definition from Eq. (2.36), simplified by dropping terms that are constant with respect to cluster index k.

Applying Lagrange multiplier methods to find the optimal assignment while obeying the hard constraint $z_n \in \{1, 2, \dots K\}$ leads to the update

$$z_n^* = \text{argmax}_{k \in 1,2,\dots K} W_{nk} \tag{2.76}$$

We can interpret this simply as choosing the cluster label $k \in \{1, 2, \dots K\}$ which has maximum posterior probability under the current cluster means μ and frequencies π. This update is guaranteed to monotonically increase the objective $\mathcal{J}$.

Global step: MAP point estimates of frequency vector π.

Applying the standard arguments for natural-parameter maximum-a-posteriori estimation from Sec. 2.3.3, we have:

$$\pi_k^* = \frac{N_k + \frac{\gamma}{K}}{N + \gamma}, \qquad \text{where } \pi^* = \arg\max_{\pi \in \Delta_K} \log p(\varphi(\pi)) + \log p(z|\pi) \tag{2.77}$$

It is important to realize that this natural-parameter MAP estimate exists for *any* value of $\gamma \geq 0$.

In contrast, the equivalent mean-parameter optimization problem, which treats $\pi \in \Delta_K$ directly as a random variable, does *not* always have a MAP solution. In particular, the MAP estimate for this parameterization is:

$$\pi_k^* = \frac{N_k + \frac{\gamma}{K} - 1}{N + \gamma - K}, \qquad \pi^* = \arg\max_{\pi \in \Delta_K} \log p(\pi) + \log p(z|\pi) \tag{2.78}$$

This solution only exists when $N_k + \gamma/K > 1$. Otherwise, the MAP is undefined.

2.4.2 Bregman k-means++ initialization of point estimates for global parameters

The point estimation algorithm in Alg. 2.1 requires as input both initial cluster means $\{\mu_k\}_{k=1}^K$, as well as an initial cluster frequency vector π. Because the overall objective $\mathcal{J}$ is non-convex, selecting a smart initialization is important to avoid bad local optima and reach a high-quality solution.

Our inspiration for a smart initialization comes from the *k-means++* algorithm introduced by Arthur and Vassilvitskii (2007), which considers the simpler k-means optimization problem using Euclidean distances instead of general purpose Bregman divergences. The *k-means++* algorithm first selects K distinct data atoms from the dataset x, and then uses the standard update for cluster mean parameters given a single observation to initialize μ_k for each of the K clusters. The selection of the K chosen data atoms is done in a sequential, distance-biased fashion. The first cluster μ_1 is formed from a single data atom chosen uniformly at random. Then, the atom for successive cluster k is chosen from the remaining atoms with probability proportional to the Euclidean distance between the atom and the closest previously-chosen mean μ_k. This distance-biased random sampling leads to choosing cluster means μ_k which are far from one another and, if K is large enough, will with high-probability adequately cover the space of all observed data x. Formally, Arthur and Vassilvitskii

Algorithm 2.2 Bregman k-means++ initialization for cluster mean point estimates.

Input:

$\{x_n\}_{n=1}^N$: dataset with N exchangeable observations

$\bar{\tau}, \bar{\nu}$: hyperparameters defining the prior $\mathrm{P}(\phi_k|\bar{\tau}, \bar{\nu})$

Output:

μ : Initial point estimates of the mean parameters

function InitClusterMeansViaBregmanSamples($(x, \bar{\tau}, \bar{\nu})$)
  $n_1 \sim$ UniformRandomSample($\{1, 2, \dots N\}$)
  $\mu_1 \leftarrow \frac{s(x_{n_1})+\bar{\tau}}{1+\bar{\nu}}$
  for $k \in 2, \dots K$ **do**
    for $n \in 1, 2, \dots N$ **do**
      $p_n = \arg\min_{\ell \in \{1,2,\dots k-1\}} D(\mu(s(x_n)), \mu_\ell)$
    $n_k \sim \mathrm{Cat}_N(p_1, \dots p_N)$
    $\mu_k \leftarrow \frac{s(x_{n_k})+\bar{\tau}}{1+\bar{\nu}}$
  return $\{\mu_k\}_{k=1}^K$

Initialization of cluster means μ_k using a random sampling scheme based on Bregman divergences. Under some mild assumptions, Ackermann and Blömer (2010) show that this procedure will deliver a solution with guaranteed multiplicative approximation ratio to the optimal objective score.

(2007) proved that the *k-means++* procedure delivers an initialization μ that is already within a multiplicative approximation factor of the optimal k-means cost function.

Ackermann and Blömer (2010) suggest extending the *k-means++* initialization from Euclidean distances to all Bregman divergences and thus all possible observation models in our probabilistic clustering framework. Ackermann and Blömer (2010)'s procedure offers the same appealing guarantees: for some Bregman divergences, the cost of the initial clustering is guaranteed to be within some constant factor of the optimal cost.

Our initialization algorithm is given in Alg. 2.2. This sequential sampling procedure selects a single data atom to represent each cluster in a distance-biased way. At each round, we sample each remaining atom with probability proportional to a divergence whose first argument is the smoothed-mean estimate for the given atom, and the second is that data atom's closest center. By using smoothing when selecting data, we are able to both compute distance values exactly (without taking limits) and be sure we are choosing mean vectors that are far apart, not just data atoms. By choosing each successive cluster mean to be far from the previously-chosen means, we hope to eventually represent all latent clusters in the dataset. Once K means are chosen, we proceed with Alg. 2.1 until either convergence occurs or a maximum budget of iterations is reached.

2.5 Posterior inference via variational optimization.

The point estimation procedure in Alg. 2.1 is simpler to understand and implement than more sophisticated methods. However, the maximum likelihood or maximum-a-posteriori approach has one key weakness when training probablistic clustering models: the maximum-likelihood pased objective function $\mathcal{J}(x, z, \pi, \phi)$ cannot be used for model selection. That is, we cannot select the right number

of clusters K using this objective function. This is because of the inherent over-fitting property of maximum-likelihood based approaches: for every solution with exactly K clusters, there exists a solution with $K+1$ clusters with a larger objective function score.

Beal (2003) covers this model selection issue extensively in his published dissertation, and motivates a more sophisticated approach: variational optimization of an approximate posterior distribution. Rather than using maximum likelihood approaches which condition on random variables like π or ϕ which grow in size with K, these variational approaches use an objective based on the marginal likelihood $\log p(x)$, thus integrating away all random variables which scale with K. Using the marginal likelihood is an effective, Bayesian way to do model selection (Jefferys and Berger, 1992; Rasmussen and Ghahramani., 2001).

In this section, we will walk through the defining the optimization problem, work out the closed-form block-coordinate ascent updates, and give an algorithm for applying this technique to the same finite mixture model we pursued earlier, to aid side-by-side comparison of algorithmic complexity. While perhaps more complex to understand, we will show that approach has the same order-of-magnitude runtime cost as the point-estimation algorithm. Thus, we gain the benefit of improved model selection without substantial additional computational cost.

2.5.1 Mean-field approximate posterior density estimation

Given a dataset x, our goal is to estimate a joint posterior $p(\phi, \pi, z \mid x)$. over three variables: the global cluster appearance probabilities π, the global shape parameters ϕ and the local assignment variables z.

Finding this posterior directly is intractable. Suppose the assignments z use all K distinct cluster labels. Then even representing the marginal posterior of the assignment variables $p(z|x)$ would require memory and time that scales with $O(K^N)$, since we would need to enumerate the probability of each possible joint configuration of the variables $\{z_n\}_{n=1}^N$ and there are K^N possible configurations.

Instead, variational inference casts approximate posterior inference as an optimization problem (Wainwright and Jordan, 2008). This approach proceeds in three steps, which are detailed in the sections below. First, we define a simplified family of probability distributions for our variables of interest ϕ, π and z. This family of distributions is parameterized by several free variables which control their moments. Our goal is to find the values of these free variables. Next, we define an optimization problem whose objective is to find the free variables under which the approximate posterior is as close as possible to the true, intractable posterior, while still remaining in the specified simpler family. Formally, we define this "closeness" via Kullback-Liebler (KL) divergence, which is the natural measure of distance between two probability distributions. Third, we define an algorithm which, given input data x, delivers the free parameters that solve our optimization problem.

Let $q(\phi, \pi, z)$ denote our approximate posterior. As prevously discussed, without any simplifying assumptions even enumerating all possible posterior values is infeasible for all but the smallest datasets. To make our optimization problem computationally feasible, we assume that each of the

assignments z_n is independent of the others and each of the clusters ϕ_k is independent of the others. That is, we can factorize the approximate density as

$$q(\phi, \pi, z) = \prod_{k=1}^{K} q(\phi_k) \cdot q(\pi) \cdot \prod_{n=1}^{N} q(z_n) \tag{2.79}$$

This independence is sometimes called a *mean-field* assumption, after methods from statistical physics (Parisi, 1988). We further assume each factor of the approximate posterior has a density that belongs to the exponential family, whose form naturally mimics the generative model when appropriate. Thus, the full parameterization is:

$$q(\phi) = \prod_{k=1}^{K} \mathrm{P}(\phi_k | \hat{\tau}_k, \hat{\nu}_k) \tag{2.80}$$

$$q(\pi) = \mathrm{Dir}_K(\pi | \hat{\theta}_1, \hat{\theta}_2, \dots \hat{\theta}_K) \tag{2.81}$$

$$q(z) = \prod_{n=1}^{N} \mathrm{Cat}_K(z_n | \hat{r}_{n1}, \dots \hat{r}_{nK}) \tag{2.82}$$

Under this chosen factorization, the variables $\hat{\nu}, \hat{\tau}, \hat{\theta}, \hat{r}$ are the free parameters of our approximate posterior. The goal of variational optimization is to find specific values of these free parameters that make $q(u, \phi, z)$ a good approximation to the true posterior. We always denote free parameters with hats to make clear which variables are instantiated and optimized by the algorithm.

Approximate posterior $q(\pi)$ for global cluster frequencies

The vector $\pi \in \Delta_K$ has an approximate posterior we assume has Dirichlet form, just like its prior. The free parameter vector $\hat{\theta}$ has K entries that are all non-negative: $\hat{\theta}_k \geq 0$. Under our assumed posterior $q(\pi)$, we have the useful expectations:

$$\mathbb{E}_q[\pi_k] = \frac{\hat{\theta}_k}{\sum_{\ell=1}^{K} \hat{\theta}_\ell} \qquad \mathbb{E}_q[\log \pi_k] = \psi(\hat{\theta}_k) - \psi(\textstyle\sum_{\ell=1}^{K} \hat{\theta}_\ell) \tag{2.83}$$

where $\psi(\cdot)$ is the digamma function, defined as the first derivative of the log gamma function.

Approximate posterior $q(\phi_k)$ for global cluster shape

Each cluster k is given an independent posterior factor $q(\phi_k)$ for its shape parameter ϕ. Following the analysis for the exact posterior in Sec. 2.1.3, we assume this factor comes from the conjugate prior family P. The factor has two free parameters: pseudo-count $\hat{\nu}_k > 0$ and vector parameter $\hat{\tau}_k \in \mathcal{M}$ which defines the cluster shape.

Local assignment factor $q(z_n)$

Under the generative model for the finite mixture, the assigned cluster label z_n for observation n could use any of the K possible global clusters indices. Thus, for the posterior we need a categorical

distribution over K possible discrete choices. We define free parameter vector $\hat{r}_n$ to have K entries that are non-negative and sum to one. We can interpret $\hat{r}_{nk}$ as the posterior responsibility that cluster k has for data atom n. Values near zero indicate that cluster k does not explain the data well, while values closer to one indicate a good explanation. Under this approximate posterior, we have the useful expectation $\mathbb{E}_q[\delta_k(z_n)] = \hat{r}_{nk}$.

2.5.2 Evidence lower-bound objective function

Given the assumed factorization of $q(\pi, \phi, z)$ above, we now set up an optimization problem over our free parameters $\hat{\nu}, \hat{\tau}, \hat{\theta}, \hat{r}$. The goal is to minimize the distance between the approximate posterior $q(\pi, \phi, z)$ and the true posterior $p(\pi, \phi, z|x)$. We measure this distance via the Kullback-Liebler divergence, which we formally define below. We then explain the derivation of an optimization problem that minimizes the KL divergence between the approximate posterior $q(\pi, \phi, z)$ and the true posterior $p(\pi, \phi, z|x)$.

Kullback-Liebler (KL) divergence functions

Discrete distribution KL divergence. Consider a discrete random variable Y with K possible outcomes, each indexed by an integer in the set $\{1, 2, \dots K\}$. Let there be two possible distributions over the variable Y, one with parameter vector $P \in \Delta_K$ and the other with parameter $Q \in \Delta$. Under the first distribution, we have $p(Y = k) = P_k$, while under the second we have $p(Y = k) = Q_k$. Then, the KL divergence from vector Q to vector P is:

$$\mathrm{KL}\left(Q \| P\right) \triangleq \sum_{k=1}^{K} Q_k \log \frac{Q_k}{P_k}, \qquad Q \in \Delta_K, P \in \Delta_k, \tag{2.84}$$

For discrete distributions, $\mathrm{KL}\left(Q \| P\right) \geq 0$ always, with equality only if $Q_k = P_k$ for all outcomes $k \in \{1, 2, \dots K\}$. This divergence is assymmetric: $\mathrm{KL}\left(Q \| P\right) \neq \mathrm{KL}\left(P \| Q\right)$.

Continuous distribution KL divergence. For a continuous random variable such as ϕ_k, consider two alternative distributions $p(\phi_k)$ and $q(\phi_k)$. Assuming that each of these provide non-zero probability to all elements $\phi_k \in \Phi$, we define the KL divergence as:

$$\mathrm{KL}\left(q \| p\right) \triangleq \int_{\phi_k \in \Phi} q(\phi_k) \log \frac{q(\phi_k)}{p(\phi_k)} \tag{2.85}$$

For distributions over continuous random variables, the KL divergence is always non-negative: $\mathrm{KL}\left(q \| p\right) \geq 0$, with equality only if q and p represent the same probability density function.

KL divergence for our approximate posterior. The KL divergence between a specific member of our approximate posterior family $q(\pi, \phi, z)$ and the true posterior $p(\pi, \phi, z|x)$ is given by

$$\mathrm{KL}\left(q(\pi, \phi, z) \| p(\pi, \phi, z)\right) \triangleq \int_{\pi \in \Delta_K} \int_{\phi_1 \in \Phi} \dots \int_{\phi_K \in \Phi} \sum_{z_1=1}^{K} \dots \sum_{z_N=1}^{K} q(\pi, z, \phi) \log \frac{q(\pi, z, \phi)}{p(\pi, z, \phi|x)} d\phi_1 d\phi_K d\pi \tag{2.86}$$

Evalating the above function is not possible, because we do not have a closed-form expression for the true posterior density function $p(\pi, z, \phi|x)$. We can alternatively write the KL divergence as an expectation with respect to the approximate posterior $q(\pi, \phi, z)$:

$$\mathrm{KL}\left(q(\pi,\phi,z)\|p(\pi,\phi,z)\right) = \mathbb{E}_{q(\pi,\phi,z)}\left[\log \frac{q(\pi,\phi,z)}{p(\pi,\phi,z|x)}\right] \tag{2.87}$$

This does not make evaluation any easier, but certainly makes the mathematical notation easier to read.

Using Bayes rule, we can rewrite the (intractable) posterior as the joint probability of all random variables (including data x) divided by the marginal probability $p(x)$.

$$p(\pi, z, \phi|x) = \frac{p(\pi,\phi,z,x)}{p(x)} \tag{2.88}$$

Our generative model specifies the functional form of the numerator. So, it is only the denominator $p(x)$ that we cannot compute. Substituting into our KL divergence expression, we have:

$$\begin{aligned}\mathrm{KL}\left(q(\pi,\phi,z)\|p(\pi,\phi,z)\right) &= \mathbb{E}_{q(\pi,\phi,z)}\left[\log \frac{q(\pi,\phi,z)p(x)}{p(\pi,\phi,z,x)}\right] \\ &= \mathbb{E}_{q(\pi,\phi,z)}\left[\log \frac{q(\pi,\phi,z)}{p(\pi,\phi,z,x)}\right] + \log p(x)\end{aligned} \tag{2.89}$$

where in the second line we've used the fact that $p(x)$ is a function independent of pi, ϕ and z to move its term outside the expectation.

We can always compute the first additive term of this KL divergence between our approximate posterior and true posterior, while the second is a constant independent of the approximate density $q(\pi, \phi, z)$.

Minimizing KL divergence from approximate posterior to true posterior

Our training goal is to find the specific approximate posterior density $q(\pi, \phi, z)$ that is nearest (in the sense of KL divergence) to the true posterior.

$$\arg\min_{\hat{\nu},\hat{\tau},\hat{\theta},\hat{r}} \mathrm{KL}(q(\pi,\phi,z|\hat{\nu},\hat{\tau},\hat{\theta},\hat{r})||p(\pi,\phi,z|x)) \tag{2.90}$$

Using the decomposition of KL divergence from Eq. 2.89, we can derive an equivalent maximization problem:

$$\arg\max_{\hat{\nu},\hat{\tau},\hat{\theta},\hat{r}} \ \mathcal{L}(x,\hat{\nu},\hat{\tau},\hat{\theta},\hat{r}) \tag{2.91}$$

The objective function $\mathcal{L}$ is defined as the difference between the log marginal probability of the data $\log p(x)$ (which is a constant with respect to our free parameters) and the earlier KL divergence:

$$\begin{aligned}\mathcal{L}(x, \hat{\nu}, \hat{\tau}, \hat{\theta}, \hat{r}) &\triangleq \log p(x) - \mathrm{KL}(q(\pi, \phi, z|\hat{\nu}, \hat{\tau}, \hat{\theta}, \hat{r})||p(\pi, \phi, z|x)) \\ &= \log p(x) - \mathbb{E}_{q(\pi,\phi,z)}\Big[\log q(\pi, \phi, z) - \log p(x, \pi, \phi, z) + \log p(x)\Big] \\ &= \mathbb{E}_{q(\pi,\phi,z)}\Big[\log p(\pi, \phi, z, x) - \log q(\pi, \phi, z)\Big]\end{aligned} \tag{2.92}$$

Under our chosen exponential family forms for each factor of the approximate posterior, the expectations that define $\mathcal{L}$ lead to a closed-form function of the free parameters $\hat{\tau}, \nu, \hat{\theta}, \hat{r}$. We can tractably evaluate $\mathcal{L}$ and take derivatives, making optimization of the free parameters possible.

We emphasize that because the KL divergence term is always non-negative, the objective $\mathcal{L}$ can be interpreted as a strict lower bound on the marginal likelihood $\log p(x)$. We thus often refer to the function $\mathcal{L}$ as the Evidence Lower BOund, or ELBO. Maximizing $\mathcal{L}$ can be interpreted as improving the approximate posterior's explanation of the data.

We can write the objective $\mathcal{L}$ as a sum of two terms:

$$\mathcal{L}(x, \hat{r}, \hat{\theta}, \hat{\tau}, \hat{\nu}) = \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta}) \tag{2.93}$$

These terms describe distinctly interpretable pieces of the overall model: $\mathcal{L}_{\text{data}}$ gathers terms related to the observation model and $\mathcal{L}_{\text{alloc}}$ gathers terms related to the allocation model. Our chosen notation for each term of the objective highlights the variational parameters involved. For example, $\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta})$ is independent of the observation parameters $\hat{\nu}, \hat{\tau}$. These terms may also be functions of the data x and hyperparameters $\mathcal{H}$, but we omit these arguments in notation for simplicity.

This term-by-term breakdown of the objective encourages a modular implementation. Given fixed assignments $\hat{r}$, the solution to finding the optimal observation model parameters $\hat{\nu}, \hat{\tau}$ must be independent of the allocation probability parameters $\hat{\theta}$, and vice versa. This modularization enables our implementation to implement free parameter updates once for each possible observation model or allocation model, and compose these modules to create an overall model.

2.5.3 Observation model term of the objective

The data term is defined by

$$\begin{aligned}\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \nu) &\triangleq \mathbb{E}_q\Big[\log p(x|z, \phi) + \log \frac{p(\phi|\bar{\nu}, \bar{\tau})}{q(\phi|\hat{\nu}, \hat{\tau})}\Big] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \hat{r}_{nk}\mathbb{E}_{q(\phi_k|\hat{\nu}_k,\hat{\tau}_k)}\Big[\log p(x_n|\phi_k)\Big] + \sum_{k=1}^{K}\mathbb{E}_{q(\phi_k|\hat{\nu}_k,\hat{\tau}_k)}\Big[\log \frac{p(\phi_k|\bar{\nu}, \bar{\tau})}{q(\phi_k|\hat{\nu}_k, \hat{\tau}_k)}\Big]\end{aligned} \tag{2.94}$$

Where we have already substituted in the soft assignment free parameters: $\hat{r}_{nk} = \mathbb{E}_{q(z_n|\hat{r}_n)}[\delta_k(z_n)]$.

We can further simplify this objective by using the derivations from Sec. 2.1.3. We have

$$\begin{aligned}\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) = &\sum_{n=1}^{N} h(x_n) + \sum_{k=1}^{K} \Big(c^P(\bar{\tau}, \bar{\nu}) - c^P(\hat{\tau}_k, \hat{\nu}_k) \Big) \\ &+ \sum_{k=1}^{K} \Big(N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k \Big) \mathbb{E}_{q(\phi_k)} \Big[- c^L(\phi_k) \Big] \\ &+ \sum_{k=1}^{K} \sum_{d=1}^{D} \Big(S_{kd}(x, \hat{r}) + \bar{\tau}_d - \hat{\nu}_{kd} \Big) \mathbb{E}_{q(\phi_k)} \Big[\phi_{kd} \Big]\end{aligned} \tag{2.95}$$

Here, we define $S_k(x, \hat{r})$ as the expected value of the earlier sufficient statistic $S_k(x, z)$ from Eq. (2.53) under $q(z)$. Similarly, $N_k(\hat{r})$ is the expected value of $N_k(z)$ from Eq. (2.53).

$$S_k(x, \hat{r}) \triangleq \sum_{n=1}^{N} \mathbb{E}_{q(z_n|\hat{r}_n)}[\delta_k(z_n) s(x_n)] = \sum_{n=1}^{N} \hat{r}_{nk} s(x_n) \tag{2.96}$$

$$N_k(\hat{r}) \triangleq \sum_{n=1}^{N} \mathbb{E}_{q(z_n|\hat{r}_n)}[\delta_k(z_n)] = \sum_{n=1}^{N} \hat{r}_{nk} \tag{2.97}$$

The remaining expectations in Eq. (2.95) are the fundamental ones of the conjugate exponential family likelihood-prior system chosen, which have closed form.

2.5.4 Allocation model term of the objective

We write the allocation model's contribution to the objective as:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta}) \triangleq \mathbb{E}_{q(z|\hat{r})q(\pi|\hat{\theta})} \Big[\log \frac{p(z|\pi)}{q(z|\hat{r})} + \log \frac{p(\pi|\gamma)}{q(\pi|\hat{\theta})} \Big] \tag{2.98}$$

Regrouping terms, we can separate this into an entropy term for the distribution $q(z|\hat{r})$ and a term which gathers all functions of $\hat{\theta}$:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta}) = \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{Dir-alloc}}(\hat{r}, \hat{\theta}) \tag{2.99}$$

Entropy term

The entropy of the assignments is a simple non-linear function of the responsibilities:

$$\mathcal{L}_{\text{entropy}}(\hat{r}) = -\sum_{n=1}^{N} \mathbb{E}_q[\log q(z_n)] = -\sum_{n=1}^{N} \hat{r}_{nk} \log \hat{r}_{nk}. \tag{2.100}$$

Dirichlet allocation term

Standard conjugate exponential family mathematics yields a simple expression for $\mathcal{L}_{\text{Dir-alloc}}$:

$$\mathcal{L}_{\text{Dir-alloc}}(\hat{r}, \hat{\theta}) = \mathbb{E}_q[\log p(z|\pi) + \log \frac{p(\pi)}{q(\pi)}] \tag{2.101}$$

$$= \sum_{k=1}^{K} c_{\text{Dir}}(\frac{\gamma}{K} \dots \frac{\gamma}{K}) - c_{\text{Dir}}(\hat{\theta}) \tag{2.102}$$

$$+ \sum_{k=1}^{K} \Big(N_k(\hat{r}) + \frac{\gamma}{K} - \hat{\theta}_k \Big) \mathbb{E}_q[\log \pi_k]$$

Here, the expectations have closed form under the assumed distribution $q(\pi|\hat{\theta})$, and the cumulant function of the Dirichlet distribution is

$$c_{\text{Dir}}(a_1, a_2, \dots a_K) \triangleq \log \Gamma(\sum_{\ell} a_\ell) - \sum_{k=1}^{K} \log \Gamma(a_k) \tag{2.103}$$

Finally, the count statistic $N_k(\hat{r})$ for each cluster k is defined above in Eq. (2.96). We can interpret N_k as the effective count of data observations assigned to cluster label k.

2.6 Variational inference algorithm for the finite mixture model

We can now formally define the constrained optimization problem for our free parameters $\hat{\tau}, \hat{\nu}, \hat{\theta}, \hat{r}$ given an observed dataset x and hyperparameters $\mathcal{H}$:

$$\arg\max_{\hat{\tau}, \hat{\nu}, \hat{\theta}, \hat{r}} \quad \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{Dir-alloc}}(\hat{r}, \hat{\theta}) \tag{2.104}$$

$$\begin{aligned}
\text{subject to } \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^{K} \hat{r}_{nk} = 1 \quad & \text{for } n = 1, 2, \dots N \\
\hat{\theta}_k \geq 0 \quad & \text{for } k = 1, 2, \dots K \\
\hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \quad & \text{for } k = 1, 2, \dots K
\end{aligned}$$

Given the internal structure of the objective, we pursue a block-coordinate ascent algorithm, which proceeds in three steps. Each step updates the free parameters of one factor among $q(z)q(u)q(\phi)$ while holding the other free parameters fixed. When applied iteratively, these steps are guaranteed to monotonically improve the whole objective $\mathcal{L}$ until it converges to a fixed point local optima.

Below, we define the optimization problem solved by each step of the block-coordinate ascent algorithm. We then describe detailed solutions which solve each problem below in closed-form in the following sections.

Local step: Update local assignment responsibilities

$$\arg\max_{\hat{r}} \quad \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{Dir-alloc}}(\hat{r}, \hat{\theta}) \tag{2.105}$$

$$\text{subject to } \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^{K} \hat{r}_{nk} = 1 \qquad \text{for } n = 1, 2, \ldots N$$

Global step: Update observation model global parameters

$$\arg\max_{\hat{\tau}, \hat{\nu}} \quad \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) \qquad \text{subject to } \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \text{ for } k = 1, 2, \ldots \tag{2.106}$$

Global step: Update allocation model global parameters

$$\arg\max_{\hat{\theta}} \quad \mathcal{L}_{\text{Dir-alloc}}(\hat{r}, \hat{\theta}) \qquad \text{subject to } \hat{\theta}_k \geq 0 \text{ for } k = 1, 2, \ldots K \tag{2.107}$$

2.6.1 Local parameter update step

To solve the local step optimization problem in Eq. (2.105), we first simplify the whole objective $\mathcal{L}$ by showing it as a function of the responsibilities at each observation n. We gather those terms that do not depend on the active responsibilities $\hat{r}$ for clusters $k \in \{1, \ldots K\}$ together into a constant term:

$$\mathcal{L}(x, \hat{r}, \hat{\eta}, \hat{\tau}, \hat{\theta}) = \text{const}(x, \hat{\eta}, \hat{\tau}, \hat{\nu}) + \sum_{n=1}^{N} \mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\theta}) \tag{2.108}$$

Now, the objective at observation n is given by

$$\mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \triangleq \sum_{k=1}^{K} \hat{r}_{nk} \left(W_{nk}(x_n, \hat{\theta}, \hat{\tau}, \hat{\nu}) - \log \hat{r}_{nk} \right) \tag{2.109}$$

$$W_{nk}(x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \triangleq \mathbb{E}_{q(\phi_k)} \Big[\log p(x_n | \phi_k) \Big] + \mathbb{E}_{q(\pi|\hat{\theta})} \Big[\log \pi_k \Big] \tag{2.110}$$

Here, we can interpret each W_{nk} as the log posterior weight that cluster k has on observation n. Larger values indicate that cluster k is more likely to explain observation n. The expectations that define W_{nk} have closed form under our chosen approximate posterior family $q(\pi)q(\phi)$.

From the decomposition of $\mathcal{L}$ into a sum of independent terms for each observation, it is clear that our objective may be optimized independently for each observation n.

$$\hat{r}_n^* = \arg\max_{\hat{r}_n} \quad \mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \tag{2.111}$$

$$\text{subject to } \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^{K} \hat{r}_{nk} = 1$$

Through standard constrained optimization methods, we find the solution for the optimal responsibility vector $\hat{r}_n^*$. We can compute the optimal vector in closed-form by setting each entry $k \in \{1, 2, \ldots K\}$ to the exponentiated posterior weight $e^{W_{nk}}$ and then normalizing:

$$\hat{r}_{nk}^* = \frac{e^{W_{nk}}}{\sum_{\ell=1}^{K} e^{W_{n\ell}}}, \tag{2.112}$$

Algorithm 2.3 Update for responsibilities given log posterior weights for mixture model.

Input: $[W_{n1}\ W_{n2}\ \dots W_{nK}]$: log posterior weights.
Output: $[\hat{r}_{n1} \dots \hat{r}_{nK}]$: responsibility values for each cluster

```
function RespFromWeights(W_n)
    for k ∈ 1, ... K do
        r̂_nk = e^{W_nk}
    s_n = Σ_{k=1}^{K} r̂_nk
    for k ∈ 1, ... K do
        r̂_nk = r̂_nk / s_n
    return r̂_n
```

RespFromWeights delivers posterior probabilities of assigning data observation at index n to each cluster k in the set $\{1, 2, \dots K\}$. See Sec. 2.6.1 for details.

which by inspection is guaranteed to obey the required constraints that responsibilities must be non-negative and sum to one. By performing the update in Eq. (2.112) at each observation n independently, we will compute the optimal responsibilities $\hat{r}$ for the whole dataset.

2.6.2 Global parameter update step

Global step for observation model

As discussed in Sec. 2.1.3, the optimal observation global parameters $\hat{\tau}_k^*, \hat{\nu}_k^*$ for every cluster $k \in \{1, 2, \dots K, \dots\}$ which solve the optimization problem in Eq. (2.106) can be found in closed form:

$$\begin{aligned} \hat{\tau}_k^* &= S_k(x, \hat{r}) + \bar{\tau} \\ \hat{\nu}_k^* &= N_k(\hat{r}) + \bar{\nu} \end{aligned} \tag{2.113}$$

These optimal values naturally satisfy the required constraints $\hat{\nu}_k^* \in \mathbb{R}^+$ and $\hat{\tau}_k^* \in \mathcal{M}$, so that $\hat{\nu}_k^*$ and $\hat{\tau}_k^*$ remain valid parameters for the density $q(\phi_k)$.

Simplified $\mathcal{L}_{\text{data}}$ objective term. Substituting the optimal points $\hat{\tau}^*, \hat{\nu}^*$ into the original objective $\mathcal{L}_{\text{data}}$ in Eq. (2.95) , we see that the last two lines involving the terms $N_k + \bar{\nu} - \hat{\nu}_k$ and $S_k + \bar{\tau} - \hat{\tau}_k$ will evaluate two zero, leaving a greatly simplified expression:

$$\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}^*, \hat{\nu}^*) = \sum_{n=1}^{N} h(x_n) + \sum_{k=1}^{K} \Big(c^{\text{P}}(\bar{\tau}, \bar{\nu}) - c^{\text{P}}(\hat{\tau}_k^*, \hat{\nu}_k^*) \Big) \tag{2.114}$$

The reference measure term remains unchanged, while the sum over the cumulant functions is over the K active clusters.

Global step for allocation model

To find the optimal values $\hat{\theta}$ for the optimization problem in Eq. (2.107), we can apply standard constrained optimization techniques like Lagrange multipliers to find the closed-form solution for

Algorithm 2.4 Variational coordinate ascent for finite mixture model

Input:

$\{x_n\}_{n=1}^N$: dataset with N exchangeable observations
$\bar{\tau}, \bar{\nu}$: hyperparameters defining the prior $\mathrm{P}(\phi_k|\bar{\tau}, \bar{\nu})$
$\hat{\theta}$: Initial global parameters of the allocation model
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: Initial global parameters of the observation model

Output:

$\hat{\theta}$: Final global parameter of allocation model
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: Final global parameters of the observation model

function VariationalCoordAscentForFiniteMixtureModel
  while not converged **do**
    for $n \in 1, 2, \ldots N$ **do** ▷ Local parameter update step
      for $k \in 1, 2, \ldots K$ **do**
        $C_{nk} \leftarrow \mathbb{E}_q[\log p(x_n|\phi_k)]$
        $W_{nk} \leftarrow C_{nk} + \mathbb{E}_q[\log \pi_k]$
      $\hat{r}_n = \textsc{RespFromWeights}(W_n)$

    for $k \in 1, 2, \ldots K$ **do** ▷ Summary step
      $S_k(x, \hat{r}) = \sum_{n=1}^N \hat{r}_{nk} s(x_n)$
      $N_k(\hat{r}) = \sum_{n=1}^N \hat{r}_{nk}$

    for $k \in 1, 2, \ldots K$ **do** ▷ Global parameter update step
      $\hat{\tau}_k \leftarrow S_k(x, \hat{r}) + \bar{\tau}$
      $\hat{\nu}_k \leftarrow N_k(\hat{r}) + \bar{\nu}$
      $\hat{\theta}_k \leftarrow N_k(\hat{r}) + \frac{\gamma}{K}$

Block coordinate ascent algorithm for approximate posterior inference for the finite mixture model. We alternate between two updates: one for the local assignment parameters $\hat{r}$ given fixed global parameters, and another for the global allocation parameters $\hat{\theta}$ and the global observation parameters $\hat{\tau}, \hat{\nu}$. All update steps scale linearly with the number of observations N and clusters K.

each cluster k:

$$\hat{\theta}_k^* = N_k(\hat{r}) + \frac{\gamma}{K} \tag{2.115}$$

Naturally, these updates preserve the required constraints $\hat{\theta} \geq 0$.

2.6.3 Full-dataset optimization algorithm

Alg. 2.4 gives the overall algorithm for finding the optimal free parameters under our optimization objective function $\mathcal{L}$ for a given dataset x. Given some initial global parameters, it iteratively cycles between the local step for $q(z_n)$ in Eq. (2.111) and the global parameter updates. The global step is then decomposed into an update for the observation model in Eq. (2.113) and the allocation model in Eq. (2.115), which can be done independently once the required sufficient statistics are computed.

2.7 Discussion

We emphasize that Alg. 2.1 and Alg. 2.4 are two possible approaches to training a finite mixture model with K possible clusters from observed data x. The first uses a MAP point estimation objective:

$$\mathcal{J}(z, \pi, \mu) = \log p(\varphi(\pi)) + \sum_{k=1}^{K} \log p(\phi_k(\mu)) + \sum_{k=1}^{K} \sum_{n=1}^{N} \delta_k(z_n) \Big[\log \pi_k + \log p(x_n | \phi_k(\mu)) \Big] \tag{2.116}$$

while the variational approach optimizes a lower bound on the *marginal likelihood*:

$$\mathcal{L}(\hat{r}, \hat{\theta}, \hat{\tau}, \hat{\nu}) \leq \log p(x | \bar{\nu}, \bar{\tau}, \gamma) \tag{2.117}$$

By inspection, we see that the steps of the full variational approach in Alg. 2.4 have almost the same runtime complexity as the point estimation algorithm. Certainly, the global step in both algorithms has exactly the same complexity given the sufficient statistics. Likewise, the computation of the weight matrix W in each algorithm has the same cost. The key difference lies in representing the point estimated integer z_n vs. the approximate posterior responsibility vector $\hat{r}_n$. The variational approach has the additional step of performing the RespFromWeights update, which requires appreciable more work than simply taking the cluster index with maximum weight. However, the modest additional cost of the full variational approach yields significant gains in model selection.

Chapter 3

Scalable variational inference for Dirichlet process mixture models

We now consider the simplest Bayesian nonparametric model for clustering: the Dirichlet process mixture model. Recall the finite mixture model with a fixed set of K possible clusters from Ch. 2. One formal interpretation of the DP mixture model is the limit of the finite mixture model as the number of clusters $K \to \infty$. By providing an infinite number of possible clusters a-priori, the DP mixture model provides a flexible way to avoid an overly restrictive fixed number of clusters and easily transition from a few hundred examples to a few million examples, because the model capacity adjusts automatically.

Both MCMC samplers (Neal, 1992; Rasmussen, 1999; Walker, 2007) and early variational optimization algorithms (Blei and Jordan, 2006; Kurihara et al., 2006) are well-known for training DP mixture models from datasets. This chapter describes and extends our earlier work described in an NIPS 2013 conference paper co-authored with Erik Sudderth (Hughes and Sudderth, 2013), whose fundamental contributions are:

Contribution 1: Improved approximations to the posterior via nested truncation. While there are an infinite number of clusters available to this model under both its prior and posterior, only a finite number of clusters can be practically represented in a computer. Previous variational approaches (Blei and Jordan, 2006; Kurihara et al., 2006) had chosen approximate posterior distributions $q(z)q(\phi)q(\pi^G)$ with stringent truncation assumptions that lead to modeling artifacts. Instead, we take inspiration from Teh et al. (2008) to consider a truncation which is more elegant yet equally fast to train.

Contribution 2: New memoized inference algorithm for scalable training from millions of examples. Inspired by the incremental expectation-maximization (EM) algorithm from

Neal and Hinton (1998), we develop our own incremental algorithm for training approximate posterior representations rather than point estimates. Using *memoized* or cached sufficient statistics, we can process small subsets of data before each global parameter update step yet maintain. Unlike stochastic algorithms, our memoized algorithm avoids nuisance learning rates entirely and formally guarantees the ELBO objective function will monotonically increase after every step.

Roadmap. We first formally define the stick-breaking transformation, which we then use to define a generative process for the DP mixture model with independently-sampled global random variables. With the generative model in place, we develop a variational optimization problem with standard mean-field assumptions and our chosen nested truncation for approximate posteriors for each assigned cluster label $q(z_n)$. Closely following the variational algorithm for finite mixture models from Ch. 2, we develop the objective function and the relevent update steps for global and local variables. We then develop two scalable alternatives: stochastic variational and memoized variational.

In this chapter, we focus on fixed-truncation algorithms and leave description of adaptive proposals for adding and removing clusters to Ch. 4.

3.1 Stick-breaking construction of Dirichlet Process

The *Dirichlet process* defines a distribution over random probability measures, or equivalently functions over a predefined event space that are non-negative and sum to one. Here we provide a formal definition of this stochastic process as well as a generative construction.

3.1.1 Dirichlet processes

Suppose that Φ is a measurable event space, either real or discrete. For example, Φ could be the set of all positive integers or the set of all real numbers. We first define a properly-normalized *base measure* P over this space. For example, if Φ is the real line, then the density P could be a univariate Gaussian distribution. Formally, we require that if Φ is a discrete space, we have $\sum_{\phi \in \Phi} \mathrm{P}(\phi) = 1$, while if continuous we have $\int_{\phi \in \Phi} \mathrm{P}(\phi) d\phi = 1$.

Now, consider another measure R over the event space Φ. We say that R is a realization of a Dirichlet process with base measure P, if for *any* complete partition of Φ into K disjoint pieces labelled $T_1, T_2, \ldots T_K$ we have:

$$[R(T_1)\ R(T_2)\ \ldots R(T_K)] \sim \mathrm{Dir}_K(\gamma \mathrm{P}(T_1), \gamma \mathrm{P}(T_2), \ldots \gamma \mathrm{P}(T_K)) \tag{3.1}$$

Here, the scalar $\gamma > 0$ is the *concentration* parameter of the Dirichlet process. We write this realization as $R \sim \mathrm{DP}(\gamma, \mathrm{P})$. For further formal details of this definition, see (Sudderth, 2006, Theorem 2.5.1).

For example, consider the event space of all countable integers $\Phi = \{1, 2, 3, 4, \ldots\}$. Let the base measure P be an infinite vector of probabilities that sum to one: $\{\mathrm{P}_k\}_{k=1}^{\infty}$. The realization R would also correspond to an infinite vector of probabilities $\{R_k\}_{k=1}^{\infty}$. Suppose we partitioned the event

space Φ into the first two clusters $T_1 = \{1\}, T_2 = \{2\}$, and all remaining clusters $T_{>2} = \{3, 4, \ldots\}$. Then, we would consider R a realization of a Dirichlet process if:

$$[R_1,\ R_2,\ R_{>2}] \sim \mathrm{Dir}_3(\gamma \mathrm{P}_1, \gamma \mathrm{P}_2, \gamma \mathrm{P}_{>2}) \tag{3.2}$$

where the subscript $_{>2}$ is defined as the infinite remaining sum of all terms beyond index 2: $R_{>2} = \sum_{k=3}^{\infty} R_k$.

This definition is *not* constructive, because it does not give us a formula for generating sample realizations R. We develop such a construction in the next section.

3.1.2 Stick-breaking transformation

Recall that under the finite mixture model from Ch. 2, we draw the K-dimensional frequency vector from a Dirichlet prior: $\pi^G \sim \mathrm{Dir}_K(\frac{\gamma}{K} \ldots \frac{\gamma}{K})$. Taking the infinite limit of this density as $K \to +\infty$ is not directly tractable. We cannot generate samples from the infinite limit the same way we would usually draw finite samples. Instead, we develop an alternative construction which lets us build an infinite-dimensional frequency vector π^G from a series of independent, identically-distributed random variables $\{u_k\}_{k=1}^{\infty}$. Formally, there is a one-to-one invertible transformation between π^G and u called the *stick-breaking* transformation, which was developed by Sethuraman (1994).

Stick-breaking transformation Consider two infinite-dimensional vector spaces:

$$\begin{aligned} \Delta_\infty &\triangleq \{\pi \in \mathbb{R}^\infty : \sum_{k=1}^{\infty} \pi_k = 1, \quad \pi_k \geq 0\ \forall k\} \\ \mathcal{U}_\infty &\triangleq \{u \in \mathbb{R}^\infty : u_k \in [0, 1]\ \forall k\} \end{aligned} \tag{3.3}$$

First, the simplex Δ_∞ contains normalized vectors π of positive real values that sums to one. Second, the space $\mathcal{U}_\infty$ contains *unnormalized* vectors u of real values between zero and one.

There exists a deterministic, invertible, one-to-one transformation between each normalized vector $\pi \in \Delta_\infty$ and a corresponding vector $u \in U_\infty$, defined as:

$$\pi_k(u) = u_k \textstyle\prod_{\ell=1}^{k-1}(1-u_\ell) \qquad u_k(\pi) = \frac{\pi_k}{1-\sum_{\ell=1}^{k-1}\pi_k} \tag{3.4}$$

The transformation in Eq. (3.4) is often called the stick-breaking transformation. Because it is invertible, we can apply it to any probability density defined over the space $\mathcal{U}_\infty$ and obtain a corresponding density over Δ_∞

Sampling a realization via stick-breaking. Consider an infinite series of values $u = \{u_k\}_{k=1}^{\infty}$ consisting of independent draws from a Beta distribution: $u_k \sim \mathrm{Beta}(1, \gamma)$. Consider a corresponding infinite series of values $\phi = \{\phi_k\}_{k=1}^{\infty}$ drawn as independent samples from the base measure: $\phi_k \sim \mathrm{P}$. We can construct a realization of the Dirichlet process as:

$$R = \sum_{k=1}^{\infty} \pi_k^G(u)\delta(\phi_k), \qquad \pi_k^G(u) = u_k \prod_{\ell=1}^{k-1}(1 - u_\ell) \tag{3.5}$$

Here, the realization R is composed of infinitely-many point-masses ϕ_k, each with a frequency value π_k^G. The frequency at cluster k, denoted π_k^G is the stick-breaking transform of the independent conditional probabilities u into the simplex. This transformed vector π^G satisfies the definition of the Dirichlet process from Eq. (3.1) (Sudderth, 2006).

Stick-breaking construction is size-biased. Under the stick-breaking construction, the expected values of each entry of frequency vector π^G will decrease as the index k increases:

$$\begin{aligned}
&\mathbb{E}_{u_k\sim\text{Beta}(1,\gamma)}[\pi_1(u)] = \frac{1}{1+\gamma} \\
&\mathbb{E}_{u_k\sim\text{Beta}(1,\gamma)}[\pi_2(u)] = \frac{1}{1+\gamma}\cdot\frac{\gamma}{1+\gamma} \\
&\mathbb{E}_{u_k\sim\text{Beta}(1,\gamma)}[\pi_3(u)] = \frac{1}{1+\gamma}\cdot\frac{\gamma}{1+\gamma}\cdot\frac{\gamma}{1+\gamma} \\
\ldots\quad &\mathbb{E}_{u_k\sim\text{Beta}(1,\gamma)}[\pi_k(u)] = \frac{\gamma^{k-1}}{(1+\gamma)^k}
\end{aligned} \tag{3.6}$$

Thus, in general, realizations of the vector $\pi^G(u)$ will tend to have larger values for early indices than later ones. We call this property *size-biasedness*.

This property does not exist in the finite mixture model, where we draw π^G from a symmetric Dirichlet distribution: $\pi^G \sim \text{Dir}_K(\frac{\gamma}{K}\ldots\frac{\gamma}{K})$. To see the connection, we imagine drawing an infinite vector $p \in \Delta_\infty$ from the following limiting density

$$p \sim \lim_{K\to\infty} \text{Dir}_K([\frac{\gamma}{K}\ldots\frac{\gamma}{K}]) \tag{3.7}$$

After sorting the resulting vector p in descending order, we'll find that the distribution of the sorted vector p is equivalent to the distribution on π^G.

Alternative stochastic process priors. The Dirichlet process (DP) serves as the underlying stochastic process for all the probabilistic clustering models we consider. Extensions to richer class of Pitman-Yor (PY) process (Pitman and Picard, 2006) or the logistic stick-breaking process should be possible and at least for the PY mixture model should be straightforward, but we do not explore these here.

3.2 Dirichlet process mixture models

The Dirichlet process mixture model, often called a DP mixture model, was first introduced by Ferguson (1973). This widely-used Bayesian nonparametric model generates an exchangeable dataset $x = \{x_1, \ldots x_N\}$ with N total observations from a countably infinite set of clusters with labels in the set of integers $\{1, 2, 3, \ldots k, k+1\ldots\}$. Each cluster with label k has two parameters: the appearance probability π_k^G and the cluster shape parameter ϕ_k. These are the global parameters. Each observation n has one local unknown variable: its discrete assignment $z_n \in \{1, 2, \ldots\}$. A directed graphical model is shown in Fig. 3.1.

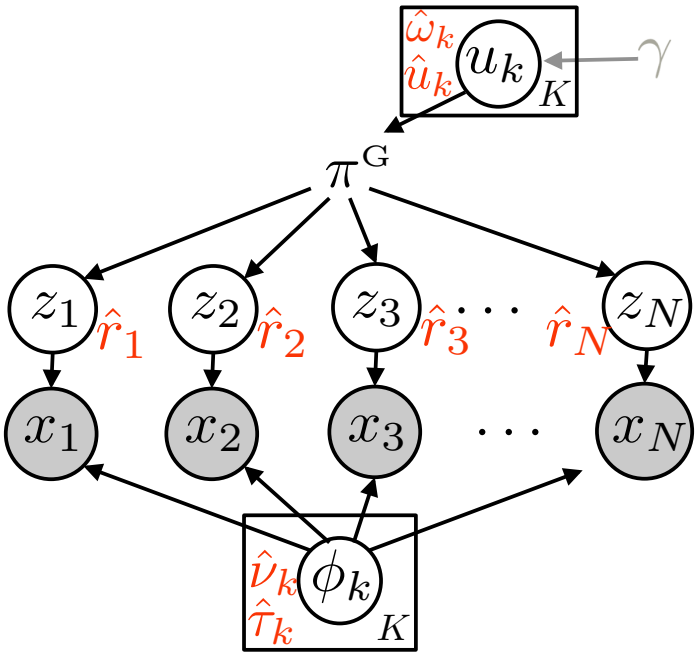

Figure 3.1: Directed graphical representation of the Dirichlet Process Mixture Model
We show the fundamental random variables (circled nodes), hyperparameters (gray), and variational free parameters (red) of the Dirichlet process (DP) mixture model, a Bayesian nonparametric model with $K \to \infty$ clusters. Each cluster is defined by two global parameters: conditional probability u_k and shape parameter ϕ_k. The conditional probabilities u are *deterministically* mapped to global cluster probabilities π^G via the invertible stick-breaking transformation. Each of the N observed data atoms x_n has an associated local cluster assignment z_n. The data is assumed to be exchangeable and conditionally independent given the global parameters. Under our chosen nested truncation for the approximate posterior, each $q(z_n)$ is defined by a responsibility vector $\hat{r}_n$ which places mass only on the first K cluster labels.

3.2.1 Generative model for global parameters

We use the earlier stick-breaking construction to generate each cluster k's two global parameters: the conditional probability u_k and the shape parameter ϕ_k.

Allocation model prior on global cluster frequencies. We generate the conditional probability $u_k \in (0, 1)$ independently from a Beta distribution: $u_k \sim \text{Beta}(1, \gamma)$. We interpret this as the probability of choosing cluster k among the infinite set of cluster labels with index at least k or larger: $\{k, k+1, \ldots\}$. From the series $\{u_\ell\}_{\ell=1}^k$, we can obtain the corresponding frequency value $\pi_k^G = \pi_k(u)$ via the stick-breaking transformation. Because this transformation is invertible, any distribution on quantity u implies a corresponding distribution on π^G. We can choose to work with either π^G or u when convenient. We find that usually it is easier to work with u.

Observation model prior on global cluster shape parameters. Specifying a DP mixture model requires a choice for the exponential family conjugate-prior density P for observation parameters $\{\phi_k\}_{k=1}^{\infty}$. As discussed in Sec. 2.1.3, each cluster's parameters are drawn *i.i.d.* from this density: $\phi_k \sim P(\phi_k|\bar{\tau},\bar{\nu})$. Recall that hyperparameter $\bar{\nu}$ determines the effective-sample-size of this prior, while $\bar{\tau}$ then determines then expected mean parameter: $\mathbb{E}_{\phi_k\sim \mathrm{P}}[\mu(\phi_k)] = \frac{\bar{\tau}}{\bar{\nu}}$.

3.2.2 Generative model for local variables

Given fixed global parameters, we sample a cluster label from the infinite set $\{1, 2, \ldots\}$ according to the probability vector $\pi(u)$, a deterministic function of u. Then, given the chosen cluster label z_n, we draw an observation x_n from the observation model's likelihood density with parameter ϕ_{z_n}.

$$z_n \sim \mathrm{Cat}_{\infty}(\pi_1(u), \ldots \pi_k(u), \ldots) \tag{3.8}$$

$$x_n \sim \mathrm{L}(\phi_{z_n}) \tag{3.9}$$

We previously introduced the exponential family likelihood density L in Sec. 2.1.3.

The complete joint probability is then

$$\log p(x, z, u, \phi) = \sum_{n=1}^{N}\sum_{k=1}^{\infty} \delta_k(z_n)\Big(\log \mathrm{L}(x_n|\phi_k) + \log \pi_k(u)\Big) + \sum_{k=1}^{\infty} \log \mathrm{P}(\phi_k) + \log \mathrm{Beta}(u_k|1,\gamma) \tag{3.10}$$

where by substituting in the definition $\pi_k(u) = u_k \prod_{\ell<k}(1-u_\ell)$ we have

$$\log \pi_k(u) = \log u_k + \sum_{\ell=1}^{k-1} \log(1-u_\ell) \tag{3.11}$$

3.3 Posterior inference as a variational optimization problem

Given a dataset x, our goal is to estimate the posterior distribution for all latent random variables in Fig. 3.1. That is, we'd like a joint posterior $p(\phi, u, z \mid x)$, over the three random variables of interest: the global conditional appearance probabilities u (which determine $\pi(u)$, the global shape parameters ϕ and the local assignment variables z. As with the finite mixture model from Ch. 2, we develop a variational optimization algorithm (Wainwright and Jordan, 2008) in three steps. First, in Sec. 3.3.1 we define a simplified family of probability distributions for our variables of interest z, u, and ϕ, which forms our approximate posterior distribution. Next, in Sec. 3.3.2 we define an optimization problem whose objective is to find the free variables under which the approximate posterior is as close as possible in KL divergence to the true, intractable posterior. Third, in Sec. 3.4 we define an algorithm which, given input data x, delivers the free parameters that solve our optimization problem.

3.3.1 Mean-field approximate posterior

Let $q(z, u, \phi)$ denote our approximate posterior. As prevously discussed, without any simplifying assumptions even enumerating all possible posterior values is infeasible for all but the smallest datasets. To make our optimization problem computationally feasible, we assume that each of the assignments z_n is independent of the others, and each of the cluster parameters u_k, ϕ_k are independent of the others. That is, we can factorize the approximate density as

$$q(u, \phi, z) = \prod_{k=1}^{\infty} q(\phi_k) \cdot \prod_{k=1}^{\infty} q(u_k) \cdot \prod_{n=1}^{N} q(z_n) \tag{3.12}$$

We further assume each independent factor of this approximate posterior has a density that belongs to the exponential family. The form of this density is chosen to mimic the generative model:

$$q(\phi) = \prod_{k=1}^{\infty} \mathrm{P}(\phi_k | \hat{\tau}_k, \hat{\nu}_k) \tag{3.13}$$

$$q(u) = \prod_{k=1}^{\infty} \mathrm{Beta}(u_k | \hat{\eta}_{k1}, \hat{\eta}_{k0}) \tag{3.14}$$

$$q(z) = \prod_{n=1}^{N} \mathrm{Cat}_{\infty}(z_n | \hat{r}_{n1}, \ldots \hat{r}_{nk} \ldots) \tag{3.15}$$

Under this chosen factorization, the variables $\hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r}$ are the free parameters of our approximate posterior. The goal of variational optimization is to find specific values of these free parameters that make $q(u, \phi, z)$ a good approximation to the true posterior. We always denote free parameters with hats to make clear which variables are instantiated and optimized by the algorithm.

Approximate posterior $q(u_k)$ for global cluster frequencies

Each cluster k in our countably infinite set is given an independent posterior factor $q(u_k)$. Just like its generative counterpart $p(u_k)$, we assume that the approximate posterior factor $q(u_k)$ takes the form of a Beta distribution. Below, we discuss two possible parameterizations of a Beta approximate posterior distribution. First, the common parameterization which is most straightforward. Second, an alternative which specifically decouples a mean and a concentration.

Common parameterization. Let factor $q(u_k)$ takes the form of a Beta distribution with common parameters $\hat{\eta}_{k1}, \hat{\eta}_{k0}$: $u_k \sim \mathrm{Beta}(u_k | \hat{\eta}_{k1}, \hat{\eta}_{k0})$. These free parameters $\hat{\eta}_{k1} > 0, \hat{\eta}_{k0} > 0$ are global parameters for the allocation model. If $\hat{\eta}_{k1} >> \hat{\eta}_{k0}$, then $\mathbb{E}_q[u_k] \approx 1$; otherwise if $\hat{\eta}_{k0} >> \hat{\eta}_{k1}$ then $\mathbb{E}_q[u_k] \approx 0$. This is why we index these free parameters with 1 and 0: they may be interpreted as pseudo-counts for the events $u_k = 1$ and $u_k = 0$.

Alternative parameterization. We can also write the approximate posterior in terms of parameters $\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^{\infty}$:

$$q(u) = \prod_{k=1}^{\infty} \text{Beta}(u_k | \hat{u}_k \hat{\omega}_k, (1 - \hat{u}_k)\hat{\omega}_k) \tag{3.16}$$

Under this parameterization, the free parameter $\hat{u}_k \in [0, 1]$ exactly sets the expected value of u_k: $\mathbb{E}_q[u_k] = \hat{u}_k$, while $\hat{\omega}_k > 0$ controls the variance.

There is a one-to-one invertible transformation between $\hat{u}, \hat{\omega}$ and $\hat{\eta}$:

$$\begin{aligned} \hat{\eta}_{k1} &= \hat{u}_k \hat{\omega}_k & \hat{u}_k(\hat{\eta}) &= \frac{\hat{\eta}_{k1}}{\hat{\eta}_{k1} + \hat{\eta}_{k0}} \\ \hat{\eta}_{k0} &= (1 - \hat{u}_k)\hat{\omega}_k & \hat{\omega}_k(\hat{\eta}) &= \hat{\eta}_{k1} + \hat{\eta}_{k0} \end{aligned} \tag{3.17}$$

This alternative parameterization is useful because the expected value of $\pi^G(u)$ can be written entirely in terms of $\hat{u}$:

$$\mathbb{E}_q[\pi^G(u)] = \hat{u}_k \prod_{\ell=1}^{k-1} (1 - \hat{u}_\ell) \tag{3.18}$$

Approximate posterior $q(\phi_k)$ for global cluster shape

Each cluster k in our countably infinite set is given an independent posterior factor $q(\phi_k)$ for its shape parameter ϕ. Following the analysis in Sec. 2.1.3, we assume this factor comes from the conjugate prior family P. The factor has two free parameters: pseudo-count $\hat{\nu}_k > 0$ and vector parameter $\hat{\tau}_k$ which defines the cluster shape.

Local assignment factor $q(z_n)$

Under the generative model, the assigned cluster label z_n for observation n could use any of the infinitely many cluster indices. Thus, we have naively written above that $q(z_n)$ defines a discrete distribution over infinitely many clusters. We can write the free parameters for this factor as a vector of countably infinite length with one entry per cluster: $\hat{r}_n = \hat{r}_{n1}, \hat{r}_{n2}, \ldots \hat{r}_{nk}, \ldots$. To be a valid parameter for the categorical distribution, this infinite vector must obey two constraints: (1) every entry is non-negative: $\hat{r}_{nk} \geq 0$, and (2) the whole vector sums to unity: $\sum_{k=1}^{\infty} \hat{r}_{nk} = 1$. Intuitively, each scalar entry $\hat{r}_{nk} \in [0, 1]$ represents the probability data x_n is assigned to cluster k under the approximate posterior. This probability value is sometimes called the *responsibility*. Formally, $\hat{r}_{nk} \triangleq \mathbb{E}_{q(z)}[\delta_k(z_n)]$.

The problem with defining $q(z_n)$ as a distribution over an unbounded number of clusters is that we cannot represent an infinite vector $\hat{r}_n$ in a practical implementation. However, despite an infinite number of clusters available *a priori* from a BNP model, given a finite dataset with N atoms only a finite set of K unique cluster labels will be assigned, where $1 \leq K \leq N$. The remaining unassigned clusters will be conditionally independent of the data. Inspired by this fact, we introduce

an additional constraint on the vector $\hat{r}_n$: only the first K entries have non-zero values; all remaining entries with cluster index $k > K$ equal exactly zero.

$$q(z_n|\hat{r}_n, K) = \text{Cat}_\infty(z_n|\hat{r}_{n1}, \hat{r}_{n2}, \dots \hat{r}_{nK}, \dots) \tag{3.19}$$

$$\text{s.t. } \hat{r}_n \geq 0, \quad \sum_{k=1}^{\infty} \hat{r}_{nk} = 1, \quad \sum_{k=K+1}^{\infty} \hat{r}_{nk} = 0 \tag{3.20}$$

Here, integer $K > 0$ is a fixed *truncation level*, which can be held constant throughout classic inference or optimized by our proposal moves in Ch. 4.

This local truncation assumption constrains only the form of the local assignment factor $q(z_n)$. No additional truncation assumptions are needed for either $q(\phi)$ or $q(u)$. Instead, a direct consequence of assuming $q(z_n)$ has positive mass only the first K clusters is that all clusters with index $k > K$ are conditionally independent of the observed data. We thus cannot learn anything about the free parameters $\{\hat{\tau}_k, \hat{\nu}_k\}_{k=K+1}^{\infty}$ from our data, nor the allocation parameters $\{\hat{\eta}_k\}_{k=K+1}^{\infty}$. Instead, these factors have closed-form optimal values which match the corresponding prior marginals $p(u_k)$ and $p(\phi_k)$ under the generative model.

3.3.2 Evidence lower-bound objective function

Given the assumed factorization of $q(u, \phi, z)$ above, we now set up an optimization problem over our free parameters $\hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r}$. The goal is to minimize KL divergence between the approximate posterior q and the true posterior (Wainwright and Jordan, 2008).

$$\arg\min_{\hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r}} \text{KL}(q(u, \phi, z|\hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r})||p(u, \phi, z|x)) \tag{3.21}$$

Computing this KL divergence directly is not possible. However, following the arguments from our variational optimization problem for finite mixture models from Sec. 2.5.2, we define an equivalent optimization problem:

$$\mathcal{L}(x, \hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r}, K) \triangleq \log p(x) - \text{KL}(q(u, \phi, z|\hat{\nu}, \hat{\tau}, \hat{\eta}, \hat{r}, K)||p(u, \phi, z|x)) \tag{3.22}$$

$$= \mathbb{E}_{q(u,\phi,z)}\Big[\log p(x, u, \phi, z) - \log q(u, \phi, z)\Big] \tag{3.23}$$

Under our chosen exponential family forms for each factor of $q(u, \phi, z)$, the expectations that define $\mathcal{L}$ lead to a closed-form function of the free parameters $\hat{\tau}, \hat{\nu}, \hat{\eta}, \hat{r}$. We can tractably evaluate $\mathcal{L}$ and take derivatives, making optimization of the free parameters possible.

We emphasize that because the KL divergence term is always non-negative, the objective $\mathcal{L}$ can be interpreted as a strict lower bound on the marginal likelihood $\log p(x)$. We thus often refer to the function $\mathcal{L}$ as the Evidence Lower BOund, or ELBO. Maximizing $\mathcal{L}$ can be interpreted as improving the approximate posterior's explanation of the data.

We can write the objective $\mathcal{L}$ as a sum of two terms:

$$\mathcal{L}(x, \hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}) = \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\eta}) \tag{3.24}$$

These terms describe distinctly interpretable pieces of the overall model: $\mathcal{L}_{\text{data}}$ gathers terms related to the observation model and $\mathcal{L}_{\text{alloc}}$ gathers terms related to the allocation model. Our chosen notation for each term of the objective highlights the variational parameters involved. For example, $\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\eta})$ is independent of the observation parameters $\hat{\nu}, \hat{\tau}$. These terms may also be functions of the data x and hyperparameters $\mathcal{H}$, but we omit these arguments in notation for simplicity.

This term-by-term breakdown of the objective encourages a modular implementation. Given fixed assignments $\hat{r}$, the solution to finding the optimal observation model parameters $\hat{\nu}, \hat{\tau}$ must be independent of the allocation probability parameters $\hat{\eta}$, and vice versa. This modularization enables our implementation to implement free parameter updates once for each possible observation model or allocation model, and compose these modules to create an overall model.

3.3.3 Observation model term of the objective

The data term is defined by

$$\begin{aligned}\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \nu) &\triangleq \mathbb{E}_q\Big[\log p(x|z,\phi) + \log \frac{p(\phi|\bar{\nu},\bar{\tau})}{q(\phi|\hat{\nu},\hat{\tau})}\Big] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{\infty} \hat{r}_{nk}\mathbb{E}_{q(\phi_k|\hat{\nu}_k,\hat{\tau}_k)}\Big[\log p(x_n|\phi_k)\Big] + \sum_{k=1}^{\infty}\mathbb{E}_{q(\phi_k|\hat{\nu}_k,\hat{\tau}_k)}\Big[\log \frac{p(\phi_k|\bar{\nu},\bar{\tau})}{q(\phi_k|\hat{\nu}_k,\hat{\tau}_k)}\Big]\end{aligned} \tag{3.25}$$

Where we have already substituted in the soft assignment free parameters: $\hat{r}_{nk} = \mathbb{E}_{q(z_n|\hat{r}_n)}[\delta_k(z_n)]$.

We can further simplify this objective by using the derivations from Sec. 2.1.3. We have

$$\begin{aligned}\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) = \sum_{n=1}^{N} h(x_n) &+ \sum_{k=1}^{\infty}\Big(c^P(\bar{\tau},\bar{\nu}) - c^P(\hat{\tau}_k,\hat{\nu}_k)\Big) \\ &+ \sum_{k=1}^{\infty}\Big(N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k\Big)\mathbb{E}_{q(\phi_k)}\Big[-c^L(\phi_k)\Big] \\ &+ \sum_{k=1}^{\infty}\sum_{d=1}^{D}\Big(S_{kd}(x,\hat{r}) + \bar{\tau}_d - \hat{\nu}_{kd}\Big)\mathbb{E}_{q(\phi_k)}\Big[\phi_{kd}\Big]\end{aligned} \tag{3.26}$$

where we have the previously defined sufficient statistics for the expected count $N_k = \sum_{n=1}^{N}\hat{r}_{nk}$ and the expected data statistic $S_k = \sum_{n=1}^{N}\hat{r}_{nk}s(x_n)$ for each cluster k. The remaining expectations are the fundamental ones of the conjugate exponential family likelihood-prior system chosen, which have closed form.

3.3.4 Allocation model term of the objective

We write the allocation model's contribution to the objective as:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\eta}) \triangleq \mathbb{E}_{q(z|\hat{r})q(u|\hat{\eta})}\Big[\log \frac{p(z|\pi)}{q(z|\hat{r})} + \sum_{k=1}^{K}\log \frac{p(u_k|\gamma)}{q(u_k|\hat{\eta})} \tag{3.27}$$

Regrouping terms, we can separate this into an entropy term for the distribution $q(z|\hat{r})$ and a term which gathers all functions of $\hat{\eta}$:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\eta}) = \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) \tag{3.28}$$

Entropy term

The entropy of the assignments is a simple non-linear function of the responsibilities:

$$\mathcal{L}_{\text{entropy}}(\hat{r}) = -\sum_{n=1}^{N} \mathbb{E}_q[\log q(z_n)] = -\sum_{n=1}^{N} \hat{r}_{nk} \log \hat{r}_{nk}. \tag{3.29}$$

Because this is the entropy of a discrete random variable, by definition we known that this term will always be positive or zero: $\mathcal{L}_{\text{entropy}}(\hat{r}) \geq 0$.

Stick-breaking allocation term

Standard conjugate exponential family mathematics yields a simple expression for $\mathcal{L}_{\text{DP-alloc}}$:

$$\mathcal{L}_{\text{DP-alloc}}(\hat{r}) = \mathbb{E}_q[\log p(z|\pi(u)) + \log \frac{p(u)}{q(u)}] \tag{3.30}$$

$$= \sum_{k=1}^{\infty} c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{\eta}_{k1}, \hat{\eta}_{k0}) \tag{3.31}$$

$$+ \sum_{k=1}^{\infty} \Big(N_k(\hat{r}) + 1 - \hat{\eta}_{k1} \Big) \mathbb{E}_q[\log u_k]$$

$$+ \sum_{k=1}^{\infty} \Big(N_k^{>}(\hat{r}) + \gamma - \hat{\eta}_{k0} \Big) \mathbb{E}_q[\log 1 - u_k]$$

Here, the expectations have closed form under the assumed Beta distribution $q(u_k|\hat{\eta}_k)$:

$$\mathbb{E}_q[\log u_k] \triangleq \psi(\hat{\eta}_{k1}) - \psi(\hat{\eta}_{k1} + \hat{\eta}_{k0}) \tag{3.32}$$

$$\mathbb{E}_q[\log 1 - u_k] \triangleq \psi(\hat{\eta}_{k0}) - \psi(\hat{\eta}_{k1} + \hat{\eta}_{k0}) \tag{3.33}$$

and the cumulant function of the Beta distribution is

$$c_{\text{Beta}}(a_1, a_0) \triangleq \log \Gamma(a_1 + a_0) - \log \Gamma(a_1) - \log \Gamma(a_0) \tag{3.34}$$

Finally, the summary statistics $N_k(\hat{r})$ and $N_k^{>}(\hat{r})$ for each cluster $k \in \{1, 2, \dots K, \dots\}$ are defined as:

$$N_k(\hat{r}) \triangleq \sum_{n=1}^{N} \mathbb{E}_{q(z)}[\delta_k(z_n)] = \begin{cases} \sum_{n=1}^{N} \hat{r}_{nk} & \text{if } k \leq K \\ 0 & \text{if } k > K \end{cases} \tag{3.35}$$

$$N_k^{>}(\hat{r}) \triangleq \sum_{n=1}^{N} \sum_{\ell=k+1}^{\infty} \mathbb{E}_{q(z)}\Big[\delta_\ell(z_n)\Big] = \begin{cases} \sum_{\ell=k+1}^{K} N_k(\hat{r}) & \text{if } k < K \\ 0 & \text{if } k \geq K \end{cases} \tag{3.36}$$

We can interpret N_k as the effective count of data observations assigned to cluster label k, and $N_k^{>}$ as the effective count of observations with a label larger than k. Under the assumed truncation level

K, for every inactive cluster label $k > K$ we have $\hat{r}_{nk} = 0$, which implies that both counts are also zero: $N_k = 0$ and $N_k^> = 0$. Further, at the final active cluster index K, we have $N_K^> = 0$ as well by definition. Both quantities N_k and $N_k^>$ in Eq. (3.35) are *linear* functions of the responsibilities $\hat{r}$.

3.4 Update steps for variational optimization

We can now formally define the constrained optimization problem for our free parameters $\hat{\tau}, \hat{\nu}, \hat{\eta}, \hat{r}$ given an observed dataset x:

$$\begin{aligned} \arg\max_{\hat{\tau},\hat{\nu},\hat{\eta},\hat{r}} \quad & \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) \\ \text{subject to } & \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^K \hat{r}_{nk} = 1 && \text{for } n = 1, 2, \ldots N \\ & \hat{\eta}_k \geq 0 && \text{for } k = 1, 2, \ldots \\ & \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} && \text{for } k = 1, 2, \ldots \end{aligned} \tag{3.37}$$

Remember that these functions implicitly require the DP concentration hyperparameter $\gamma > 0$, as well as the observation model hyperparameters $\bar{\tau}, \bar{\nu}$, though our notation omits them for simplicity.

Given the internal structure of the objective, we pursue a block-coordinate ascent algorithm, which proceeds in three steps. Each step updates the free parameters of one factor among $q(z)$, $q(u)$, and $q(\phi)$ while holding the other free parameters fixed. When applied iteratively, these steps are guaranteed to monotonically improve the whole objective $\mathcal{L}$ until it converges to a fixed point local optima.

Below, we define the optimization problem solved by each step of the block-coordinate ascent algorithm. We then describe detailed solutions which solve each problem below in closed-form in the following sections.

Local step: Update local assignment responsibilities

$$\begin{aligned} \arg\max_{\hat{r}} \quad & \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) \\ \text{subject to } & \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^K \hat{r}_{nk} = 1 && \text{for } n = 1, 2, \ldots N \end{aligned} \tag{3.38}$$

Global step: Update observation model global parameters

$$\arg\max_{\hat{\tau},\hat{\nu}} \quad \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) \qquad \text{subject to } \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \text{ for } k = 1, 2, \ldots \tag{3.39}$$

Global step: Update allocation model global parameters

$$\arg\max_{\hat{\eta}} \quad \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) \qquad \text{subject to } \hat{\eta}_k \geq 0 \text{ for } k = 1, 2, \ldots \tag{3.40}$$

3.4.1 Global parameter update step

Global step for observation model

As discussed in Sec. 2.1.3, the optimal observation global parameters $\hat{\tau}_k^*, \hat{\nu}_k^*$ for every cluster $k \in \{1, 2, \dots K, \dots\}$ which solve the optimization problem in Eq. (3.39) can be found in closed form:

$$\begin{aligned} \hat{\tau}_k^* &= S_k(x, \hat{r}) + \bar{\tau} \\ \hat{\nu}_k^* &= N_k(\hat{r}) + \bar{\nu} \end{aligned} \tag{3.41}$$

These optimal values naturally satisfy the required constraints $\hat{\nu}_k^* \in \mathbb{R}^+$ and $\hat{\tau}_k^* \in \mathcal{M}$, so that $\hat{\nu}_k^*$ and $\hat{\tau}_k^*$ remain valid parameters for the density $q(\phi_k)$.

Updates for inactive clusters. For every inactive cluster $k > K$, we have by definition $N_k = 0$ and $S_k = 0$. Thus, the optimal points reduce to the values of the prior hyperparameters: $\hat{\tau}_k^* = \bar{\tau}$ and $\hat{\nu}_k^* = \bar{\nu}$. Throughout our optimization, we assume that the infinitely many inactive cluster parameters are set this way. They need not actually be represented explicitly, because they do not influence any of the other variational free parameters or the exact calculation of the ELBO objective function $\mathcal{L}$.

Simplified $\mathcal{L}_{\text{data}}$ objective term. Substituting the optimal points $\hat{\tau}_k^*, \hat{\nu}_k^*$ into the original objective $\mathcal{L}_{\text{data}}$ in Eq. (3.26) for both active and inactive clusters, we see that the last two lines involving the terms $N_k + \bar{\nu} - \hat{\nu}_k$ and $S_k + \bar{\tau} - \hat{\tau}_k$ will evaluate to zero for every cluster index k. This leaves a greatly simplified expression:

$$\mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}^*, \hat{\nu}^*) = \sum_{n=1}^{N} h(x_n) + \sum_{k=1}^{K} \Big(c^{\text{P}}(\bar{\tau}, \bar{\nu}) - c^{\text{P}}(\hat{\tau}_k^*, \hat{\nu}_k^*) \Big) \tag{3.42}$$

The reference measure term remains unchanged, while the sum over the cumulant functions is now only over the K active clusters, not all countably infinite clusters. This occurs because for all inactive clusters, we have $\hat{\tau}_k^* = \bar{\tau}$ and $\hat{\nu}_k^* = \bar{\nu}$, and thus the difference of c^{P} functions at inactive clusters will necessarily be zero without any explicit evaluation required.

Global step for allocation model

To find the optimal values $\hat{\eta}$ for the optimization problem in Eq. (3.40), we can apply standard constrained optimization techniques like Lagrange multipliers to find the closed-form solution for each cluster k:

$$\begin{aligned} \hat{\eta}_{k1}^* &= N_k(\hat{r}) + 1 \\ \hat{\eta}_{k0}^* &= N_k^{>}(\hat{r}) + \gamma \end{aligned} \tag{3.43}$$

Naturally, these updates preserve the required constraints $\hat{\eta} \geq 0$.

Update for inactive clusters. For any inactive cluster $k > K$, by definition $N_k = 0$ and $N_k^> = 0$, which leaves the fixed points equal to the prior hyperparameters: $\hat{\eta}_{k1}^* = 1$ and $\hat{\eta}_{k0}^* = \gamma$. Throughout optimization, we may assume the inactive cluster parameters are set this way. They need not be represented in memory.

Simplified $\mathcal{L}_{\text{DP-alloc}}$ objective term. Substituting the optimal points $\hat{\eta}^*$ into the function $\mathcal{L}_{\text{DP-alloc}}$ in Eq. (3.30), we find that the coefficients $N_k + 1 - \hat{\eta}_{k1}$ and $N_k^> + \gamma - \hat{\eta}_{k0}$ will evaluate to zero, leaving a simplified expression:

$$\mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}^*) = \sum_{k=1}^{K} c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{\eta}_{k1}, \hat{\eta}_{k0}) \tag{3.44}$$

where again, the difference in cumulant functions for $k > K$ will always evaluate to zero under the optimal settings of $\hat{\eta}^*$, and thus we need not represent these inactive terms of the sum at all.

3.4.2 Local parameter update step

To solve the local step optimization problem in Eq. (3.38), we first simplify the whole objective $\mathcal{L}$ by showing it as a function of the responsibilities at each observation n. We gather those terms that do not depend on the active responsibilities $\hat{r}$ for clusters $k \in \{1, \dots K\}$ together into a constant term:

$$\mathcal{L}(x, \hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}) = \text{const}(x, \hat{\eta}, \hat{\tau}, \hat{\nu}) + \sum_{n=1}^{N} \mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \tag{3.45}$$

Now, the relevant objective function $\mathcal{L}_n$, which captures all terms from $\mathcal{L}_{\text{data}}, \mathcal{L}_{\text{entropy}}, \mathcal{L}_{\text{DP-alloc}}$ relevant to observation n, is given by:

$$\mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \triangleq \sum_{k=1}^{K} \hat{r}_{nk} \left(W_{nk}(x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) - \log \hat{r}_{nk} \right) \tag{3.46}$$

$$W_{nk}(x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \triangleq \mathbb{E}_{q(\phi_k)} \Big[\log p(x_n | \phi_k) \Big] + \mathbb{E}_{q(u)} \Big[\log \pi_k(u) \Big] \tag{3.47}$$

Here, we can interpret each W_{nk} as the log posterior weight that cluster k has on observation n. Larger values indicate that cluster k is more likely to explain observation n. The expectations that define W_{nk} have closed form when $q(\phi_k)$ and $q(u_k)$ come from our chosen exponential family forms.

From the decomposition of $\mathcal{L}$ into a sum of independent terms for each observation, it is clear that our objective may be optimized independently for each observation n.

$$\hat{r}_n^* = \arg\max_{\hat{r}_n} \quad \mathcal{L}_n(\hat{r}_n, x_n, \hat{\eta}, \hat{\tau}, \hat{\nu}) \tag{3.48}$$
$$\text{subject to } \hat{r}_n \geq 0 \text{ and } \textstyle\sum_{k=1}^{K} \hat{r}_{nk} = 1$$

Through standard constrained optimization methods, we find the solution for the optimal responsibility vector $\hat{r}_n^*$. We can compute the optimal vector in closed-form by setting each entry

$k \in \{1, 2, \dots K\}$ to the exponentiated posterior weight $e^{W_{nk}}$ and then normalizing:

$$\hat{r}^*_{nk} = \frac{e^{W_{nk}}}{\sum_{\ell=1}^{K} e^{W_{n\ell}}}, \tag{3.49}$$

which by inspection is guaranteed to obey the required constraints that responsibilities must be non-negative and sum to one. By performing the update in Eq. (3.49) at each observation n independently, we will compute the optimal responsibilities $\hat{r}$ for the whole dataset.

3.4.3 Nested truncation w.r.t. number of active clusters

One advantage of our chosen truncation scheme for $q(z)$ is that the resulting simplified family of approximate posterior distributions is *nested* across truncation levels. This means for any truncation level K and accompanying free parameters $\Psi_K = \{\hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}\}$ for active clusters, we can construct a configuration $\Psi_{K+1} = \{\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'\}$ with truncation level $K+1$ with *exactly* the same score under the objective $\mathcal{L}$. Here is the construction:

$$\hat{r}'_{nk} = \begin{cases} \hat{r}_{nk} & \text{if } k \leq K \\ 0 & \text{if } k = K+1 \end{cases} \quad \hat{\eta}'_{k1} = \begin{cases} \hat{\eta}_{k1} & \text{if } k \leq K \\ 1 & \text{if } k = K+1 \end{cases} \quad \hat{\eta}'_{k0} = \begin{cases} \hat{\eta}_{k0} & \text{if } k \leq K \\ \gamma & \text{if } k = K+1 \end{cases} \tag{3.50}$$

$$\hat{\tau}'_k = \begin{cases} \hat{\tau}_k & \text{if } k \leq K \\ \bar{\tau} & \text{if } k = K+1 \end{cases} \quad \hat{\nu}'_k = \begin{cases} \hat{\nu}_k & \text{if } k \leq K \\ \bar{\nu} & \text{if } k = K+1 \end{cases}$$

Using a variational family with nested truncation has several benefits. First, it is easy to *compare* alternative approximate posteriors with different values of K: whichever has the larger score under the objective function $\mathcal{L}$ is the superior model. Second, we can easily *construct* candidate local and global parameters that differ from some current assignments $\hat{r}$ by adding or removing one cluster. Both these properties are used in our new proposal moves that adapt the truncation K, which we develop in Ch. 4.

Previous work on truncation. Our chosen nested truncation was previously suggested for topic models (Teh et al., 2008; Bryant and Sudderth, 2012), and has been more recently applied to some hidden Markov models (Johnson and Willsky, 2014). One well-known alternative is direct truncation of the global parameters of the stick-breaking process (Blei and Jordan, 2006). This choice enforces a more restrictive assumption about the last active index K: $\mathbb{E}_q[u_K] \triangleq 1 - \sum_{k=1}^{K-1} \mathbb{E}_q[\pi_k]$. In other words, the entire unit probability mass must be allocated to the first K clusters, leaving zero probability for inactive clusters despite the DP prior always placing small positive probability there. In addition to artificially inflating the final active cluster, this assumption leads to more complicated updates for $\hat{\eta}$ when the truncation level is changed. Our nested truncation is also more more efficient and broadly applicable than the truncation suggested by Kurihara et al. (2006), which explicitly enforces that the aggregate probability mass assigned to inactive topics matches its prior value.

Algorithm 3.1 Variational coordinate ascent for DP mixture models

Input:

x : dataset
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: initial global parameters of allocation model
γ: allocation model hyperparameter
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: updated global parameters of allocation model

function VariationalCoordAscentForDPMix($x, K, \hat{\eta}, \hat{\tau}, \hat{\nu}$)
  while not converged **do**
    for $n \in 1, 2, \dots N$ **do** ▷ Local step
      for $k \in 1, 2, \dots K$ **do**
        $C_{nk} \leftarrow \mathbb{E}_q[\log p(x_n|\phi_k)]$
        $W_{nk} \leftarrow C_{nk} + \mathbb{E}_q[\log \pi_k(u)]$
      $\hat{r}_n \leftarrow$ RespFromWeights(W_n)

    for $k \in 1, 2, \dots K$ **do** ▷ Summary step
      $S_k \leftarrow \sum_{n=1}^N \hat{r}_{nk} s(x_n)$
      $N_k \leftarrow \sum_{n=1}^N \hat{r}_{nk}$
      $N_k^> \leftarrow \sum_{k+1}^K N_k$

    for $k \in 1, 2, \dots K$ **do**
      $\hat{\tau}_k \leftarrow S_k + \bar{\tau}$ ▷ Global step for observation parameters
      $\hat{\nu}_k \leftarrow N_k + \bar{\nu}$
      $\hat{\eta}_{k1} \leftarrow N_k + 1$ ▷ Global step for allocation parameters
      $\hat{\eta}_{k0} \leftarrow N_k^> + \gamma$
  return $\hat{\eta}, \hat{\tau}, \hat{\nu}$

Block coordinate ascent algorithm for approximate posterior inference for the DP mixture model. Each converged solution represents a local optima of the optimization problem in Eq. (3.37).

3.5 Algorithms for variational optimization

3.5.1 Full-dataset block coordinate ascent algorithm

Alg. 3.1 defines the block-coordinate ascent update steps derived in the previous section as a coherent algorithm. Given some initial global parameters and a dataset, it iteratively cycles between the local step for $q(z_n)$ in Eq. (3.48) and the global parameter updates for the observation model in Eq. (3.41) and the allocation model in Eq. (3.43).

Summary statistics for global updates

Taking a step back, we emphasize the fundamental roles that the summary statistics N and S play in Alg. 3.1. First, they are sufficient statistics for the global updates. This means that given N and S, we have everything we need to compute the optimal global parameters for both the allocation model

and the observation model. Second, these quantities N and S share two key properties: *scalability* and *additivity*.

First, a summary is *scalable* if its size is independent of the size of the dataset, measured by the number of documents D or atoms N. Practically, this means it is cheap to store and manipulate these quantities, even as the dataset grows toward infinite size. Performing any global update will always be a constant-time operation with respect to the number of documents or atoms, once the summary statistics are computed.

Second, we say a summary is *additive* if it is computable by summing over independent terms, where each term comes from either one data atom or one document (group). Additivity means that if we want to summarize a large dataset x with N total atoms, we could first divide it into two disjoint pieces x_1 and x_2, with N_1 and N_2 atoms each, satisfying $N_1 + N_2 = N$. Given summaries for the separate pieces S_{1k} and S_{2k}, we can compute the summary for the whole dataset simply: $S_k^G = S_{1k} + S_{2k}$. Throughout, we will use the superscript G notation for whole-dataset summaries, like N^G or S^G, to differentiate these from summaries for subsets of the data. When summaries are additive, both *parallel* and *online* processing becomes possible, allowing variational inference for DP mixture models and extensions to scale to large datasets.

Summaries for computing the objective $\mathcal{L}$

For all observation models, we assume that the observation-model objective $\mathcal{L}_{\text{data}}$ is a purely *linear* function of the summaries N and S, as defined in Eq. (3.26). In contrast, the allocation-model objective $\mathcal{L}_{\text{alloc}}$ for DP mixture models has a non-linear function of $\hat{r}$: the entropy term $\mathcal{L}_{\text{entropy}}(\hat{r})$ in Eq. (3.29). We can create a summary statistic H, a vector with an entry for each active cluster, for calculating this entropy:

$$H_k(\hat{r}) \triangleq -\sum_{n=1}^{N} \hat{r}_{nk} \log \hat{r}_{nk} \tag{3.51}$$

Then, we simply compute $\mathcal{L}_{\text{entropy}} = \sum_{k=1}^{K} H_k$. The summary H has the properties of *scalable size* and *additivity*, just like the others N and S.

Given a set of possible assignment responsibilities $\hat{r}$ for data x, we can compute the summaries $\{S, N, H\}$, discard $\hat{r}$ afterwards, and have everything we need to find the optimal global parameters and compute the resulting objective score $\mathcal{L}$.

Towards scalable algorithms

The variational optimization algorithm in Alg. 3.1 is sometimes called a *full-dataset* algorithm or a *batch* algorithm, implying it performs a local step for every observation in the dataset before each global step update. If the number of observations N in the dataset is large, for example in the hundreds of thousands or more, this algorithm will be slow to propagate information between the local updates and the global parameters. A natural idea is to perform global updates at more frequent intervals, after processing a small subset or *batch* of data. In the sections below, we will

Algorithm 3.2 Stochastic variational coordinate ascent for DP mixture models

Input:

x : dataset
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: initial global parameters of allocation model
γ: allocation model hyperparameter
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters
N_b : integer, desired size of each minibatch
δ : positive real, "delay" in learning rate schedule
κ : value in $(0.5, 1.0]$, controls rate of learning rate decay

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: updated global parameters of allocation model

function StochasticVariationalForDPMix($x, K, \hat{\eta}, \hat{\tau}, \hat{\nu}$)
  for iteration $t \in 1, 2, \ldots$ **do**
    $\mathcal{D}_t \leftarrow$ SampleWithoutReplacement($\{1, 2, \ldots N\}, N_b$) ▷ Sample current minibatch
    for $n \in \mathcal{D}_t$ **do** ▷ Local step
      for $k \in 1, 2, \ldots K$ **do**
        $C_{nk} \leftarrow \mathbb{E}_q[\log p(x_n|\phi_k)]$
        $W_{nk} \leftarrow C_{nk} + \mathbb{E}_q[\log \pi_k(u)]$
      $\hat{r}_n \leftarrow$ RespFromWeights(W_n)
    for $k \in 1, 2, \ldots K$ **do** ▷ Summary step
      $S_{tk} \leftarrow \sum_n \hat{r}_{nk} s(x_n)$
      $N_{tk} \leftarrow \sum_n \hat{r}_{nk}$
    $\xi_t = (\delta + t)^{-\kappa}$ ▷ Update step size
    for $k \in 1, 2, \ldots K$ **do**
      $\hat{\tau}_k \leftarrow (1 - \xi_t)\hat{\tau}_k + \xi_t\Big(\bar{\tau} + \frac{N}{N_b} S_{tk}\Big)$ ▷ Global step for observation parameters
      $\hat{\nu}_k \leftarrow (1 - \xi_t)\hat{\nu}_k + \xi_t\Big(\bar{\nu} + \frac{N}{N_b} N_{tk}\Big)$
      $\hat{\eta}_{k1} \leftarrow (1 - \xi_t)\hat{\eta}_{k1} + \xi_t\Big(1 + \frac{N}{N_b} N_{tk}\Big)$
      $\hat{\eta}_{k0} \leftarrow (1 - \xi_t)\hat{\eta}_{k0} + \xi_t\Big(\gamma + \frac{N}{N_b} \sum_{\ell=k+1}^K N_{t\ell}\Big)$
  return $\hat{\eta}, \hat{\tau}, \hat{\nu}$

Stochastic block coordinate ascent algorithm for solving the optimization problem in Eq. (3.37). At each iteration, the algorithm first samples a small batch of documents and then executes local and global optimization steps. The global step differs from the full-dataset algorithm by involving a learning rate ξ and a natural gradient step.

describe two common methods for scaling variational coordinate ascent to large datasets: stochastic variational inference and memoized variational inference.

3.5.2 Stochastic variational inference

Stochastic variational inference (SVI) was first introduced for topic models by Hoffman et al. (2010) and later described more comprehensively in Hoffman et al. (2013). SVI scales to a large dataset

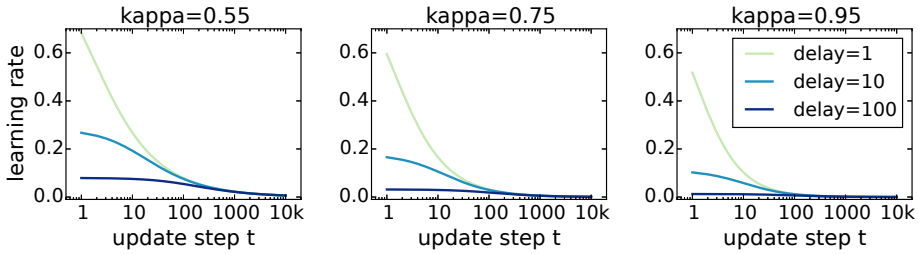

Figure 3.2: Illustration of possible learning rate schedules for stochastic variational inference. We set the learning rate ξ_t at update step t to be $\xi_t = (\delta + t)^{-\kappa}$, where the delay parameter $\delta \geq 0$ downweights early iterations at larger values, and the exponential decay parameter κ determines how fast the rate decays to zero. To obtain the Robbins-Munroe guarantees, we require that $\kappa \in (0.5, 1]$.

of N observations by performing two conceptual steps at each iteration t: First, a local step to find the optimal responsibilities for a small subset or *batch* of data. Second, a global step which stochastically updates the global parameters to minimize $\mathcal{L}$ via a noisy gradient step computed using only the current batch. Each iteration's batch of data is sampled uniformly at random from the whole dataset. The complete algorithm for our DP mixture model optimization problem is specified in Alg. 3.2, and the relevant steps are detailed below. The usual formulation assumes that the whole dataset has a finite total number of atoms N that is known in advance, but later work has extended this to never-ending data streams (Theis and Hoffman, 2015).

SVI local step. The local step of SVI is unchanged from the full-dataset algorithm, except that it only processes a subset of data at every iteration. Denote the subset of data indices selected at iteration t as $\mathcal{D}^t \subset \{1, 2, \dots N\}$. This set is chosen uniformly at random without replacement at every iteration. The size of this set, which we denote $N_b = |\mathcal{D}_t|$, must be pre-specified and obey the constraints $1 \leq N_b \leq N$. Once the set $\mathcal{D}_t$ is chosen, the local step update occurs: for every data index n in the current batch, we compute the optimal responsibilities $\hat{r}_n^*$ for the approximate posterior factor $q(z_n)$ via Eq. (3.48). We emphasize that these updates exactly optimize the single observation objective $\mathcal{L}_n$. Aggregating over all observations in the set $\mathcal{D}_t$, we can produce summaries for the effective counts N_{tk} and effective data-statistic S_{tk} for each active cluster in the batch at iteration t.

SVI global step. The global step update for SVI is noticeably different from that in Alg. 3.1. As a gradient descent algorithm, the value of a global parameter at iteration t depends on the value at iteration $t-1$. For example, global parameter $\hat{\tau}_k$ will start at some initial value $\hat{\tau}_k^1$ and then proceed through a sequence of values $\hat{\tau}_k^2, \dots, \hat{\tau}_k^t, \hat{\tau}_k^{t+1} \dots$ as more batches are seen. The value at iteration t depends on a learning rate ξ^t, a positive scalar that decays over iterations. The larger ξ_t is, the more "forgetful" the algorithm is of previous iterations.

For global variables with exponential family conditionals, Hoffman et al. (2013) recommend using

the *natural gradient* global step at each batch. This is the gradient under the natural parameterization, not the mean parameterization, of the relevant exponential family density. Natural gradients yield closed-form computations that share much in common with full-dataset updates. For each global parameter, the natural gradient update has three steps, which we walk through below for the data shape parameter $\hat{\tau}_k$.

First, create a noisy estimate of the full-dataset summary statistic $S_k \approx \frac{N}{N_b} S_{tk}$. Here, the amplification factor $\frac{N}{N_b} \geq 1$ scales up the current batch's statistic to have the effective size of the full dataset. Next, using this estimated summary to perform a global step, creating a batch-specific global parameter $\hat{\tau}_{tk} \approx \bar{\tau} + \frac{N}{N_b} S_{bk}$. Finally, interpolate between the batch-optimal value and the previous global value with the current learning rate ξ_t.

$$\hat{\tau}_k = \xi_t \Big(\bar{\tau} + \frac{N}{N_b} S_{tk} \Big) + (1 - \xi_t) \hat{\tau}_k \tag{3.52}$$

When the learning rate decays according to the Robbins-Munro conditions, the stochastic variational algorithm provably converges to a *local* optimum of the whole-dataset objective function $\mathcal{L}$ (Hoffman et al., 2013). Hoffman et al. (2013) suggest defining the learning rate decay schedule such that $\xi_t = (\delta + t)^{-\kappa}$, where the parameter $\delta \geq 0$ acts as a delay where larger values downweight the impact of early iterations, and parameter $\kappa \in (0.5, 1]$ is a decay factor that determines how fast the step size ξ_t approaches zero as t increases. See Fig. 3.2 for a visualization of how the learning rate ξ_t decays as the number of iterations t grows for a variety of possible decay schedules.

SVI has several advantages over full-dataset inference, including much faster propagation between local and global updates and a potential for the random gradient steps to help escape very shallow local optima that might trap the full-dataset method. However, performance is extremely sensitive to the how aggressively the step size ξ_t decays as more updates occur. The decay schedule for this parameter often has nuisance parameters that must be tuned for ideal performance, which can be prohibitively expensive. Performance can also be sensitive to chosen batch size N_b.

When conditional distributions do not come from the exponential family, we may still perform stochastic updates by computing the gradient of the objective $\mathcal{L}$ with respect to the global parameter of interest. However, this can require costly derivation effort by the practioner.

3.5.3 Memoized variational inference

Memoized variational inference (MVI) is an alternative to SVI that is just as scalable but avoids the nuisance of learning rates and derivation complexities altogether. MVI was first introduced in Hughes and Sudderth (2013) and inspired by the incremental expectation-maximimization algorithm of Neal and Hinton (1998). With just one batch, MVI is exactly equal to whole-dataset inference. However, if the dataset is divided into B smaller batches, MVI can be much more scalable than the full-dataset algorithm.

Like SVI, each update iteration consists of a local step at a single batch followed immediately by a global step. Unlike SVI, each global update uses aggregated summaries that represent the whole dataset, not just the current batch. These whole-dataset summaries are incrementally updated at

Algorithm 3.3 Memoized variational coordinate ascent for DP mixture models

Input:

x : dataset, divided into B fixed batches
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: initial global parameters of allocation model
γ: allocation model hyperparameter
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{\eta}_k\}_{k=1}^K$: updated global parameters of allocation model

function MemoizedVariationalForDPMix($x, K, \hat{\eta}, \hat{\tau}, \hat{\nu}$)
  $N^G, S^G \leftarrow 0$ ▷ Initialize global statistics
  for batch $b \in 1, 2, \dots B$ **do**
    $N_b, S_b \leftarrow 0$ ▷ Initialize batch statistics

  for lap $\ell \in 1, 2, \dots L_{max}$ **do**
    for batch $b \in$ Shuffle$(1, 2, \dots B)$ **do**
      for $n \in \mathcal{D}_b$ **do** ▷ Local step for batch b
        for $k \in 1, 2, \dots K$ **do**
          $C_{nk} \leftarrow \mathbb{E}_q[\log p(x_n|\phi_k)]$
          $W_{nk} \leftarrow C_{nk} + \mathbb{E}_q[\log \pi_k(u)]$
        $\hat{r}_n \leftarrow$ RespFromWeights(W_n)

      $S^G \leftarrow S^G - S_b$ ▷ Decrement previous batch statistics
      $N^G \leftarrow N^G - N_b$

      for $k \in 1, 2, \dots K$ **do** ▷ Summary step for batch b
        $S_{bk} \leftarrow \sum_{n \in \mathcal{D}_b} \hat{r}_{nk} s(x_n)$
        $N_{bk} \leftarrow \sum_{n \in \mathcal{D}_b} \hat{r}_{nk}$

      $S^G \leftarrow S^G + S_b$ ▷ Increment new batch statistics
      $N^G \leftarrow N^G + N_b$

      for $k \in 1, 2, \dots K$ **do**
        $\hat{\tau}_k \leftarrow S_k^G + \bar{\tau}$ ▷ Global step for observation parameters
        $\hat{\nu}_k \leftarrow N_k^G + \bar{\nu}$
        $\hat{\eta}_{k1} \leftarrow 1 + N_k^G$ ▷ Global step for allocation parameters
        $\hat{\eta}_{k0} \leftarrow \gamma + \sum_{\ell=k+1}^K N_\ell^G$
  return $\hat{\eta}, \hat{\tau}, \hat{\nu}$

Block coordinate ascent algorithm for solving the optimization problem in Eq. (3.37). Assumes data is divided into a fixed set of B batches before processing begins. At each batch, we perform a local optimization step and a global optimization step. For the DP mixture model, this algorithm is guaranteed to monotonically increase the ELBO objective function $\mathcal{L}$ after every step once the first *lap* through the data is completed.

each batch. With a modest cost of additional memory required for tracking batch-specific summaries that enable these incremental updates, we can perform noise-free global steps that exactly optimize

the full-dataset objective $\mathcal{L}$. The target application of MVI is datasets of large but finite size through which we can afford to make dozens but maybe not thousands of update iterations. Variants of MVI easily generalize to streaming settings, but our experiments show that when it is affordable to do so, making dozens of repeated passes through a dataset yields substantially better results.

MVI for fixed-size datasets. Given a fixed dataset of D total documents, we must divide it into B distinct batches before memoized inference can begin. Any assignment of documents to batches may be used; random assignment is effective but not necessary. We need only require that the same batches are used throughout the algorithm, as we make repeated passes through the dataset. We denote the set of atom indices n assigned to batch b as $\mathcal{D}_b$.

Alg. 3.3 provides the formal steps of the standard memoized training algorithm for our DP mixture model. The algorithm completes many passes through the complete dataset, processing one batch at a time. We call each pass through the dataset a *lap*. In each lap, we visit each batch exactly once. Any ordering is allowed so long as all batches are visited evenutally. As a default, we recommend a different random order for each lap.

When visiting batch b, MVI first performs a local step to obtain batch specific summaries N_b, S_b, H_b. These fresh values are then used to *incrementally* update whole-dataset summaries N^G, S^G, H^G, where again the superscript G indicates the summary value is a global aggregation of all batch values. During the *first* visit to batch b, each global summary is simply incremented: $N_k^G = N_k^G + N_{bk}$. On later visits, each global summary update requires adding the new batch-summary and subtracting the previous batch-specific value: $N_k^G = N_k^G + N_{bk} - N_{bk}^{memory}$. After every incremental update of a global summary, we rewrite the memoized value for batch b: $N_{bk}^{memory} = N_k^b$. After each batch-specific local step and resulting incremental summary step, the global summaries N^G, S^G are fully consistent with the most recent assignments to all batches. We may thus use these summaries in a global step to obtain optimal global parameters, just as in the standard full-dataset algorithm.

Overall, memoized inference has the same per-batch run-time complexity as stochastic variational inference. However, it does have modest additional storage cost that grows linearly with the number of batches B and number of clusters K – $\mathcal{O}(BK)$ – due to required storage for N_{bk}^{memory} and other memoized batch summaries. Importantly, this storage requirement does not need to scale directly with the number of documents or data atoms, only the number of batches.

Memoized inference is advantageous because each global step delivers global parameters that are optimal under the whole-dataset objective, avoiding the noisiness inherent in stochastic updates and the complexities of selecting a learning rate. Furthermore, during MVI inference we can exactly evaluate the whole-dataset objective $\mathcal{L}$ at any point after the first complete lap. This allows the design of birth and merge proposal moves that are checked exactly against the whole-dataset objective, and only accepted if they improve the $\mathcal{L}$ score. This guaranteed improvement makes our merge moves more reliable than earlier work that made decisions about removing components based on a noisy objective estimate based only on a single batch (Bryant and Sudderth, 2012).

Memoized algorithm and monotonicity guarantees

For the DP mixture model, the memoized algorithm is guaranteed to monotonically improve the objective $\mathcal{L}$ at every iteration. Crucially, this can be done without storing the complete responsiblities $\hat{r}$ for the full dataset, and instead only computing the per-batch responsibilities $\hat{r}_b$ on demand. The reason is that the local step objective for $\hat{r}_b$ in Eq. (3.38) is convex and therefore has one universal optimum given fixed global parameters $\hat{\tau}, \hat{\nu}$ and $\hat{\eta}$.

The guaranteed increase of the objective function $\mathcal{L}$ provides a useful practical test to verify the correctness of any implementation. Any implementation which yields a non-monotonically increasing $\mathcal{L}$ trace must have a bug. No such simple test exists for stochastic variational methods.

Memoized algorithm for streaming

It is straightforward to apply the ideas behind memoized inference to applications that require learning from a never-ending stream of data. At more data arrives, we may simply visit each never-before-seen batch b, execute a local step that outputs batch-specific summaries like N_{bk}, and use an incremental update $N_k^G = N_k^G + N_{bk}$ to obtain whole-dataset statistics exactly consistent with all data seen thus far. The aggregated summaries can be used in standard global step to deliver global parameters that are optimal given the local assignments from all previously-visited batches. This procedure is equivalent to a synchronous version of streaming variational inference presented in Alg. 3 of Broderick et al. (2013). Broderick et al. (2013) also present a version of the algorithm which iterates at each batch b through many local-then-global steps until convergence. If a batch is never visited more than once, no memory for per-batch summaries is needed.

3.5.4 Initialization of global parameters

Our iterative optimization algorithm requires an initialization of the global variational parameters for the observation model $\hat{\nu}, \hat{\tau}$ and allocation model $\hat{\eta}$. Any initialization that provides parameter values in the correct domain is valid, but some are better than others as later experiments will show.

To initialize all variational training algorithms for DP mixtures, we recommend using the Bregman k-means++ procedure from Alg. 2.1. This algorithm applies to the entire class of exponential-family observation models and yields solutions for which some formal approximation ratios have been proved for simpler but related objectives.

Many alternatives to initialize these global cluster parameters could exist. For example, Future work could investigate the alternative choice of using a small-variance asymptotics algorithm like DP-means, which can also handle all exponential family (EF) observation models (Jiang et al., 2012).

Initializing MVI. Initializing MVI requires specifying global parameter vectors $\hat{\nu}, \hat{\tau}$, and $\hat{\eta}$. These are all we need to perform a local step at the first batch. However, during the first lap each global step uses summaries aggregated only from visited data batches, not the entire dataset. For example, after the first batch the summaries provided to the global step represent only $|\mathcal{D}_1|$ observations,

not all N observations. Consequently, after seeing only one batch, the global step is optimizing the global parameters only for the terms in the objective related to contents of one batch, not the full dataset. Every global step update during the first lap optimizes a different, but related, objective. This evolving objective converges to the full-dataset objective after the completion of the first lap.

Consider the case of trying to initialize MVI using a smart, data-informed guess for global parameters. Perhaps this guess comes from running MVI on a related dataset, or uses some hypothesized "ground-truth" parameters. The data in the first few batches may not be completely representative of the clusters present this initialization. Not every batch will contain many examples of useful but rare clusters. Thus, naively employing the global step during the first few batches can yield new parameters that are quite far from our smart initial guess. If some initial cluster that is useful for later batches does not appear in the first batch, it will be completely erased after batch one and never be recovered by later batches. The greedy nature of the few global steps of MVI can yield parameters that underperform on the full-dataset objective when unseen batches are taken into account.

We first and foremost hope that our birth proposal moves fix this problem by creating new, useful clusters as needed. However, if birth moves are unavailable and an informative initialization is used, we recommend delaying any global steps until sufficiently many batches are seen, perhaps even the full dataset if it is small enough. Until the total number of units processed exceeds a user-specified threshold, at each batch we perform local steps and increment global summaries, but simply keep the global parameters at their initial values. However, without proposal moves we find this delay strategy can often dramatically improve performance.

3.6 Experimental results

In this section, we demonstrate the basic behavior of our full-dataset variational training algorithm on several toy datasets. Our goal is to help the reader understand how performance varies with initialization, with the number of initial clusters K, and with the choice of the observation model ("likelihood"). A specific take-away from all the experiments below should be that these algorithms are highly-vulnerable to local optima, even when using data-driven initialization techniques and large values for the number of initial clusters K. We hope this motivates the exploration of proposal moves in Ch. 4 that make large changes by adding or removing clusters.

Several experiments with the scalable memoized or stochastic algorithms are available in the experiments section of Ch. 4, where we explore how these training methods perform with and without our new proposal moves.

3.6.1 Clustering image patches with zero-mean Gaussian likelihoods

Motivated by success of Gaussian mixture models for image patches by (Zoran and Weiss, 2011), as illustrated earlier in Fig. 1.2, we consider training DP mixture models with zero-mean Gaussian likelihoods.

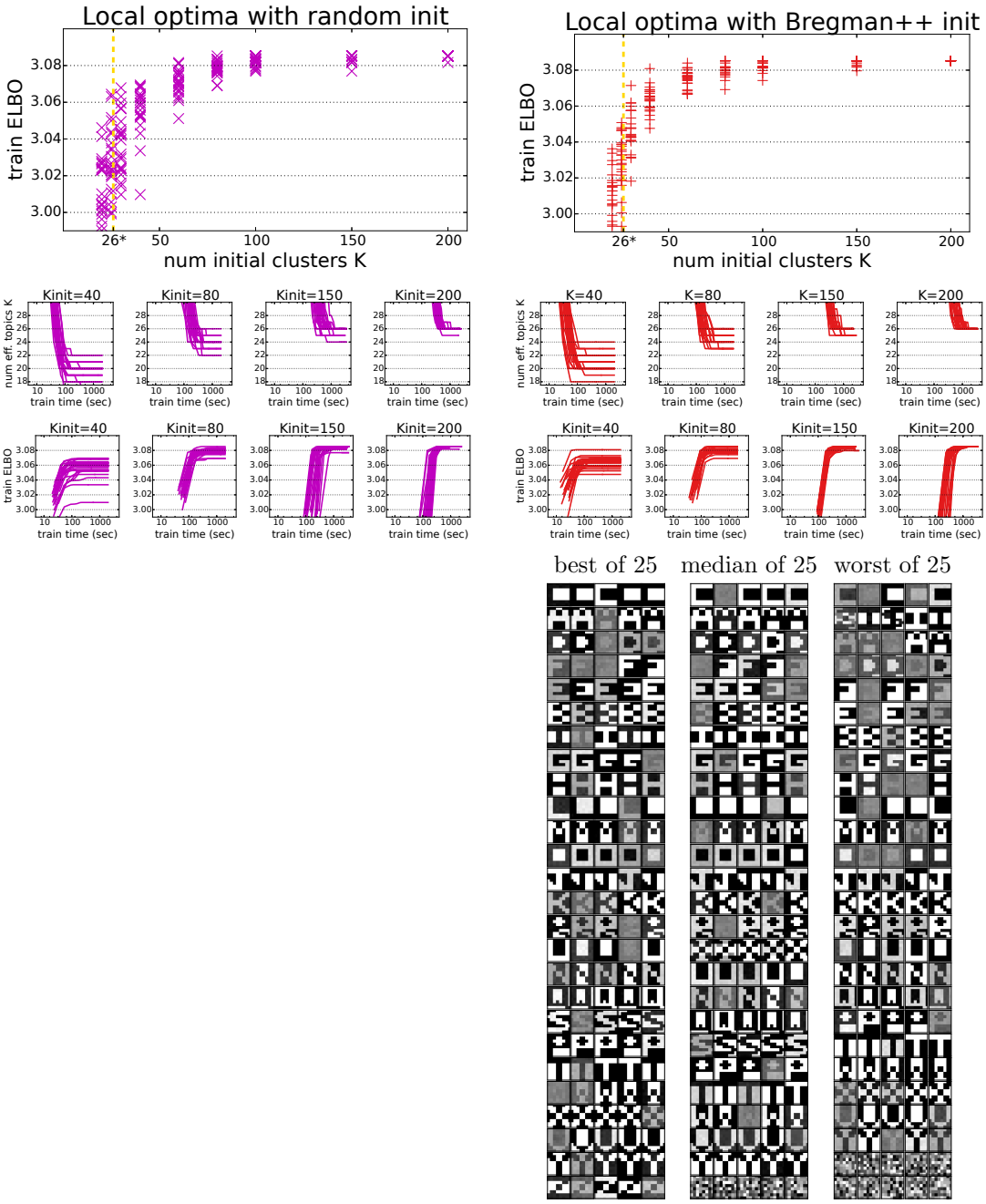

Figure 3.3: Local optima found when clustering toy alphabet images with DP mixture models. Dataset: 50000 synthetic images generated from 26 true clusters (one for each letter of the English alphabet: A, B, C, ... Z). We compare initializations for the variational inference algorithm in Alg. 3.1 and expose local optima problems. *Left column:* random initialization, which chooses K initial images uniformly at random without replacement to initialize K clusters. *Right column:* Bregman k-means++ initialization, which chooses K initial documents as cluster centers via Alg. 2.2. *Bottom Row:* Best, median, and worst of 25 runs, as ranked by $\mathcal{L}$, from $K = 100$ initial clusters.

First, we consider a synthetic dataset with a set of $K = 26$ true clusters, each one generating 8x8 pixel image patches which show a single image from the English alphabet ("A", "B", "C", etc.). Each true cluster is defined by a covariance matrix ϕ_k over the image patch space, as well as a frequency π_k^G, which we set to be monotonically decreasing with a letter's index in the alphabet, so A is most common and Z is least common. Each successive letter has 95% of the probability mass of the previous letter.

After randomly generating $N = 50,000$ distinct observations from this model, we observed 3695 "A" patches, 3272 "B" patches, and 4929 "C" patches. Among the least common letters, we observed 955 "X" patches, 771 "Y" patches, and 529 "Z" patches. Fig. 3.3 shows the results of extensive local optima experiments on the task of clustering this synthetic dataset. We consider several factors governing the initialization: the number of initial clusters and the cluster selection procedure. As a baseline, we consider a random example initialization, where K distinct data observations are selected uniformly at random without replacement from the index set $\{1, 2, \dots N\}$, and these K choices are used to initialize separate clusters. In contrast, we consider the Bregman k-means++ procedure in Alg. 2.2, which selects K observations in a data-driven fashion which tends to find K observations far from each other using the Bregman divergence that comes from our chosen zero-mean Gaussian likelihood.

Our conclusions are listed below:

Rich-get-richer prior bias lets some initial clusters decay to zero usage. To understand what happens here, in the second row of Fig. 3.3 we plot the number of *effective* clusters for each method, which is defined as the number of active clusters assigned to at least 1 data atom across the entire dataset: $K_{\text{eff}} = |\{k : N_k(\hat{r}) \geq 1\}$. We see that even with over 100 initial clusters actively used after the first few steps, the updates of Alg. 3.1 enforce the rich-get-richer prior bias of the Dirichlet process prior and gradually favor only a small subset of the total number of initial clusters while letting the rest decay to zero usage. The original number of clusters is still *actively* represented in memory, just with some parameters set for zero usage.

Initializations with near the true number of clusters rarely do well. Using many more clusters is consistently better. Across all initializations, using a large number of initial clusters $K > 100$ relative to the true number of clusters $K = 26$ leads to much better ELBO scores than say $K = 40$ or $K = 60$. For an initialization with 30 clusters to do well at capturing the ground-truth trends, it must adequately represent all 26 true clusters. This requires a very lucky selection of examples for the initialization, even for the data-driven Bregman++ procedure. We see in practice that until we have over 100 clusters, we do not reliably recover the true clusters even in the best of 25 initializations. Using a large initialization hugely increases the chances that an initial cluster representing each of the true clusters will be selected.

The number of initial clusters matters more than the initialization procedure. Comparing the local optima plots (top row) for both random initialization and Bregman k-means++

initialization, we see the same dominant trend in both plots: using a large initial number of clusters leads to reliably good performance. Any differences between these procedures is of secondary importance.

Bregman k-means++ initialization shows modest improvement over random initialization. Consider the top row of Fig. 3.3, which shows the final training ELBO value found from each of 25 independent initializations across many different initial number of clusters. Looking closely at $K = 40$ or $K = 80$, we see that the Bregman++ initialization has much lower variance than the random example initialization, consistently yielding larger ELBO values. We might have expected a larger difference, but the data-driven method does indeed offer visibly better performance.

3.6.2 Clustering documents with multinomial likelihoods

We now consider the task of clustering data observations which represent entire text documents. Using the DP mixture model, we might hope to learn informative single-membership partitions of news articles which might be post-hoc interpreted as "political news" or "economics" or "film." Here, we take each observation x_n to be a *bag-of-words* vector. That is, $x_n \in \mathbb{R}^V$ is a vectors of non-negative integer word counts from a predefined vocabulary. We use a multinomial likelihood model with corresponding Dirichlet prior.

Toy Bars experiments

We first consider a "toy bars" experiment, inspired by Griffiths and Steyvers (2004). We arrange the 900-word vocabulary into a 30x30 grid, where each word represents one image pixel. Each cluster has a strong probability of producing words from one horizontal or vertical stripe or "bar" on this grid. There are 10 true bar clusters. We generate documents x_n directly from the finite mixture generative model, with roughly equal probability for each bar.

Fig. 3.4 shows the results of extensive local optima experiments on the task of clustering this synthetic dataset. Like the earlier experiments, we compare the impact of the number of initial clusters and the chosen initialization procedure. Our conclusions are:

For multinomial likelihoods, local optima prevent rich-get-richer prior bias from encouraging cluster usage decay. In Fig. 3.4 for zero-mean Gaussian likelihoods, we saw that large K initializations have many clusters with usage dropping to zero. In contrast, here in Fig. 3.4 we see that $K = 40$ initializations with 4x the number of true clusters consistently see *no* cluster that becomes zeroed out across many runs. Because the multinomial likelihood is very flexible, it can exactly match the relevant mean of any member document and thus discourage usage dropping to zero.

Too many clusters incur substantial penalty in ELBO. Results with $K = 40$ initial clusters doubtlessly include multiple copies of all 10 true bars. These runs achieve ELBO values which are

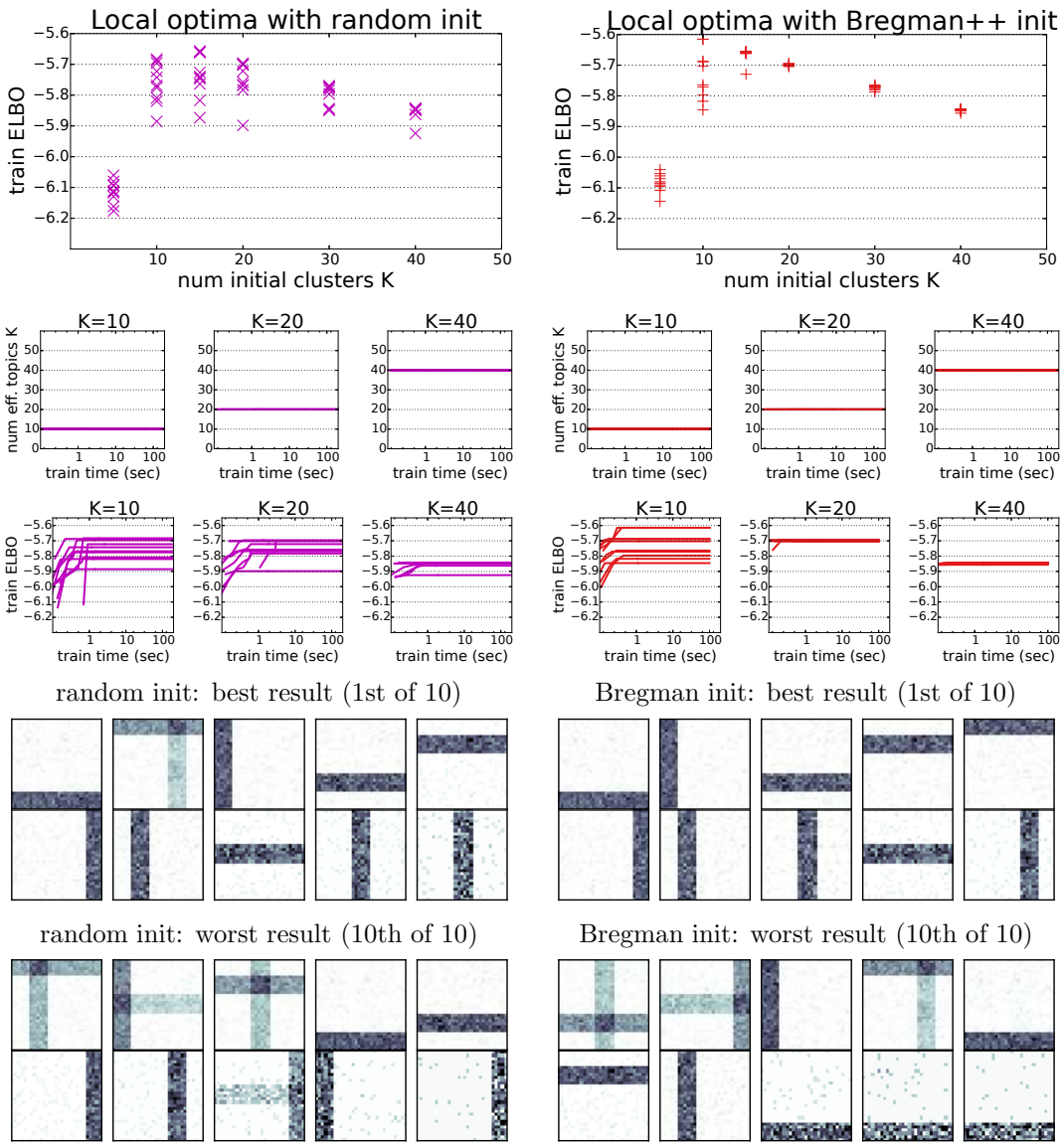

Figure 3.4: Local optima found when clustering toy bars data with DP mixture models. Dataset: 100 synthetic documents generated from 10 true bars topics via a true finite mixture model. We compare various initializations for the variational inference algorithm in Alg. 3.1 with the goal of exposing local optima problems. *Left column:* random initialization, which chooses K initial documents uniformly at random without replacement to initialize K clusters. *Right column:* Bregman k-means++ initialization, which chooses K initial documents as cluster centers via Alg. 2.2.

roughly half-way between the best score of the ideal $K = 10$ clusters and using only $K = 5$ clusters. Thus, the DP mixture prior leads to harshly penalizing redudant and irrelevant active clusters, even if the update steps of the algorithm cannot escape such local optima.

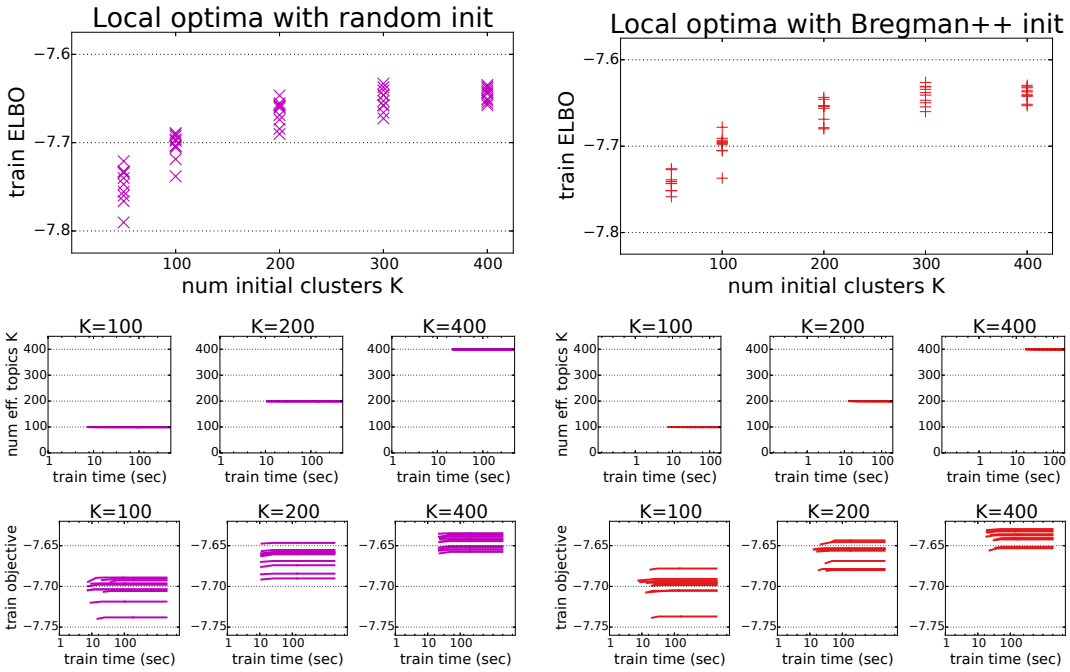

Figure 3.5: Local optima found when clustering NIPS articles with DP mixture models. Dataset: 1392 articles published in the proceedings of the Neural Information Processing Systems (NIPS) academic conference. We compare various initializations for the variational inference algorithm in Alg. 3.1 with the goal of exposing local optima problems. *Left column:* random initialization, which chooses K initial documents uniformly at random without replacement to initialize K clusters. *Right column:* slightly-better Bregman k-means++ initialization, which chooses K initial documents as cluster centers via Alg. 2.2.

Data-driven Bregman initialization yields slightly better results than random. Comparing the top row plots of Fig. 3.4 shows a noticeably reduced spread in ELBO performance for Bregman k-means++ initialization compared to random initialization across runs no matter what initial number of clusters is used.

NIPS experiments

Fig. 3.5 shows the results of extensive local optima experiments on the task of clustering articles from the proceedings of the Neural Information Processing Systems (NIPS) conference. This is a popular benchmark dataset among topic modelers. Our conclusions are:

For this dataset, hundreds of clusters seem useful. Across both initialization methods, the best scores are achieved with nearly 300 or 400 active clusters at initialization.

For multinomial likelihoods, local optima prevent rich-get-richer prior bias from encouraging cluster usage decay. Looking at the number of effective topics plots in the second

row of Fig. 3.5, we see that across 10 runs the number of topics assigned to at least one observation *never* decreases from its initial value. This confirms our earlier toy data experiment.

Data-driven Bregman++ initialization does not seem visibly better than random. Somewhat surprisingly, except when the number of clusters is very small ($K = 50$), the set of final ELBO values across multiple initializations does not seem to differ in spread or mean between the two methods.

3.7 Discussion

Our full-dataset variational algorithm from Alg. 3.1 and its two more scalable alternatives, stochastic (Alg. 3.2) and memoized (Alg. 3.3) represent promising training algorithms when given a reasonable initialization. However, our experiments here demonstrate that achieving good initializations can actually be surprisingly complicated and that as long as the number of clusters K remains fixed these algorithms are sorely limited in their ability to escape from a poor initialization. In the next section, we show that additional proposal moves that add and remove clusters during training lead to greatly improved performance.

Chapter 4

Proposal moves to escape local optima for DP mixture models

Both standard full-dataset variational inference and scalable variants like stochastic (SVI) and memoized (MVI) can get stuck in poor local optima, as shown in the experiments of Sec. 3.6. This can occur in part because each local or global step only updates a limited subset of the variational parameters. To escape the current basin of attraction for better regions of the parameter space, we need proposal moves that can change all free parameters *jointly*, making bigger changes than the coordinate ascent steps of local and global updates. In this chapter, we describe several possible proposals moves specifically for the Dirichlet process (DP) mixture model. In later chapters, we will discuss how these moves might change for more complex models.

We consider three possible proposal moves to escape local optima, some that remove clusters and others that add clusters. First, *merge* moves eliminate redundancy by combining two current clusters into a single combined cluster. Second, *birth* moves try to introduce new clusters in data-informed fashion to improve the explanation of the data. Finally, *delete* moves remove a single junk cluster by reassigning its mass across many remaining clusters. Each proposal move is executed by first transforming some original configuration of variational parameters into a candidate new configuration, and then deciding whether this new state improves the variational optimization objective $\mathcal{L}$. The correctness of these moves follows simply from the fact that they are not accepted unless the objective $\mathcal{L}$ improves.

Several previous efforts (Ueda et al., 2000; Bryant and Sudderth, 2012) employ similar proposal moves to escape local optima for specific Bayesian models within optimization-based algorithms. Ueda et al. developed split and merge moves for Baysian finite mixture models for an expectation-maximization algorithm (Ueda et al., 2000) as well as a full variational algorithm (Ueda and Ghahramani, 2002). Beal and Ghahramani (1999) provided a variational algorithm for training mixtures of factor analyzers which could add new clusters, albeit without any formal guarantees about improving the objective function. While interesting, these foundational efforts had

several limitations: they examined finite mixture models rather than Bayesian nonparametric models, they did not pursue scalable algorithms at all, and did not include anything like a delete proposal. In contrast, we make scalable construction and evaluation a priority, and also show that additional delete moves are required to escape some local optima.

Among stochastic variational methods, Bryant and Sudderth (2012) developed split and merge moves for stochastic variational inference of topic models. While these moves are somewhat effective, we find again that our delete move is crucially more powerful than these approaches. Furthermore, as shown later in our study of topic models in Ch. 5 using proposals with SVI can yield pathological uncontrolled growth in the number of clusters due to the lack of rigorous guarantees about improving the objective function $\mathcal{L}$. Our approach leads to much more reliable behavior.

Finally, a thread of research has used split and merge proposals within MCMC sampling training algorithms. Jain and Neal (2004) were the first to deploy split and merge moves, while recent work has seen many specialized proposals for DP mixtures (Chang and Fisher III, 2013) and HDP topic models (Chang and Fisher III, 2014). The requirement of maintaining detailed-balance within the Markov chain of posterior samples can often hinder the design of sampler-based split and merge moves for new models. Furthermore, these proposals are also difficult to scale to large datasets, though Chang and Fisher III (2014) have invested some laudable implementation effort in parallelization and other improvements.

To summarize, our goal is to develop proposal moves that can be constructed and evaluated using minimal computation in a batch-by-batch processing fashion, have a unified framework across several Bayesian nonparametric models, and preserve some guarantees about monotonic increase in the optimization objective $\mathcal{L}$ when possible. Preliminary versions of proposal moves for the DP mixture model were presented in our NIPS 2013 conference paper (Hughes and Sudderth, 2013). However, we have since made several improvements, such as formal guarantees of ELBO objective score improvement when a birth is accepted.

4.1 Merge moves for DP mixture models

The goal of the merge proposal is to create a candidate set of the variational free parameters $\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'$ which modifies the current state $\hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}$ by combining two chosen clusters k_A and k_B into a single cluster. As shown in Fig. 4.1, this can remove redundant clusters and lead to more compact models that are simpler to interpret and faster to train (because algorithm runtime scales linearly with K).

4.1.1 Merge proposal construction

Consider a current model with truncation level K. Let integers k_A and k_B be the labels of the pair of clusters we wish to try mergin. Formally, we have $1 \leq k_A < k_B \leq K$. We will assemble the proposed set of variational free parameters $\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'$ in two steps. First, we create $\hat{r}'$ via a deterministic merge step from the original responsibilities $\hat{r}$. This new set of clusters will only have $K-1$ clusters.

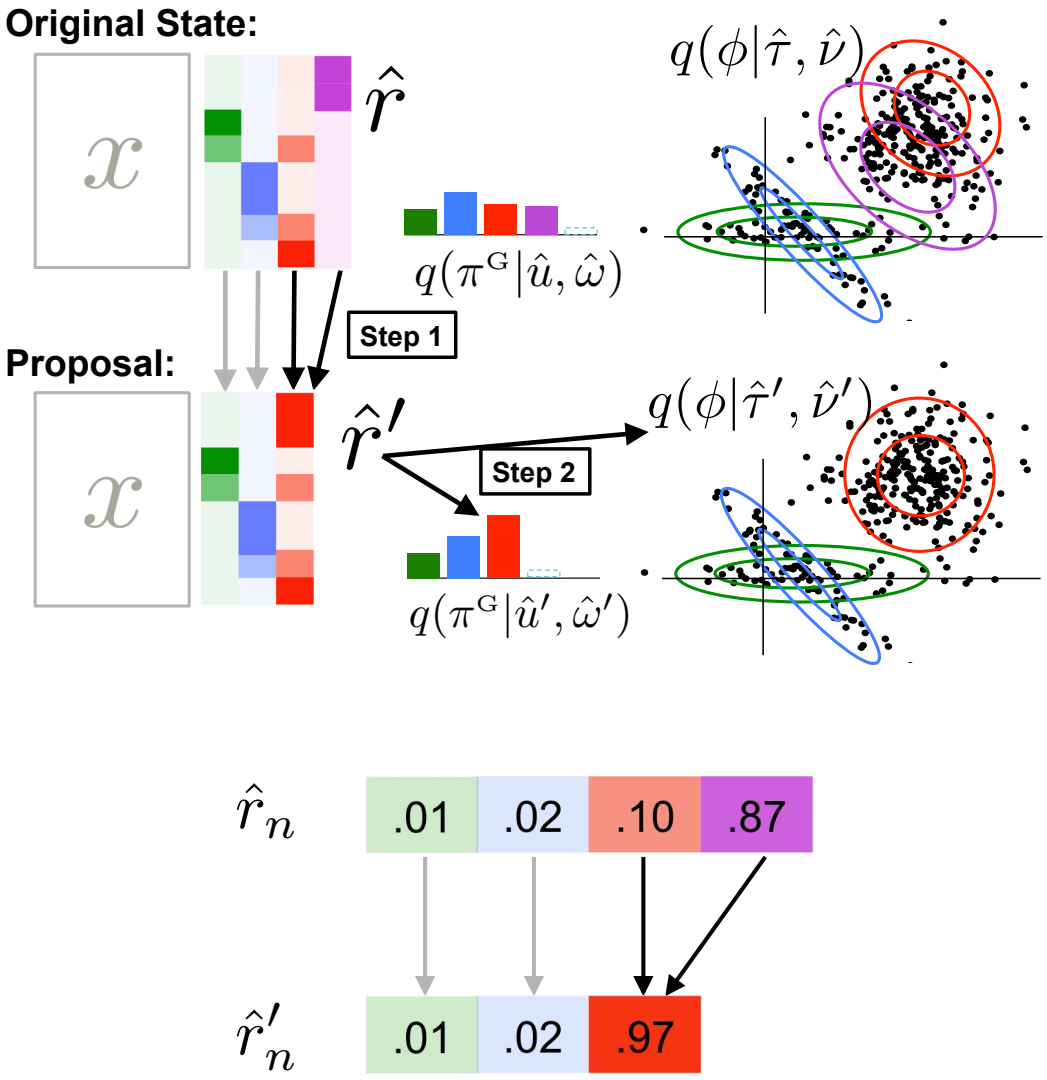

Figure 4.1: Illustration of merge proposal for DP mixtures.
Top: Holistic view of merge proposal construction. We start with an existing model with 4 active clusters. This original state is captured by the local assignment responsibilities $\hat{r}$, the cluster frequency parameters $\hat{u}, \hat{\omega}$, and the observation model parameters $\hat{\tau}, \hat{\nu}$. The first step is a deterministic construction of new assignment responsibilities $\hat{r}'$, where the values for red and purple clusters are combined. The second step is construction of new global parameters $\hat{\tau}', \hat{\nu}', \hat{u}', \hat{\omega}'$ given the responsibilities $\hat{r}'$. The complete model has only 3 active clusters. *Bottom detail:* Example deterministic construction of merged responsibilities. Any responsibility mass originally cluster labels k_A (red) and k_B (purple) is merged via addition into a single entry of the new responsbility vector.

Second, we use the closed-form global parameter updates to arrive at the corresponding global parameters. That is, we find the optimal allocation parameters $\hat{\eta}'$ via Eq. (3.43), and the optimal observation parameters $\hat{\tau}', \hat{\nu}'$ via Eq. (3.41).

Merge construction for local responsibilities

Our chosen rule for constructing candidate $\hat{r}'$ from the existing responsibilities $\hat{r}$ is simply to take any posterior mass in $q(z)$ assigned to cluster k_B and reassign this mass to the cluster k_A, while also retaining any existing mass on cluster k_A. This effectively combines these two clusters into a single cluster with label k_A. To deal with the removal of the cluster label k_B, we shift all larger cluster indices $k > k_B$ down by one. This leaves an effective set of new responsibilities $\hat{r}'$ with truncation level $K - 1$.

$$\text{for } k \in 1, 2, \dots K-1: \quad \hat{r}'_{nk} = \begin{cases} \hat{r}_{nk_A} + \hat{r}_{nk_B} & \text{if } k = k_A \\ \hat{r}_{nk} & \text{else if } k < k_B \\ \hat{r}_{nk+1} & \text{else if } k \geq k_B \end{cases} \tag{4.1}$$

Merge construction of summary statistics for local responsibilities

Given proposed responsibilities $\hat{r}'$ for $K - 1$ clusters, we need to compute the summary statistics $S(\hat{r}', x), N(\hat{r}')$ defined in Eq. (3.35). These statistics play a fundamental role in later global parameter updates and ELBO objective evaluation for our proposed configuration. Because these statistics are *linear* functions of the responsibilities $\hat{r}'$, they need not be computed directly from the new responsibilities $\hat{r}'$ via Eq. (3.35), which would require $O(NK)$ operations. Instead, they can be directly computed from the existing summary statistics $S(\hat{r}), N(\hat{r})$ for each of the K original clusters:

$$\text{for } k \in 1, 2, \dots K-1: \quad N_k(\hat{r}') = \begin{cases} N_{k_A}(\hat{r}) + N_{k_B}(\hat{r}) & \text{if } k = k_A \\ N_k(\hat{r}) & \text{else if } k < k_B \\ N_{k+1}(\hat{r}) & \text{else if } k \geq k_B \end{cases} \tag{4.2}$$

$$\text{for } k \in 1, 2, \dots K-1: \quad S'_k(\hat{r}', x) = \begin{cases} S_{k_A}(\hat{r}, x) + S_{k_B}(\hat{r}, x) & \text{if } k = k_A \\ S_k(\hat{r}, x) & \text{else if } k < k_B \\ S_{k+1}(\hat{r}, x) & \text{else if } k \geq k_B \end{cases}$$

This direct construction from existing summaries requires only one sum operation for each of $N(\hat{r}')$ and $S(\hat{r}', x)$, which is an $O(1)$ operation independent of the number of observations N. Thus, it is much more efficient than the $O(NK)$ cost of evaluating Eq. (3.35).

Merge construction of entropy statistics for local responsibilities

Unlike the statistics above, the entropy statistic $H_k(\hat{r}')$ defined in Eq. (3.29) is a *non-linear* function of $\hat{r}$. Thus, while some entries of the original entropy statistic vector $H(\hat{r})$ can be reused after appropriate index shifting, the entry corresponding to the newly-merged cluster k_A must be computed

anew from the corresponding column of $\hat{r}'$.

$$\text{for } k \in 1, 2, \dots K-1: \quad H_k(\hat{r}') = \begin{cases} \sum_{n=1}^{N} -\hat{r}'_{nk} \log \hat{r}'_{nk} & \text{if } k = k_A \\ H_k(\hat{r}) & \text{else if } k < k_B \\ H_{k+1}(\hat{r}) & \text{else if } k \geq k_B \end{cases} \tag{4.3}$$

Computing the exact value of $\mathcal{L}_{\text{entropy}}(\hat{r}')$ thus requires $O(N)$ work for each merge pair. We stress that this value is *not* needed to *create* a valid set of proposed free parameters $\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'$. However, it is needed to *evaluate* the entropy term $\mathcal{L}_{\text{entropy}}(\hat{r}')$ of the objective function $\mathcal{L}(\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}')$, which is used to decide if the proposed set of free parameters improves over the current set.

Merge construction of observation model global parameters

Given a concrete set of responsibilities $\hat{r}'$, or equivalently the summary statistics $N(\hat{r}')$ and $S(\hat{r}', x)$, we can obtain proposed values for the observation model global parameters $\hat{\tau}', \hat{\nu}'$ via the standard global update step in Eq. (3.41). This produces values $\hat{\tau}'_k, \hat{\nu}'_k$ for each cluster k in the set of the $K-1$ active clusters. By definition, this global step update solves the optimization problem of maximizing $\mathcal{L}_{\text{data}}(\hat{r}', \hat{\tau}', \hat{\nu}')$ given fixed $\hat{r}'$.

For original cluster labels k not involved in the merge – that is, $k \notin \{k_A, k_B\}$ – there will be new parameters equal to the original parameters $\hat{\tau}_k, \hat{\nu}_k$, though perhaps reindexed due to the label-shifting operation required to remove the cluster k_B. Thus, savvy implementations need not recompute these values and can instead just reindex the original parameters.

Merge construction of DP-allocation-model global parameters

For the DP mixture model, we can find the optimal global allocation parameters $\hat{\eta}'$ given $\hat{r}'$ via the update in Eq. (3.43). Any cluster with index $k < k_A$ in the original model will be unchanged after this update. However, all clusters $k \in [k_A, k_B]$ will have new values after this merge update. This is due to the dependence of the updates on the effective count statistic $N^{>}$, which after a merge may be computed as:

$$\text{for } k \in 1, 2, \dots K-1: \quad N^{>}_k(\hat{r}') = \begin{cases} \sum_{\ell=k+1}^{K} N_k(\hat{r}') & \text{if } k_A \leq k < k_B \\ N^{>}_k(\hat{r}) & \text{if } k < k_A \\ N^{>}_{k+1}(\hat{r}) & \text{if } k \geq k_B \end{cases} \tag{4.4}$$

4.1.2 Evalation of merge proposals

We represent the existing state of the DP mixture variational approximate posterior with K active clusters via the full set of free parameters $\hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}$. After a merge of k_A and k_B, we have a fully-constructed candidate state with $K-1$ active clusters: $\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'$. We decide whether to accept or reject this proposal by comparing the ELBO objective scores for both states. If our merge candidate state improves the ELBO score, we accept it. Otherwise, we reject it.

The acceptance score is defined as:

$$\begin{aligned}\mathcal{L}(x,\hat{r}',\hat{\eta}',\hat{\tau}',\hat{\nu}') - \mathcal{L}(x,\hat{r},\hat{\eta},\hat{\tau},\hat{\nu}) \triangleq\ & \mathcal{L}_{\text{data}}(x,\hat{r}',\hat{\tau}',\hat{\nu}') - \mathcal{L}_{\text{data}}(x,\hat{r},\hat{\tau},\hat{\nu}) \\ & + \mathcal{L}_{\text{entropy}}(\hat{r}') - \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{DP-alloc}}(\hat{r}',\hat{\eta}') - \mathcal{L}_{\text{DP-alloc}}(\hat{r},\hat{\eta})\end{aligned} \tag{4.5}$$

Below, we discuss crucial simplifications for each of the $\mathcal{L}_{\text{data}}$ term, the $\mathcal{L}_{\text{entropy}}$ term, and the $\mathcal{L}_{\text{DP-alloc}}$ that make rapid evaluation possible.

Simplification of data term. First, we evaluate the relative improvement in the $\mathcal{L}_{\text{data}}$ term. We assume that each state has constructed its global parameters optimally given the local responsibilities. That is, for each existing cluster k, we have $\hat{\tau}_k = S_k(\hat{r},x)+\bar{\tau}$ and $\hat{\nu}_k = N_k(\hat{r})+\bar{\nu}$ via Eq. (3.41). Likewise, $\hat{\tau}',\hat{\nu}'$ are functions of (statistics of) $\hat{r}'$. Then the slack-term simplifications of $\mathcal{L}_{\text{data}}$ kick in and we have

$$\begin{aligned}\mathcal{L}_{\text{data}}(x,\hat{r}',\hat{\tau}',\hat{\nu}') - \mathcal{L}_{\text{data}}(x,\hat{r},\hat{\tau},\hat{\nu}) &= c^P(\hat{\tau}'_{k_A},\hat{\nu}'_{k_A}) - c^P(\hat{\tau}_{k_A},\hat{\nu}_{k_A}) - c^P(\hat{\tau}_{k_B},\hat{\nu}_{k_B}) \\ &= c^P(S_{k_A}(\hat{r}',x)+\bar{\tau}, N_{k_A}(\hat{r}')+\bar{\nu}) \\ &\quad - c^P(S_{k_A}(\hat{r},x)+\bar{\tau}, N_{k_A}(\hat{r})+\bar{\nu})) \\ &\quad - c^P(S_{k_B}(\hat{r},x)+\bar{\tau}, N_{k_B}(\hat{r})+\bar{\nu}))\end{aligned} \tag{4.6}$$

Thus, accepting a merge sensibly requires only terms related to clusters involved in a merge. Any other existing cluster $k \notin \{k_A, k_B\}$ has no impact on the acceptance decision.

Simplification of entropy term. Assuming that $\hat{r}'$ was constructed from $\hat{r}$ via Eq. (4.1), and consequently $H(\hat{r}')$ is defined as in Eq. (4.3), we have:

$$\begin{aligned}\mathcal{L}_{\text{entropy}}(\hat{r}') - \mathcal{L}_{\text{entropy}}(\hat{r}) &= \sum_{k=1}^{K-1} H_k(\hat{r}') - \sum_{k=1}^{K} H_k(\hat{r}) \\ &= H_{k_A}(\hat{r}') - H_{k_A}(\hat{r}) - H_{k_B}(\hat{r})\end{aligned} \tag{4.7}$$

Again, this sensibly only involves terms related to the pair of clusters k_A, k_B being merged. This simplified difference-of-entropies will always be *negative* for any merge pair, because by definition $H_{k_A}(\hat{r}') \leq H_{k_A}(\hat{r}) - H_{k_B}(\hat{r})$.

Simplification of DP-allocation term. Assuming that both the current parameters $\hat{\eta}$ and proposed parameters $\hat{\eta}'$ were computed by applying the update of Eq. (3.43) using the correponding counts $N(\hat{r})$ and $N(\hat{r}')$, we can find similar simplifications for the allocation model objective, which is given by

$$\begin{aligned}\mathcal{L}_{\text{DP-alloc}}(\hat{r}',\hat{\eta}') - \mathcal{L}_{\text{DP-alloc}}(\hat{r},\hat{\eta}) = &\sum_{k=1}^{K-1} \Big(c_{\text{Beta}}(\hat{\eta}'_{k1},\hat{\eta}'_{k0}) - c_{\text{Beta}}(1,\gamma) \Big) \\ &- \sum_{k=1}^{K} \Big(c_{\text{Beta}}(\hat{\eta}_{k1},\hat{\eta}_{k0}) - c_{\text{Beta}}(1,\gamma) \Big)\end{aligned} \tag{4.8}$$

After substituting in Eq. (4.4) and simplifying, we have:

$$\begin{aligned}\mathcal{L}_{\text{DP-alloc}}(\hat{r}', \hat{\eta}') - \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) &= c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{\eta}_{k_B 1}, \hat{\eta}_{k_B 0}) \\ &\quad + \sum_{k=k_A}^{k_B-1} c_{\text{Beta}}(\hat{\eta}'_{k1}, \hat{\eta}'_{k0}) - c_{\text{Beta}}(\hat{\eta}_{k1}, \hat{\eta}_{k0}) \\ &= c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{\eta}_{k_B 1}, \hat{\eta}_{k_B 0}) \\ &\quad + c_{\text{Beta}}(\hat{\eta}_{k_A 1} + N_{k_B}, \hat{\eta}_{k_A 0} - N_{k_B}) - c_{\text{Beta}}(\hat{\eta}_{k_A 1}, \hat{\eta}_{k_A 0}) \\ &\quad + \sum_{k=k_A+1}^{k_B-1} c_{\text{Beta}}(\hat{\eta}_{k1}, \hat{\eta}_{k0} - N_{k_B}) - c_{\text{Beta}}(\hat{\eta}_{k1}, \hat{\eta}_{k0})\end{aligned} \tag{4.9}$$

Here, this involves not only the clusters k_A, k_B directly involved in the merge, but all clusters with indices between k_A and k_B. This is due to the adjusted value of $N^{>}$ at those in-between indices in Eq. (4.4).

4.1.3 Scalable construction and evaluation via memoized statistics

For large datasets, the simplified decision rules above suggest that we need not directly instantiate the whole $N \times K - 1$ responsibilities matrix $\hat{r}'$ to accept or reject a candidate merge of clusters k_A and k_B. Instead, in the next pass through the full dataset we need only track the typical whole-dataset summary statistics N^G, S^G, H^G of our memoized algorithm in Alg. 3.3 as well as the merged entropy scalar $H_{k_A}(\hat{r}')$. From these statistics alone, we can directly create all candidate global summaries N'^G, S'^G, H'^G for the merged configuration.

Thus, before a full pass through the dataset, we only need to decide on a pair k_A, k_B to consider (see Sec. 4.1.4 for this decision process). Then, we proceed with normal memoized updates at each batch, with the one additional requirement of computing $H_{b,k_A}(\hat{r}') = \sum_{n \in \mathcal{D}_b} (\hat{r}_{nk_A} + \hat{r}_{nk_B}) \log(\hat{r}_{nk_A} + \hat{r}_{nk_B})$. That one extra scalar at each batch is all that is needed beyond the original fixed-truncation algorithm. After the full pass, we can then use the simplified acceptance rules from the previous section to decide whether to accept or reject the merge.

We emphasize that these simplified construction rules apply because both statistics N, S are linear functions of $\hat{r}$. Given current assignment summaries N^G, S^G, we can always construct a candidate observation model $N'^G, S'^G, \hat{\nu}', \hat{\tau}'$ for any valid merge pair and evaluate this candidate's objective score in Eq. (4.6) in time independent of the dataset size and without any additional local inference steps.

Accepting multiple merge proposals

Suppose that before the start of a lap we have a list of M candidate merge pairs: $\{k_{mA}, k_{mB}\}_{m=1}^{M}$, where for each pair indexed by m we have $k_{mA} < k_{mB}$. Then, during a pass through each batch b of a dataset, we compute the merge entropy $H_{bm}(\hat{r}')$, and aggregate these to a global quantity $H_m^G(\hat{r}')$ for each pair of clusters m. After visiting all batches, we can then sequentially evaluate each of the M possible merge pairs one-by-one, accepting or rejecting each according to whether it improves the

objective $\mathcal{L}$ and updating the "existing" set of whole-dataset sufficient statistics N^G, S^G, H^G after every acceptance.

The above greedy evaluation procedure allows us to accept many merge proposals after a single pass through the dataset, so long as each accepted pair involves a distinct set of clusters. That is, if the first pair ($m = 1$) with cluster indices $(1, 2)$ was accepted already, then the pair $(1, 3)$ appearing later in the list of possible candidates must be rejected without evaluation. This is because its entropy term $H_m(\hat{r}')$ was computed using an old value of responsibilities for cluster 1 that does not include the recently accepted merge of cluster 1 and 2. Again, it is only the non-linear entropy term that creates this complication. Overall, we are free to consider many candidate pairs during a single pass of the data, but each original cluster k can appear in at most one of the (shorter) list of accepted pairs. This still allows up to $K/2$ acceptances in every pass through the dataset.

4.1.4 Selecting a pair of clusters to try merging

Even with the simplified evaluation terms above, computing the entropy terms for all $\frac{K(K-1)}{2}$ possible merge pairs would have runtime cost of $O(NK^2)$, which is prohibitively expensive. Instead, we recommend a *selection* step at the beginning of each lap through the dataset which identifies at most M candidate pairs to consider, where M is specified by the user as a maximum budget.

To rank possible candidates, we suggest using a score function based on the subset of the simplified acceptance criteria:

$$\begin{aligned} \text{m-score}(k_A, k_B) = \mathcal{L}_{\text{data}}(x, \hat{r}', \hat{\tau}', \hat{\nu}') - \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}), \\ + \mathcal{L}_{\text{DP-alloc}}(\hat{r}', \hat{\eta}') - \mathcal{L}_{\text{DP-alloc}}(\hat{r}, \hat{\eta}) \end{aligned} \tag{4.10}$$

where we create the candidate local parameters $\hat{r}'$ by merging clusters k_A, k_B and we compute these terms via their simplified forms in Eq. (4.6) and Eq. (4.8). These terms can be evaluated exactly from the existing global summaries N^G, S^G and parameters $\hat{\tau}, \hat{\nu}, \hat{\eta}$. Computing the difference of $\mathcal{L}_{\text{DP-alloc}}$ terms has cost which is $O(K)$ at worst, when $k_A = 1$ and $k_B = K$. Computing the difference of $\mathcal{L}_{\text{data}}$ terms always has cost independent of K and N.

For the DP mixture model, the value of the m-score for a given pair k_A, k_B is a strict lower bound of the actual difference of $\mathcal{L}$ values used to accept or reject the merge pair. Any pair with m-score below zero will certainly be rejected once the difference-of-$\mathcal{L}_{\text{entropy}}$ terms are accounted for, because this term is always negative. Thus, at the end of the selection step we retain only set of candidate cluster pairs k_A, k_B with positive scores. This is the set of candidate merge pairs we track during all local steps in the current lap.

4.2 Birth moves

The goal of a birth proposal is to create a candidate set of variational parameters $\hat{r}', \hat{\tau}', \hat{\nu}', \hat{\eta}'$ which modifies the current state $\hat{r}, \hat{\tau}, \hat{\nu}, \hat{\eta}$ by splitting an existing cluster j into *two or more* new clusters.

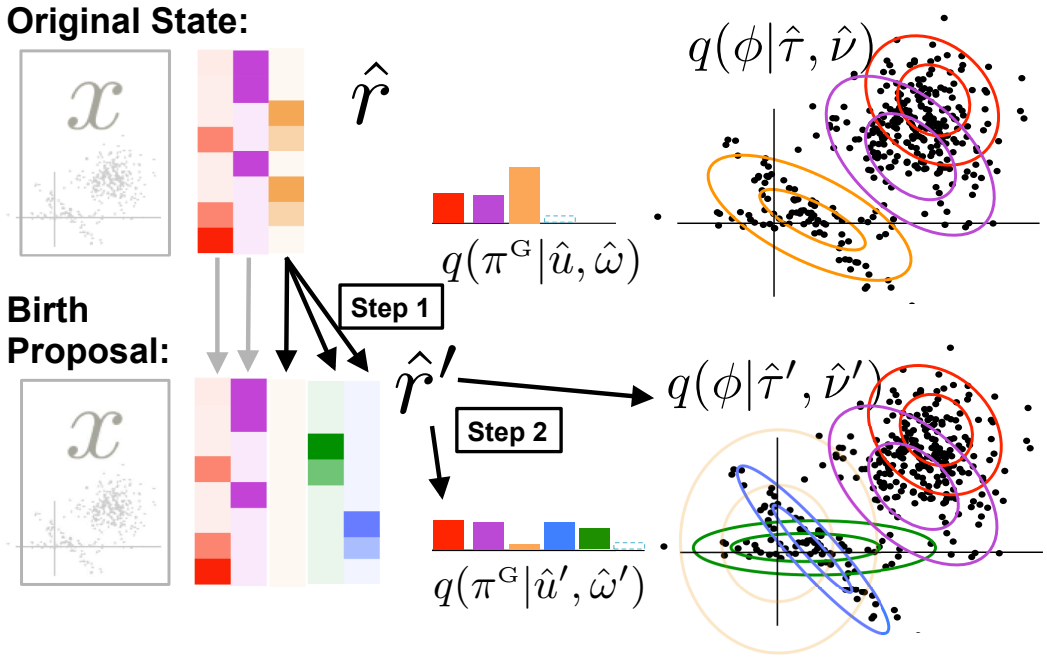

Figure 4.2: Illustration of birth proposal for DP mixtures.
Holistic view of birth proposal construction. We start with an existing model with 3 active clusters. This original state is captured by the local assignment responsibilities $\hat{r}$, the cluster frequency parameters $\hat{u}, \hat{\omega}$, and the observation model parameters $\hat{\tau}, \hat{\nu}$. We select the orange cluster to target with a birth proposal. The first step is the construction of new assignment responsibilities $\hat{r}'$ where an mass previously assigned to orange is now distributed across several new clusters, shown here in blue and green. The second step is construction of new global parameters $\hat{\tau}', \hat{\nu}', \hat{u}', \hat{\omega}'$ given the responsibilities $\hat{r}'$. The complete proposal now has 5 active clusters, including the original, now-empty orange cluster. This could be discarded immediately, but we show it here for later connections to multi-batch proposals.

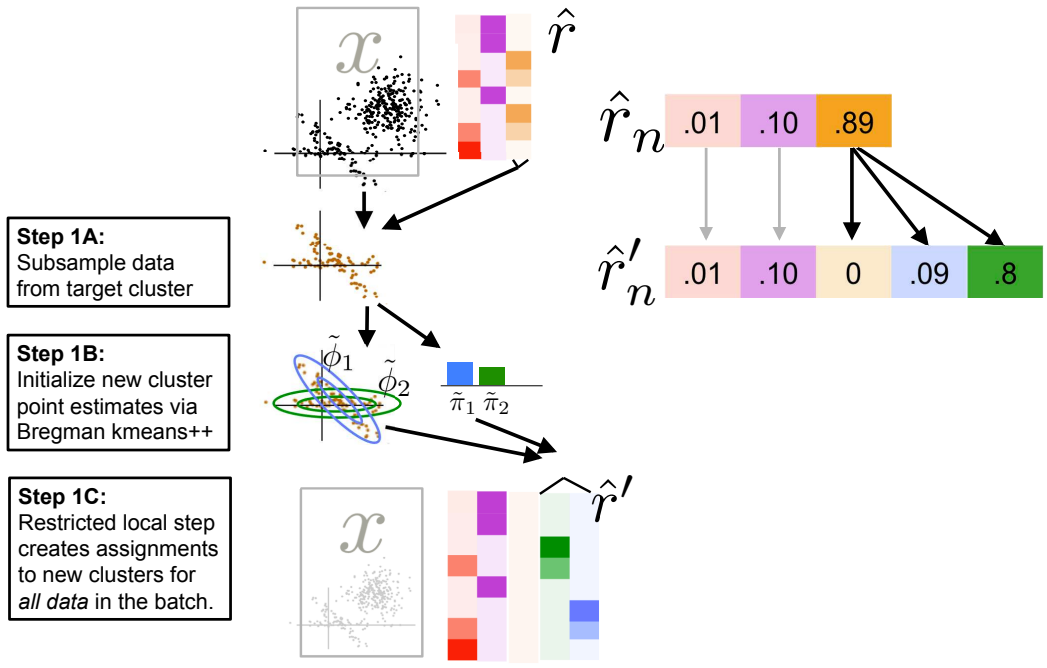

Figure 4.3: Illustration of local responsibility construction under birth proposal
Details of Step 1 from Fig. 4.2. *Step 1A:* Given the original responsibilities and a chosen cluster (orange) to target, we first create a subsampled dataset. *Step 1B:* We use the Bregman k-means algorithm from Alg. 2.1, with distance-biased initialization from Alg. 2.2, to create point estimates for the global parameters of each new cluster, represented by $\tilde{\phi}$ and $\tilde{\pi}$. Variables denoted with $\tilde{}$ have dimension corresponding to the number of newborn clusters. *Step 1C:* We perform a restricted local step to obtain proposed responsibilities $\hat{r}'$. There is one $\hat{r}'_n$ vector for each data atom n in the original dataset.

That is, we take a model with active truncation level K, and produce a model with truncation level $K + J'$, where $J' \geq 2$ is the total number of new clusters created by the birth proposal.

Like the merge move, a birth move proceeds in three steps. First, we create candidate responsibilities $\hat{r}'$ from the existing set $\hat{r}$. This leads immediately to the corresponding summary statistics $N(\hat{r}'), S(\hat{r}'), H(\hat{r}')$. Second, we create the associated global parameters $\hat{\tau}', \hat{\nu}', \hat{\eta}'$ via the global update step, so they achieve optimal values under the proposed responsibilities $\hat{r}'$. Finally, we decide to accept or reject the proposed approximate posterior, based on whether it improves the overall objective.

We first discuss these steps in detail for the simpler case of a single-batch dataset. We later discuss handling proposals for large-scale, multi-batch memoized evaluation.

4.2.1 Birth proposal construction

To create a birth proposal under our DP mixture model, we require as input a chosen active cluster index k_{target} satisfying $1 \leq k_{\text{target}} \leq K$. Index k_{target} specifies our *target* cluster, which we will propose breaking up into two or more subclusters. In this section, we discuss how to create the candidate parameters given this choice. Selection of k_{target} is described later in Sec. 4.3.4.

Given the chosen cluster k_{target}, we need to create local responsibilities $\hat{r}'$ and global parameters. We first construct the responsibilities $\hat{r}'$ for the dataset under this proposal. This construction is done in two stages: we first create temporary global parameters for the assigned clusters, and then we use these temporary parameters to construct $\hat{r}'$. Later, we create the proposed values $\hat{\eta}', \hat{\tau}', \hat{\nu}'$ of the global parameters via a global optimization step $\hat{r}'$.

Birth construction of initial global clusters

Given the whole dataset x and current responsibilities $\hat{r}$, we identify a *targeted subset* x' which is primarily assigned to cluster k_{target} by selecting those atoms with responsibility values over a threshold. That is, $x' = \{x_n : \hat{r}_{nk_{\text{target}}} > \epsilon\}$, where $\epsilon \approx 0.1$ in practical implementations. Using this subset, we can quickly identify a set of J' initial cluster shape parameters by running the InitClusterMeansViaBregmanSamples algorithm from Alg. 2.2. This yields a set of mean parameters $\{\mu'_j\}_{j=1}^{J'}$. Further iterations of the Bregman k-means algorithm in Alg. 2.1 from this initialization produces a set of point estimates μ', π' which are likely to explain the target dataset x' well. The cost of each iteration of this creation step is linear in the size of the *target* dataset, not the whole dataset. The cost is also linear in the new cluster label space $\mathbb{J} = \{1, 2, \ldots J'\}$, which could be much smaller than the original label space of all K active clusters.

Using these point estimates, we have defined a finite mixture model over the newborn clusters in $\mathbb{J}$ which has a point-estimate for the cluster frequency parameter $\tilde{\pi}$ (a vector of size J' that sums-to-one) and a cluster mean parameter $\tilde{\mu}_j$ for each newborn cluster. Let $\tilde{N}_j$

We can further combine this model over the newborn clusters with our current model over the existing clusters to create a candidate expanded model with a point estimated frequency probability

π_k for each cluster $k \in 1, 2, \dots K + J'$.

$$\pi_k = \begin{cases} \mathbb{E}_q[\pi_k] & \text{if } k \leq K, k \neq k_{\text{target}} \\ 0 & \text{if } k = k_{\text{target}} \\ \mathbb{E}_q[\pi_{k_{\text{target}}}]\tilde{\pi}_{(k-K)} & \text{if } k > K \end{cases} \tag{4.11}$$

Birth construction of local responsibilities

We now need to use these new clusters to form responsibility parameters $\hat{r}'$ for all clusters (including the original K clusters and the J' new clusters) which are consistent with the whole dataset. As illustrated in Fig. 4.2, we will do this by conceptually reassigning probability mass from the original target cluster k_{target} to the new clusters at indices $\{K+1, K+2, \dots K+J'$, while leaving all mass at other non-target clusters alone. This mass transfer operation for creating each atom's responsibility vector $\hat{r}'_n$ from $\hat{r}_n$ obeys the original constraints on this vector: it must be non-negative and sum-to-one.

Under these constraints, we can effectively reparameterize the full vector $\hat{r}'_n$ into a deterministic function of the original vector $\hat{r}_n$ and a smaller vector $\tilde{p}_n = [\tilde{p}_{n1} \dots \tilde{p}_{nJ}]$ of length J'. We interpret each entry $\tilde{p}_{nj}$ as the fraction of the original mass on cluster k_{target} that is transfered to new cluster j. This mass transfer vector is non-negative and sums-to-one. Using this reparameterization, we have:

$$\text{for } k \in 1, 2, \dots K, K+1, \dots K+J' : \quad \hat{r}'_{nk} = \begin{cases} \hat{r}_{nk} & \text{if } k \leq K, k \neq k_{\text{target}} \\ 0 & \text{if } k = k_{\text{target}} \\ \hat{r}_{nk_{\text{target}}}\tilde{p}_{n,(k-K)} & \text{if } k > K \end{cases} \tag{4.12}$$

This construction by definition satisfies the desired sum-to-one constraints on $\hat{r}'_n$.

Restricted local step. The creation step above provides concrete values for the initial global parameter estimates $\{\tilde{\tau}_j, \tilde{\nu}_j$, and $\tilde{\pi}_j\}$ for the set of new clusters labels $\mathbb{J}$. Given these, we'd like to find optimal responsibilities $\hat{r}'_n$ for each data atom n under our local assignment objective from Eq. (3.46). As a function of $\hat{r}'_n$, the objective is:

$$\mathcal{L}_n(x_n, \hat{r}'_n) = \sum_{k=1} \hat{r}'_{nk} W_{nk} - \hat{r}'_{nk} \log \hat{r}'_{nk} \tag{4.13}$$

which we can decompose into a sum over the original clusters and a sum over the newborn clusters in label space $\mathbb{J}$:

$$= \sum_{k=1}^{K} \hat{r}_{nk} W_{nk} - \hat{r}_{nk} \log \hat{r}_{nk} + \sum_{j=1}^{J} \hat{r}_{nk_{\text{target}}} \tilde{p}_{nj} \tilde{W}_{nj} - \hat{r}_{nk_{\text{target}}} \tilde{p}_{nj} \log[\hat{r}_{nk_{\text{target}}} \tilde{p}_{nj}] \tag{4.14}$$

where the posterior weights for new clusters are given by:

$$\tilde{W}_{nj}(x_n, \tilde{\tau}, \tilde{\nu}, \tilde{\pi}) = \mathbb{E}_{q(\phi_j|\tilde{\nu}_j, \tilde{\tau}_j)}[\log p(x_n|\phi_j)] + \log \tilde{\pi}_j \tag{4.15}$$

which as a function of the reassignment probabilities p'_n simplifies to

$$\mathcal{L}_n(x_n, \hat{r}_n, p'_n) = \hat{r}_{nk_{\text{target}}} \Big[\sum_{j=1}^{J} \tilde{p}_{nj} \tilde{W}_{nj} - \tilde{p}_{nj} \log \tilde{p}_{nj} \Big] \tag{4.16}$$

Given the posterior weights $\tilde{W}$ for the new clusters, we can find the optimal reassignment probabilities by using $\tilde{p}^*_n = \textsc{RespFromWeights}(\tilde{W}_n)$. This delivers a vector of size J' that sums-to-one, as required by our constraints.

Thus, we may compute the optimal reassignment probabilities via a version of the local update step from Eq. (3.48) which is *restricted* to the new label space $\mathbb{J}$. This restricted local step will be much more affordable in general than computing $\hat{r}'$ for many new clusters.

The runtime cost of the restricted local step scales linearly with the number of new states J' and remains independent of the total number of states K. This keeps proposal construction manageable even for large numbers of clusters. The cost is, however, linear in the total number of atoms N, which may be slow. To make costs more manageable, we can focus on the subset of atoms with significant mass on the targeted cluster k_{target}: $\mathcal{D}_n = \{n : \hat{r}_{nk_{\text{target}}} \geq \epsilon\}$. We may perform restricted local steps only on these atoms, and for any others use a deterministic rule where $\tilde{p}_n$ is set to an indicator vector for the newborn cluster with maximum weight, or the closest cluster in the newborn set.

Birth construction of sufficient statistics

After computing the reassignment vector $\tilde{p}_n$ for each new cluster, we can immediately compute the effective count statistic N'_k at each cluster index in the proposed model $k \in \{1, 2, \dots K + J'\}$:

$$N_k(\hat{r}') = \begin{cases} N_k(\hat{r}) & \text{if } k \leq K, k \neq k_{\text{target}} \\ 0 & \text{if } k = k_{\text{target}} \\ \sum_{n=1}^{N} \hat{r}_{nk_{\text{target}}} \tilde{p}_{n(k-K)} & \text{if } k > K \end{cases} \tag{4.17}$$

Similarly, we compute the cluster shape sufficient statistic S'_k at each cluster index in the proposed model $k \in \{1, 2, \dots K + J'\}$:

$$S_k(\hat{r}', x) = \begin{cases} S_k(\hat{r}, x) & \text{if } k \leq K, k \neq k_{\text{target}} \\ 0 & \text{if } k = k_{\text{target}} \\ \sum_{n=1}^{N} \hat{r}_{nk_{\text{target}}} \tilde{p}_{n(k-K)} s(x_n) & \text{if } k > K \end{cases} \tag{4.18}$$

Finally, for the entropy statistic at each cluster index $k \in \{1, 2, \dots K + J'\}$:

$$H_k(\hat{r}') = \begin{cases} H_k(\hat{r}) & \text{if } k \leq K, k \neq k_{\text{target}} \\ 0 & \text{if } k = k_{\text{target}} \\ \sum_{n=1}^{N} \hat{r}_{nk_{\text{target}}} \tilde{p}_{n(k-K)} \log[\hat{r}_{nk_{\text{target}}} \tilde{p}_{n(k-K)}] & \text{if } k > K \end{cases} \tag{4.19}$$

Birth construction of global parameters

After the above restricted local step, we have a candidate set of responsibilities $\hat{r}'$ for the *expanded* model with $K + J'$ clusters, as well as the appropriate sufficient statistics N^G, S^G, H^G summarizing these assignments.

The next step is to obtain corresponding *global* free parameters for our DP mixture model. That is, we need observation model parameters $\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^{K+J'}$ for each of the K original clusters *and* J' new clusters. We also need allocation parameters $\hat{\eta}_k$ for this expanded set of clusters.

Birth proposal global step for observation model. Eq. (3.41) gives a closed-form expression for the parameters $\hat{\tau}'_k, \hat{\nu}'_k$ given the summary statistics $N^G_k(\hat{r}'), S^G_k(\hat{r}')$. We need to do this explicitly only for the newborn clusters at indices $K+1, \ldots K+J'$. Other indices will have the same global parameters as before, so for these values we can simply copy the original state: for $k \leq K, k \neq k_{\text{target}}$, we have $\hat{\nu}'_k = \hat{\nu}_k$ and $\hat{\tau}'_k = \hat{\tau}_k$. Finally, our proposal assigns zero mass to the target cluster, so its optimal global parameters will match prior hyperparameters: $\hat{\nu}'_{k_{\text{target}}} = \bar{\nu}$ and $\hat{\tau}'_{k_{\text{target}}} = \bar{\tau}$.

Birth proposal global step for allocation model. Eq. (3.43) gives a closed-form update for the parameters $\hat{\eta}'$ given fixed $\hat{r}'$. Like the merge proposal, this construction will change not only the target cluster and the newborn clusters, but any non-target cluster in the set $\{k_{\text{target}}+1, k_{\text{target}}+2, \ldots K\}$. This is due to inserting the newborn cluster indices after the original set of cluster labels in stick-breaking order.

4.2.2 Birth proposal evaluation

As with merge proposals, we accept a candidate state $\hat{r}', \hat{\eta}', \hat{\tau}', \hat{\nu}'$ using the decision score in Eq. (4.5). Several terms like the differences of $\mathcal{L}_{\text{data}}$ terms and $\mathcal{L}_{\text{DP-alloc}}$ terms can be simplified using similar analysis as in the derivation of the corresponding merge terms. The $\mathcal{L}_{\text{data}}$ term simplifies to:

$$\begin{aligned}\mathcal{L}_{\text{data}}(\hat{r}', \hat{\tau}', \hat{\nu}') - \mathcal{L}_{\text{data}}(\hat{r}, \hat{\tau}, \hat{\nu}) &= \sum_{k=1}^{K+J'} \left(c^P(\hat{\tau}'_k, \hat{\nu}'_k) - c^P(\bar{\tau}, \bar{\nu})\right) - \sum_{k=1}^{K} \left(c^P(\hat{\tau}_k, \hat{\nu}_k) - c^P(\bar{\tau}, \bar{\nu})\right) \quad (4.20)\\ &= c^P(\bar{\tau}, \bar{\nu}) - c^P(\hat{\tau}_{k_{\text{target}}}, \hat{\nu}_{k_{\text{target}}}) + \sum_{j=1}^{J'} \left(c^P(\hat{\tau}'_{K+j}, \hat{\nu}'_{K+j}) - c^P(\bar{\tau}, \bar{\nu})\right)\end{aligned}$$

This can be interpreted as a log ratios of cluster-specific marginal likelihoods, which is sensible given that our overall objective $\mathcal{L}$ is interpreted as a lower bound of marginal likelihood.

4.2.3 Construction and evaluation via memoized statistics

When the full-dataset is divided into many batches $B > 1$, we do not want to wait until visiting all batches before accepting a potential birth proposal. Accepting a proposal as soon as possible, even after trying it on only one of the B batches, will reduce extra computation and propagate useful changes quickly. However, this introduces some complications in the memoized tracking of

batch-specific summaries. After an accepted birth proposal at batch b that adds J' new clusters, the new batch-specific summaries N_b, S_b, H_b will have larger truncation level $K + J'$, while other batches remain with only K active clusters.

We need to be sure we can track whole-dataset summaries that exactly summarize all visited batches even if they have different truncation levels. For the DP mixture model, our nested truncation means that at every batch b we can always *insert empty clusters* to the assignment statistics without changing the optimization objective function score $\mathcal{L}$. Thus, after a birth of J' new clusters at some batch is accepted, we can effectively update every other batch b by inserting J' zeros to each summary vector. For example, for the new count statistic N'_b we have:

$$N'_b(\hat{r}) = [N_{b1}(\hat{r}) \; N_{b2}(\hat{r}) \; \dots N_{bK}(\hat{r}) \; 0, 0 \; , \dots 0] \tag{4.21}$$

Inserting zeros allows on-demand expansion of any batch-specific summary vector from original trunction K to any desired larger truncation $K + J'$ where only the first K clusters are assigned to data.

Using these zero-expanded batch-specific statistics, we can then aggregate whole-dataset statistics N^G, S^G which accurately represent the entire dataset's assignment to the active set of $K+J'$ clusters. After a birth at batch b' only, we can create the new candidate value N'^G for the whole dataset from the original values N^G via:

$$N'^G_k = \begin{cases} N^G_k - N_{b'k_{\text{target}}} & \text{if } k = k_{\text{target}} \\ N^G_k & \text{else if } k \leq K \\ N'_{b'k} & \text{else if } k > K \end{cases} \tag{4.22}$$

Similar incremental updates can be used for whole-dataset cluster shape summaries S'^G and entropies H'^G.

Importantly, we can compute the whole-dataset statistics without requiring an expansion at every batch first. We can thus perform expansion at other batches *lazily* after an accepted birth, waiting until the next visit to batch b before applying Eq. (4.21) to its cached values of N_b, S_b, H_b.

4.2.4 Selecting clusters to target with birth proposals

We could attempt a birth at every cluster, but this would be expensive. To save time, we often wish to prioritize some clusters before others and perhaps limit ourselves to a fixed budget of the number of births we can attempt at each batch.

When visiting each batch, we need to determine which cluster $k \in \{1, 2, \dots K\}$ to target with a birth move, if any. We suggest an approach that accounts for two distinct cues: (1) how well each cluster represents or fits its assigned data, and (2) how well each cluster has performed in past birth moves. To quantify the model-fit criterion, we measure the following score function at each cluster

k:

$$\text{b-score}(k) = \frac{1}{N_k}\mathcal{L}_{\text{data}}(N, S, \hat{\nu}, \hat{\tau}, k) \tag{4.23}$$

$$= \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\mathbb{E}[\log p(x_n|\phi_k)] \tag{4.24}$$

Intuitively, this score is an average log probability. Large-magnitude negative values indicate clusters whose assigned data are poorly predicted on average. At every batch, we greedily choose to target the clusters with the lowest measured b-score value.

In addition to modeling quality, we also track for each distinct cluster how often we have tried a birth in the past, and how often these tries have succeeded. We always prioritize births that have never failed before, and only try a cluster that has failed before if (a) the cluster has changed considerably since its last failure, (b) the current batch has more atoms assigned to the target than our last birht attempt, or (c) all clusters have at least one failure, so we must choose a last-resort option.

Finally, when choosing a cluster to target for birth moves, we also account for several practical constraints. For example, we always skip any cluster that does not have adequate size (as measured by the summary N_{bk}) in the current batch, because we cannot learn a refined clustering from only a few data atoms. Typically, this threshold is very small, on the order of 3-50. Furthermore, we always skip any cluster that is currently a target for merge or delete moves. Only one type of proposal move can be accepted at each cluster in a given lap.

Trying multiple births at each batch. Under our proposed construction of candidate states, a birth targeting cluster k never impacts the assignments of a non-involved original cluster $j \neq k$. We may thus consider creating and evaluating multiple birth moves at once, so long as each targets a distinct cluster.

4.3 Delete moves

The goal of a delete proposal move is to create a candidate set of variational parameters $\hat{r}', \hat{\tau}', \hat{\nu}', \hat{\eta}'$ with one fewer cluster than the current state. Merge moves can remove a cluster only by reassigning its mass to one other existing state. Deletes are designed to be more flexible, because they can redistribute the target cluster's assignments across *many* other clusters. However, this flexibility comes with an increased runtime cost, since it is more expensive to infer how to redistribute among multiple clusters than to use a deterministic formula to reassign responsibility mass.

A delete proposal begins by selecting a single cluster index, denoted k_{target}, to target for deletion. Without loss of generality, we will assume that the target cluster is last in stick-breaking order: $k_{\text{target}} = K$. This allows the indices of remaining clusters to be the same in the current and proposed states.

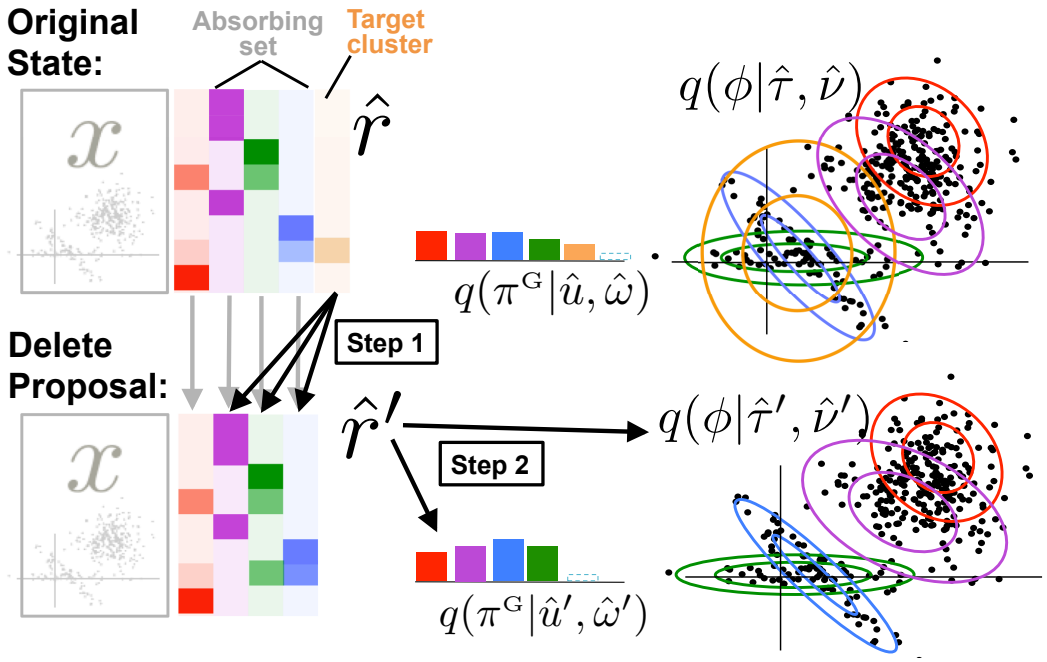

Figure 4.4: Illustration of delete proposal for DP mixtures.
Holistic view of delete proposal construction of candidate state. We start with an existing model with 6 active clusters in the original state, which is defined by the local assignment responsibilities $\hat{r}$, the cluster frequency parameters $\hat{u}, \hat{\omega}$, and the observation model parameters $\hat{\tau}, \hat{\nu}$. We select the orange cluster to target with a delete proposal as well as several overlapping clusters as the designated absorbing set. The first step is the construction of new assignment responsibilities $\hat{r}'$ where any mass previously assigned to orange reassigned among the absorbing set, adding to any preexisting mass already on these clusters. The second step is construction of new global parameters $\hat{\tau}', \hat{\nu}', \hat{u}', \hat{\omega}'$ given the responsibilities $\hat{r}'$. The complete candidate state has only $K = 4$ active clusters, having *deleted* the original orange cluster.

Next, we select a subset of other clusters $A \subset \{1, 2, \dots K\}$ that will absorb the mass from the target cluster. If we select only one absorbing cluster, the resulting delete proposal will be the equivalent to a merge proposal. Selecting more than one absorbing cluster leads to potentially larger changes. Once the target cluster and its absorbing set are chosen, the move proceeds through the standard steps of creating a proposed state and then deciding whether to accept or reject it.

4.3.1 Delete proposal construction of local responsibilities

A delete proposal consists of new local assignments $\hat{r}'_n$ for each data atom in the entire dataset. Within the proposed assigment vector $\hat{r}'_n$, all mass formerly on the target cluster is reassigned among the absorbing set A.

As usual, the candidate responsibility vector $\hat{r}'_n$ for data atom n must obey the non-negativity and sum-to-one constraints of any categorical distribution parameters. We further assume that all clusters not involved in the absorbing set remain at their previous values in $\hat{r}$. Thus, like birth moves we can parameterize the delete in terms of a reassignment probability vector $\tilde{p}_n$ over the $|A|$ entries of the absorbing set.

$$\text{for } k \in 1, 2, \dots K-1: \quad \hat{r}'_{nk} = \begin{cases} \hat{r}_{nk} & \text{if } k \notin A \\ (\hat{r}_{nk_{\text{target}}} + \sum_{\ell \in A} \hat{r}_{n\ell}) \tilde{p}_{n,j} & \text{if } k = A_j \end{cases} \tag{4.25}$$

Again, the vector $\tilde{p}_n$ has $|A|$ non-negative entries and sums to one. The construction of $\hat{r}'_n$ above is completely determined by the original vector $\hat{r}_n$ and the reassignment probabilities $\tilde{p}_n$.

Restricted local step for $\tilde{p}_n$. Like the birth move, our delete move proceeds by estimating the best reassignment probabilities $\tilde{p}_n$ for each data atom n via a *restricted local step*.

We first imagine creating a finite mixture model for the restricted subset of $|A|$ absorbing clusters from our current DP mixture model. This limited model has point-estimates $\tilde{\pi}$ for each of the absorbing clusters:

$$\tilde{\pi}_j = \mathbb{E}_q[\pi_k], \qquad k = A_j \tag{4.26}$$

as well as observation model parameters $\{\tilde{\tau}_j, \tilde{\nu}_j\}$ copied directly from the corresponding absorbing cluster indices within the complete original set $\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$.

Given these global parameters, we can compute posterior weights for each cluster in the absorbing set:

$$\text{for } j \in 1, 2, \dots |A|: \quad \tilde{W}_{nj} = \log \tilde{\pi}_j + \mathbb{E}_q[\log p(x_n | \phi_j)] \tag{4.27}$$

After computing the weights, we set reassignment probabilities via $\tilde{p}_n \leftarrow \textsc{RespFromWeights}(\tilde{W}_n)$. This gives a procedure for constructing $\hat{r}'$ given the current assignments $\hat{r}$ and the global parameters $\hat{\eta}, \hat{\tau}, \hat{\nu}$ which has runtime that scales with the size of the absorbing set, not the total number of clusters.

4.3.2 Delete proposal construction of global responsibilities

A delete proposal's local construction delivers proposed assignments $\hat{r}'$ for the $K-1$ remaining clusters, as well as corresponding sufficient statistics. We can construct the global parameters via the standard global optimization steps.

4.3.3 Scalable construction and evaluation with memoized statistics

For large datasets with $B > 1$ batches processed one batch at a time via our memoized algorithm, we can still successfully construct and evaluate delete proposals where candidates are consistent representations of the entire dataset. The basic template is much like the merge move, where at the beginning of each pass through the dataset we identify the target cluster and the absorbing set. In the first batch, we construct the proposed local parameters $\hat{r}'_b$ and save the corresponding statistics $\tilde{N}_b, \tilde{S}_b, \tilde{H}_b$ for all involved clusters. Each of these tracked values scales with the number of absorbing clusters $|A|$, which could be much less than the total number of active clusters K. At each subsequent batch, we perform similar tracking and aggregate the proposed statistics. Finally, after seeing all batches, we can construct the relevant whole-dataset statistics N'^G, S'^G, H'^G, which represent in aggregate the proposed assignments $\hat{r}'$ across all batches. Finally, these summary statistics are fed into the standard global step to obtain candidate global parameters and we can evaluate the resulting candidate.

Extensions to multiple delete proposals. There are two ways to think about performing delete proposals for multiple clusters. First, we could consider performing 2 separate delete proposals for two candidate clusters j and k. Each proposal would be constructed and evaluated independently. As with merge and birth proposals, we cannot simultaneously accept proposals which involve the same clusters, either as the target or in the absorbing set. However, we can *consider* multiple such proposals and only accept the one which is best. If the total involved sets of clusters in two or moves are disjoint, we can accept both.

The second way to think about multiple proposals is to imagine removing two or more clusters *at the same time* and redistributing their mass among the remaining clusters. This is certainly possible, but risks combinatorial explosion in the number of possible pairs or triples of clusters with the number of absorbing sets.

4.3.4 Selecting the target cluster to delete

Two key choices are required in the specification of a delete move: the targeted cluster k_{target} and the absorbing set of clusters $A \subset \{1, 2, \dots K\}$. We first discuss the choice of k_{target}, and later the choice of A.

We can systematically try all possible values of k_{target} in each lap, but this may be expensive since we can accept only one option at a time. Instead, we recommend first tracking which clusters have failed attempts at deletion in the past. After a cluster has failed more than a prescribed number

of proposal attempts, it should not be tried again. Among the remaining clusters, we might use a size bias (preferring to try to delete small but non-zero clusters) or choose them at random. As a practical concern, we avoid trying to simultaneously target a cluster for birth, merge, and deletion.

If speed is not a concern, the absorbing set may be chosen simply as all clusters except k_{target}. This will provide the greatest chance of acceptance, but has cost that scales with $O(K-1)$. When K is in the hundreds or thousands, tracking such a large absorbing set may be problematic. Instead, we can more cleverly identify a much smaller set of 2 to 10 "nearest neighbors" to the chosen target cluster k_{target}. One option would be to choose the closest clusters in the sense of Bregman divergence from the current target's mean parameters. Another option might be to examine the pair-wise cooccurance within the responsibilities $\hat{r}$ and choose potential absorbing clusters this way.

4.4 Experimental results

4.4.1 Toy example where deletes outperform merges

To illustrate the practical benefits of our delete move, we consider the task of training a DP mixture model on a synthetic dataset with $N = 25,000$ observations all drawn from a single Gaussian cluster with mean 0 and variance 1. We consider only the full-dataset training algorithm here with different possible proposal moves, to focus on the question of how well the different moves help us recover the true cluster. For hyperparameters, we set $\gamma = 10$ and the observation model's prior variance to 1.

The resulting inference task is very simple: recover the single true cluster. Ideally, every training algorithm could find the single true cluster from any initialization, regardless of how many clusters we start with. However, we will see in practice that this is not the case.

Fig. 4.5 shows what happens when training a model from $K = 5$ initial clusters with merge moves only. Early on, the merge moves remove two clusters and identify a $K = 3$ model. However, this represents a tricky local optima, where no further merge move is accepted. The standard coordinate ascent iterations have not strictly converged but change *very* slowly. After hundreds of iterations from this fixed point, the ELBO score changes imperceptably.

Although further merge moves are attempted, no merge of the remaining three clusters (illustrated as red, yellow, and blue in Fig. 4.5) improves the ELBO objective. Some merges (such as red and blue) result in small but noticeable ELBO drops, while a merge proposal combining blue and yellow sensibly leads to a much bigger drop due to the much worse configuration that yields.

In contrast to merges, delete moves are more flexible. Delete proposals involve iterations that improve on all proposed responsibilities in the absorbing set, while merges by definition deliver a single fixed set of responsibilities. Even in this simple case, the iterative nature of delete proposals leads to much better candidate models, as the ELBO traces in Fig. 4.5 illustrate. Of course, we note that we need to run the delete proposal for sufficiently many iterations. It can be difficult to know on arbitrary datasets how long is required.

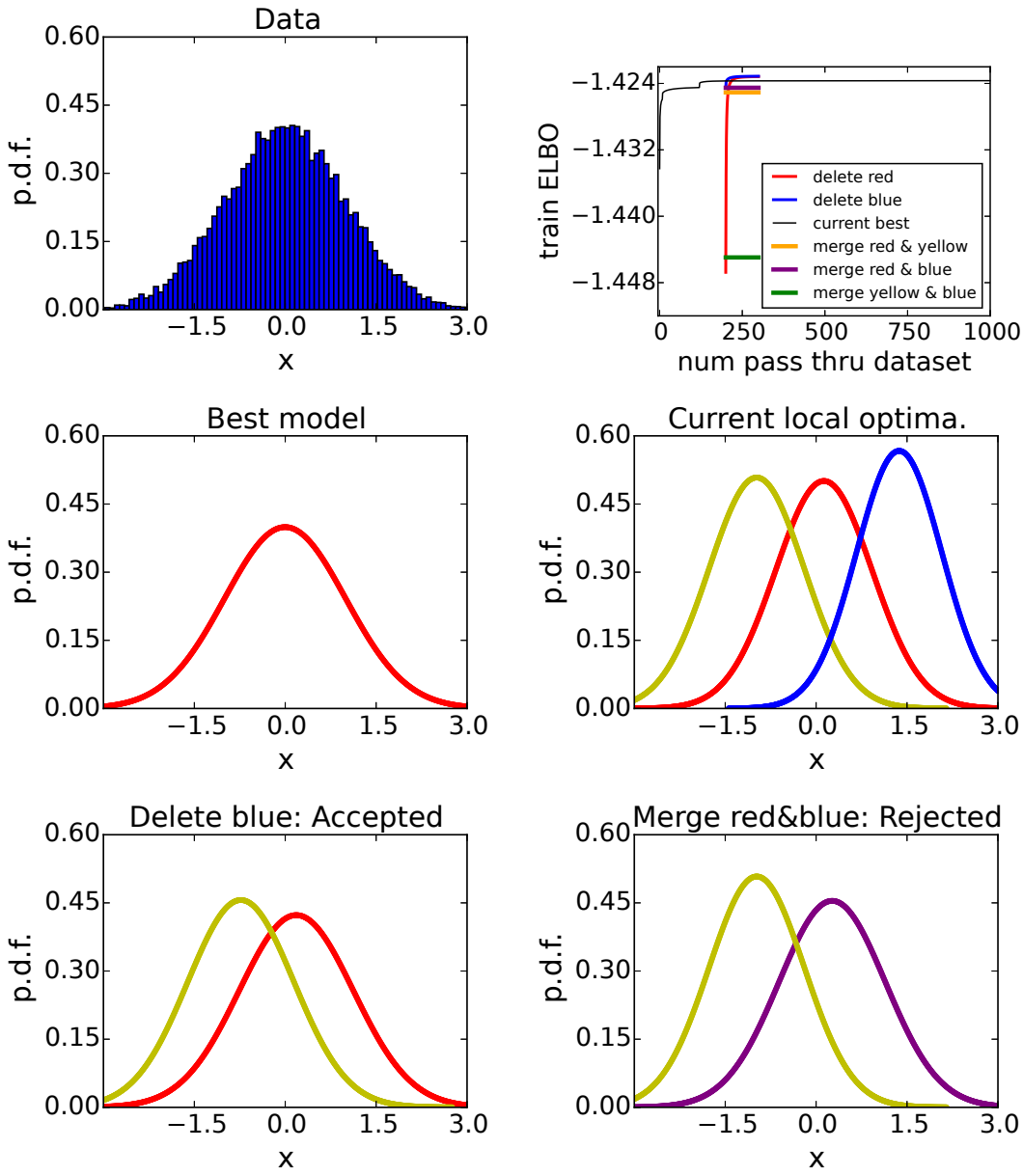

Figure 4.5: 1D Gaussian toy example where merges fail but deletes succeed
Example of poor local optima on simple dataset of 1D samples from single Gaussian cluster with mean 0 and variance 1. Using an initialization with 5 clusters, training with merge moves only leads to a fixed point with 3 clusters which persists for hundreds of iterations. A merge proposal for any pair of clusters leads to lower ELBO score and thus rejection. However, the more flexible delete proposal is accepted. *Top left:* histogram of raw dataset. *Top right:* training ELBO vs. iterations for the merge-only training run, as well as example results of merges and deletes performed at iteration 200. *Middle:* Illustrations of the optimal cluster (used to generate the data) as well as the current local optima. *Bottom:* Illustrations of proposed models trying to escape from the $K = 3$ current local optima. The delete move is accepted, while the merge is rejected.

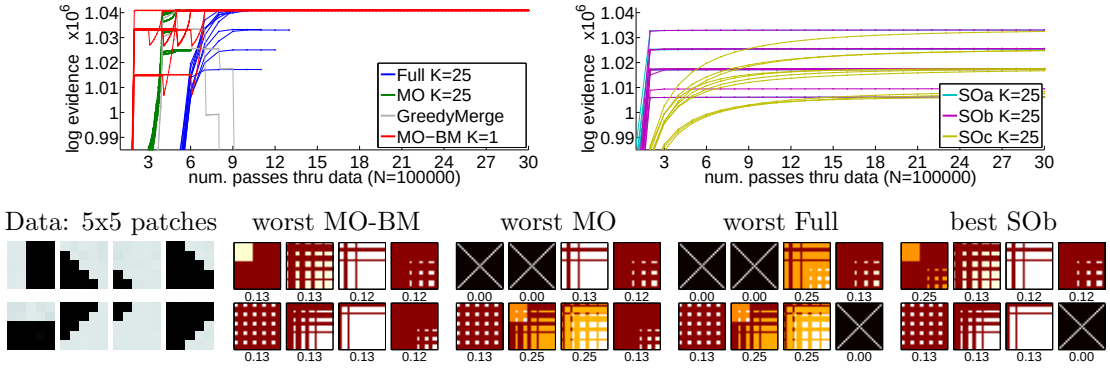

Figure 4.6: Comparison of scalable DP mixture algorithms on synthetic image patch dataset. Results on toy dataset where generated 5x5 patches represent strong edges and corners, inspired by samples from a "dead leaves" model (Zoran and Weiss, 2012). Top: Trace of ELBO during training across 10 runs. Stochastic (SO) algorithm compared with different learning rates a,b,c. Memoized algorithm with birth and merge moves (MO-BM) uses original birth proposals from (Hughes and Sudderth, 2013) which are not guaranteed to improve the objective, but nevertheless consistently find high-quality solutions. *Bottom Left:* Example patch generated by each of the $K = 8$ true clusters. *Bottom Center:* Illustration of the covariance matrices for the trained clusters discovered by one run of each inference method, aligned to the 8 true clusters. "X" indicates no comparable component found. Each covariance matrix has its global appearance probability π_k^G listed below. The true probabilites are 0.125 for each of the 8 clusters.

4.4.2 Toy image patch data

We now compare algorithms for learning DP-Gaussian mixture models (DP-GMM), using our own implementations of full-dataset, stochastic online (SO), and memoized online (MO) inference, as well as our new birth-merge memoized algorithm (MO-BM). To examine SO's sensitivity to learning rate, we use a recommended Hoffman et al. (2013) decay schedule $\xi_t = (t + d)^{-\kappa}$ with three diverse settings: a) $\kappa = 0.5, d = 10$, b) $\kappa = 0.5, d = 100$, and c)$\kappa = 0.9, d = 10$.

For some experiments, we also compare against Kurihara's public implementation of full-dataset split-move variational inference Kurihara et al. (2006). Hyperparameters were chosen for each dataset in an empirical Bayes manner.

We first study $N = 100000$ synthetic image patches generated by a zero-mean GMM with 8 equally-common components. Each one is defined by a 25×25 covariance matrix producing 5×5 patches with a strong edge. We investigate whether algorithms recover the true $K = 8$ structure. Each fixed-truncation method runs from 10 fixed random initializations with $K = 25$, while MO-BM starts at $K = 1$. Online methods traverse 100 batches (1000 examples per batch).

Fig. 4.6 traces the training-set ELBO as more data arrives for each algorithm and shows estimated covariance matrices for the top 8 components for select runs. Even the best runs of SO do not recover ideal structure. In contrast, all 10 runs of our birth-merge algorithm find all 8 components, despite initialization at $K = 1$. The ELBO trace plots show this method escaping local optima, with slight

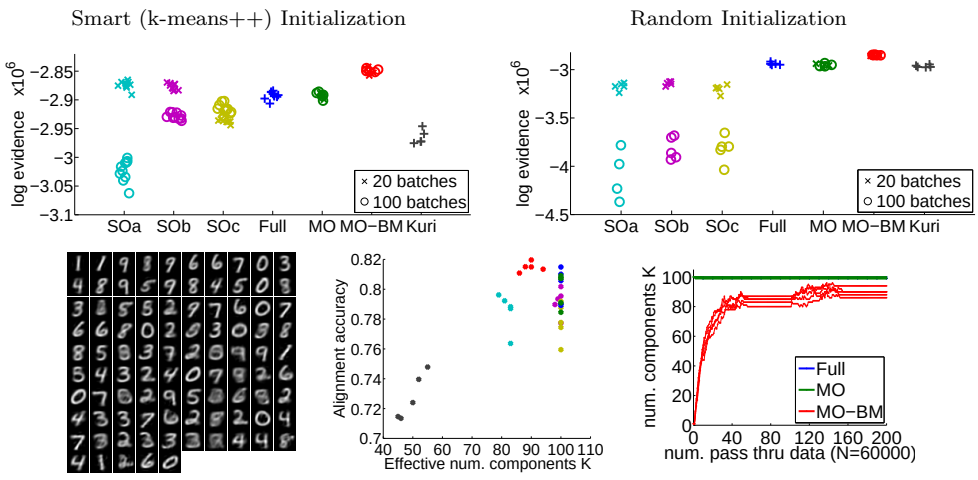

Figure 4.7: Comparison of scalable DP mixture algorithms on MNIST dataset. Results on MNIST dataset. *Top:* Comparison of final ELBO for multiple runs of each method, varying initialization and number of batches. Stochastic online (SO) compared at learning rates a,b,c. *Bottom left:* Visualization of cluster means for MO-BM's best run. *Bottom center:* Evaluation of cluster alignment to true digit label. *Bottom right:* Growth in truncation-level K as more data visited with MO-BM.

drops indicating addition of new components followed by rapid increases as these are adopted. They further suggest that our fixed-truncation memoized algorithm competes favorably with full-data inference, often converging to similar or better solutions after fewer passes through the data.

The fact that our MO-BM algorithm only performs merges that improve the full-data ELBO is crucial. Fig. 4.6 shows trace plots of GreedyMerge, a memoized online variant that instead uses only the current-batch ELBO to assess a proposed merge, as done in Bryant and Sudderth (2012). Given small batches (1000 examples each), there is not always enough data to warrant many distinct 25×25 covariance components. Thus, this method favors merges that in fact remove vital structure. All 5 runs of this GreedyMerge algorithm ruinously accept merges that decrease the full objective, consistently collapsing down to just one component. Our memoized approach ensures merges are always globally beneficial.

4.4.3 MNIST digit clustering

We now compare algorithms for clustering $N = 60000$ MNIST images of handwritten digits 0-9. We preprocess as in Kurihara et al. (2006), projecting each image down to $D = 50$ dimensions via PCA. Here, we also compare to Kurihara's public implementation of variational inference with split moves Kurihara et al. (2006). MO-BM and Kurihara start at $K = 1$, while other methods are given 10 runs from two $K = 100$ initialization routines: random and smart (based on k-means++ Arthur and Vassilvitskii (2007)). For online methods, we compare 20 and 100 batches, and three learning rates. All runs complete 200 passes through the full dataset.

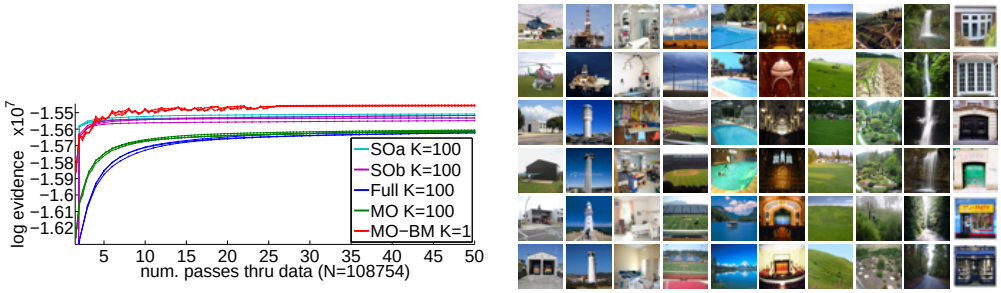

Figure 4.8: Comparison of scalable DP mixture algorithms for clustering tiny images. Observed data: 32x32 color images from SUN dataset, projected via PCA so each observation x_n has dimension 50. *Left:* ELBO during training. *Right:* Visualization of 10 of 28 learned clusters for best MO-BM run. Each column shows two images from the top 3 categories aligned to one cluster.

The final ELBO values for every run of each method are shown in Fig. 4.7. SO's performance varies dramatically across initialization, learning rate, and number of batches. Under random initialization, SO reaches especially poor local optima (note lower y-axis scale). In contrast, our memoized approach consistently delivers solutions on par with full inference, with no apparent sensitivity to the number of batches. With births and merges enabled, MO-BM expands from $K = 1$ to over 80 components, finding better solutions than every smart $K = 100$ initialization. MO-BM even outperforms Kurihara's offline split algorithm, yielding 30-40 more components and higher ELBO values. Altogether, Fig. 4.7 exposes SO's extreme sensitivity, validates MO as a more reliable alternative, and shows that our birth-merge algorithm is more effective at avoiding local optima.

Fig. 4.7 also shows cluster means learned by the best MO-BM run, covering many styles of each digit. We further compute a hard segmentation of the data using the $q(z)$ from smart initialization runs. Each DP-GMM cluster is aligned to one digit by majority vote of its members. A plot of alignment accuracy in Fig. 4.7 shows our MO-BM consistently among the best, with SO lagging significantly.

4.4.4 Clustering tiny images

We next learn a full-mean DP-GMM for tiny, 32×32 images from the SUN-397 scene categories dataset (Xiao et al., 2010). We preprocess all 108754 color images via PCA, projecting each example down to $D = 50$ dimensions. We start MO-BM at $K = 1$, while other methods have fixed $K = 100$. Fig. 4.8 plots the training ELBO as more data is seen. Our MO-BM runs surpass all other algorithms.

To verify quality, Fig. 4.8 shows images from the 3 most-related scene categories for each of several clusters found by MO-BM. For each learned cluster k, we rank all 397 categories to find those with the largest fraction of members assigned to k via $\hat{r}_{\cdot k}$. The result is quite sensible, with clusters for tall free-standing objects, swimming pools and lakes, doorways, and waterfalls.

4.5 Discussion

Across many applications of the DP mixture model, we see clear gains from our adaptive proposal moves compared to fixed-truncation baselines. The merge moves are particularly effective at selecting promising pairs of clusters and successfully merging them without too much additional cost over the standard fixed-truncation algorithm. When possible, using moves that provably increase the objective $\mathcal{L}$ makes the process both easier to debug and more reliable when applied to new applications.

Chapter 5

Scalable variational inference for HDP Topic Models

Ch. 1 motivated topic models (Blei, 2012) as a flexible extension of mixture models which allow each group or *document* d to have its own specific mixture weights π_d. Teh et al. (2006) introduced the hierarchical Dirichlet process (HDP) as a Bayesian nonparametric prior for these document-specific random variables π_d, which leads to improved statistical strength in estimating these values and generalizing to novel documents.

We now develop algorithms for scalable training of approximate posterior representations of the hierarchical Dirichlet process (HDP) topic model. This chapter describes and extends earlier work described in an AISTATS 2015 conference paper, with collaborators Dae Il Kim and Erik Sudderth (Hughes et al., 2015a). The fundamental contributions here identified below:

Contribution 1: New optimization problem for topic models with model selection capability. First, we set up a novel optimization problem for inference which learns approximate posteriors, not point estimates, for *all* global random variables u, ϕ as well as *all* local random variables $\{\pi_d, z_d\}$. We show that this approach leads to beneficial model selection properties, while the point estimation strategy for π^G (equivalently for u) used in previous approaches (Bryant and Sudderth, 2012; Liang et al., 2007) has problems. Achieving this requires a careful surrogate bound to deal with non-conjugacy in the HDP.

Contribution 2: Improved local step single-document inference, with applications to finite and HDP topic models. When visiting a single document, the problem of updating the posteriors for local random variables $q(z_d)$ and $q(\pi_d)$ in any mean-field approach is non-convex with abundant local optima. We first bring attention to this problem by characterizing the practical issues on real data. We then develop novel restart proposals to mitigate this issue.

Contribution 3: Scalable memoized training algorithm for HDP topic model. Until recently, only stochastic variational algorithms were known for training the finite LDA or infinite HDP topic models. We have developed a memoized algorithm with the same per-batch runtime cost and modest storage. Due to non-convexity in the local update step, the algorithm for any topic model lacks monotonicity guarantees but we find in practice that it performs well.

Contribution 4: Birth, merge, and delete proposals to escape local optima. We finally develop proposal moves to escape local optima by adding or removing active clusters. We show that the topic model with multinomial likelihoods is vulnerable to especially cruel local optima, but our moves can often escape from these.

Roadmap. After fully specifying the model, we identify our chosen family of approximate posterior distributions and corresponding free parameters, define the optimization problem's objective function under this chosen family, and derive the basic coordinate ascent updates. Next, we describe scalable versions of our algorithms and discuss the details of proposal moves for the HDP topic model optimization problem.

5.1 Hierarchical Dirichlet process (HDP) topic models

Topic models, also called admixture models, explain datasets which are partitioned into D exchangeable groups $x = \{x_1 \dots x_D\}$. We will often refer to each group as a document, because text modeling is a primary application. However, the notion of grouping may apply equally well to observations from different images, or different hospitals, or any similar application.

Each group or document d contains T_d observations, denoted $x_d = \{x_{d1}, \dots x_{dT_d}\}$. In text analysis, each x_{dt} might represent an individual word from a predefined vocabulary. In image analysis, each x_{dt} might represent the real pixel values of the t-th patch from the d-th image. Remember that we can also index observations without the document structure using the index n, and the total number of observations N is equal to $\sum_{d=1}^{D} T_d$.

Our goal in analyzing the dataset x is to identify a common set of clusters or topics common across all groups, while allowing each group to have variability in topic usage. Teh et al. (2006) introduced the hierarchical Dirichlet process (HDP) topic model illustrated in Fig. 5.1 as a natural model for this goal. The HDP is a hierarchical extension of the Dirichlet process mixture model. It explains the data with an infinite set of possible clusters, $\{\phi_k\}_{k=1}^{K}$, but allows each group or document to have a specific set of appearance probabilities π_d.

Generative model

Like the DP mixture model, in the HDP topic model each topic k is defined by two global variables: the data-generating cluster shape parameters ϕ_k, and a conditional cluster frequency u_k. As in Sec. 2.1.3, we will assume each cluster shape parameter is independently drawn from some prior

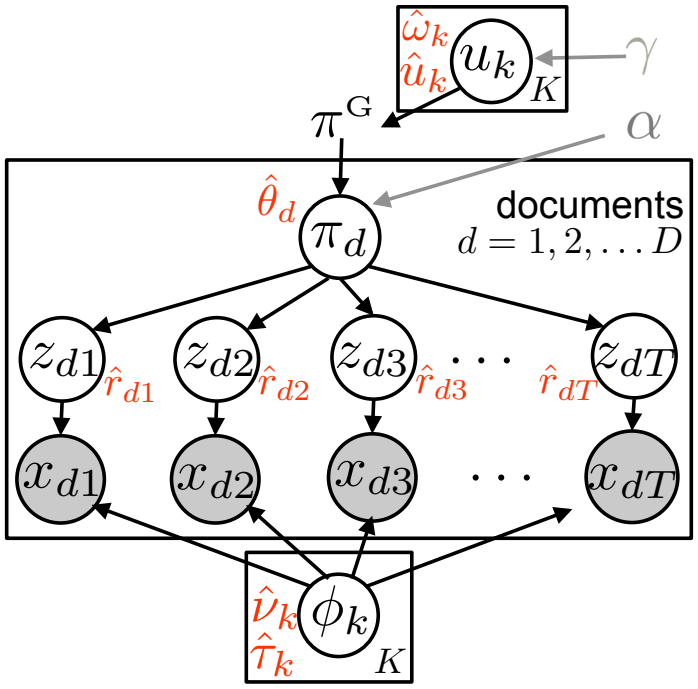

Figure 5.1: Directed graphical representation of hierarchical Dirichlet process topic model. The diagram shows the fundamental random variables (circled nodes), hyperparameters (gray), and variational free parameters (red) of the hierarchical Dirichlet process (HDP) topic model. This Bayesian nonparametric model generates a countably infinite number of clusters under the prior, of which some number $K \leq N$ are assigned to data. Each cluster is defined by two global parameters: conditional probability u_k and shape parameter ϕ_k. The conditional probabilities u are *deterministically* mapped to global cluster probabilities π^G via the invertible stick-breaking transformation. The dataset x consists of D groups or documents, and is assumed to be within-group exchangeable. In addition to the local cluster assignments z_{dt} for each observation in the document, we also assume local document-specific frequency vectors π_d.

distribution $\phi_k \sim \mathrm{P}$. Our focus will be on the generative model for cluster frequencies, because the HDP introduces a more flexible model for this case.

Let each scalar $0 < u_k < 1$ define the *conditional* probability of sampling topic k given that the first $k-1$ topics were *not* sampled. That is, the probability of selecting label k among the infinite set $\{k, k+1, k+2, \ldots\}$. Each value u_k is sampled independently from a beta distribution: $u_k \sim \mathrm{Beta}(1, \gamma)$.

Using the stick-breaking transformation (Sethuraman, 1994; Blei and Jordan, 2006), we can deterministically produce a vector π^G of global cluster probabilities given the vector u.

$$\pi^G_k(u) \triangleq u_k \prod_{\ell=1}^{k-1}(1-u_\ell). \tag{5.1}$$

We interpret the scalar π^G_k as the global probability of topic k appearing in any document. Unlike the vector u, the vector π^G will sum to one by construction.

Next, each group or document has a uniquely-specified topic frequency vector π_d. This vector

has infinitely many entries and will sum-to-one. As a more tractable representatation, we can write this as a finite vector of size $K+1$, where the first K entries represent the first K cluster indices in stick-breaking order and the final entry represents the aggregate mass of all entries beyond index K.

$$\pi_d = [\pi_{d1}\ \pi_{d2} \dots \pi_{dK}\ \pi_{d>K}], \qquad \pi_{d>K} \triangleq= \sum_{\ell=K+1}^{\infty} \pi_{d\ell} \tag{5.2}$$

Given this finite partition of the infinite set of clusters, we define the generative model for the document-specific vector π_d given the global vector π^G:

$$[\pi_{d1} \dots\ \pi_{dK}\ \pi_{d>K}] \sim \mathrm{Dir}(\alpha\pi_1^G, \dots\ \alpha\pi_K^G,\ \alpha\pi_{>K}^G). \tag{5.3}$$

This generative distribution implies the mean of the document-specific probability π_{dk} is the global probability π_k^G. Each document can fluctuate around this mean with variance determined by the concentration parameter $\alpha > 0$.

We complete the generative model by describing the token creation process. First, we sample the t-th observed token's assigned cluster label $z_{dt} \in \{1, 2, \dots K \dots\}$ from a categorical distribution with parameter π_d:

$$z_{dt} \sim \mathrm{Cat}_\infty(\pi_{d1}, \pi_{d2}, \dots \pi_{dK} \dots) \tag{5.4}$$

Finally, like the mixture model we generate each token's data x_{dt} from the chosen likelihood model:

$$x_{dt} \sim \mathrm{L}(x_{dt}|\phi_{z_{dt}}) \tag{5.5}$$

where again we assume the likelihood density L belongs to the exponential family, as described in Sec. 2.1.3.

Interpretation: Document-specific mixture with hierarchically-related frequencies

When studying the model described above and illustrated in Fig. 5.1, we emphasize a specific interpretation: each document's exchangeable observations are generated from a document-specific mixture model. This mixture model is generated by a *hierarchical* chain of Dirichlet process realizations. First, we draw cluster frequencies and shapes $\{\pi_k^G, \phi_k\}_{k=1}^\infty$ from the root or global Dirichlet process. Second, we draw for each document a set of frequency-shape pairs $\{\pi_{dk}, \phi_k\}_{k=1}^\infty$, where the the shape parameters are common across all documents but the frequencies are document-specific random variables related by a common mean vector π^G.

Global and local variables

By inspection of Fig. 5.1, we recognize two global parameters for the HDP topic model: the cluster frequency parameters u_k and cluster shape parameters ϕ_k. We then have two sets of *local* variables at each document: the document-specific probability vector π_d and the discrete assignments $z_d = \{z_{d1}, z_{d2}, \dots z_{dT_d}\}$.

5.2 Posterior inference as a variational optimization problem

Our goal in performing posterior inference for an HDP topic model is to estimate the joint posterior $p(\phi, u, \{\pi_d, z_d\}_{d=1}^D | x)$ given observed documents $x = \{x_1, \dots x_D\}$. As in DP mixture models, estimating the joint posterior directly is intractable, so we set up a variational inference optimization problem instead. The approach described fully below was first given in our earlier conference paper (Hughes et al., 2015a).

5.2.1 Mean-field approximate posterior

Let $q(\phi, u, \{\pi_d, z_d\}_{d=1}^D)$ denote our approximate posterior. We will make the standard mean-field simplifying assumptions about factorizing q. That is, each document d has independent factors π_d and z_d, while each global cluster k has independent factors for u_k and ϕ_k. That is, we can factorize the approximate density as

$$q(\phi, u, \{\pi_d, z_d\}_{d=1}^D) = \prod_{k=1}^{\infty} q(\phi_k) \cdot \prod_{k=1}^{\infty} q(u_k) \cdot \prod_{d=1}^{D} q(\pi_d) \cdot \prod_{d=1}^{D} \prod_{t=1}^{T_d} q(z_{dt}) \tag{5.6}$$

We further assume each factor of the approximate posterior has a density that belongs to the exponential family, whose form naturally mimics the generative model when appropriate. The full parameterization is given here:

$$q(\phi) = \prod_{k=1}^{\infty} \mathrm{P}(\phi_k | \hat{\tau}_k, \hat{\nu}_k) \tag{5.7}$$

$$q(u) = \prod_{k=1}^{\infty} \mathrm{Beta}(u_k | \hat{u}_k \hat{\omega}_k, (1 - \hat{u}_k)\hat{\omega}_k) \tag{5.8}$$

$$q(\pi) = \prod_{d=1}^{D} \mathrm{Dir}_{K+1}(\pi_d | \hat{\theta}_{d1} \dots \hat{\theta}_{dK} \ \hat{\theta}_{d>K}) \tag{5.9}$$

$$q(z) = \prod_{d=1}^{D} \prod_{t=1}^{T_d} \mathrm{Cat}_{\infty}(z_{dt} | \hat{r}_{dt1}, \dots \hat{r}_{dtk} \dots) \tag{5.10}$$

The goal of variational optimization is to find specific values of these free parameters that make $q(u, \phi, \{\pi_d, z_d\}_{d=1}^D)$ a good approximation to the true posterior. For each factor, we denote the free parameters with hats to make clear which variables are instantiated and optimized during inference. Below, we discuss the free parameters in each factor in detail.

Global allocation model free parameters

In Eq. 5.8, we define $q(u_k)$ as a Beta distribution with two parameters: $\hat{u}_k \in [0, 1]$ defines the mean value of u_k, while $\hat{\omega}_k > 0$ defines the variance of u_k. Under this distribution, we have the expectations:

$$\begin{aligned} \mathbb{E}_q[u_k] &= \hat{u}_k \qquad \mathbb{E}_q[1 - u_k] = 1 - \hat{u}_k \qquad & \mathbb{E}_q[\log u_k] &= \psi(\hat{u}_k \hat{\omega}_k) - \psi(\hat{\omega}_k) \\ \mathbb{E}_q[\pi_k^G(u)] &= \hat{u}_k \textstyle\prod_{\ell=1}^{k-1}(1 - \hat{u}_\ell) & \mathbb{E}_q[\log 1 - u_k] &= \psi((1 - \hat{u}_k)\hat{\omega}_k) - \psi(\hat{\omega}_k) \end{aligned} \tag{5.11}$$

Global observation model free parameters

For each cluster k, we have an independent posterior $q(\phi_k)$ over the cluster shape parameter ϕ_k. The definition of $q(\phi_k)$ in Eq. 5.7 follows the standard procedure from Sec. 2.1.3 using free parameters $\hat{\nu}_k$ and $\hat{\tau}_k$. There is nothing new here compared to the version of $q(\phi_k)$ used for the DP mixture model in Ch. 3.

Local parameters for document-specific cluster assignments

As with the DP mixture model, we have a non-negative responsibility vector $\hat{r}_{dt}$ for each observation t in document d. We can interpret the scalar value $\hat{r}_{dtk} \in [0, 1]$ as the probability of assigning this observation to cluster k. To be a valid parameter of a categorical distribution, we have the constraint that the vector $\hat{r}_{dt}$ must be non-negative and sum to one. To gain tractability, we follow the methods in Sec. 3.3.1 and *truncate* $\hat{r}_{dt}$ so that only the first K entries may have positive probability mass. All inactive clusters $k > K$ are constrained by assumption so that $\hat{r}_{dtk} = 0$.

Local parameters for document-specific cluster frequencies

The factor $q(\pi_d)$ has the free parameter $\hat{\theta}_d \geq 0$, which is a vector of size $K+1$ whose entries are non-negative. The first K entries of $\hat{\theta}_d$ can be interpreted as a pseudocount of the total assigned mass for the first K active clusters within document d. The remaining final entry defines the aggregate mass of all inactive clusters. Here are useful expectations for active clusters $k \leq K$:

$$\mathbb{E}_q[\pi_{dk}] = \frac{\hat{\theta}_{dk}}{\hat{\theta}_{d>K} + \sum_{k=1}^{K} \hat{\theta}_{dk}} \qquad \mathbb{E}_q[\log \pi_{dk}] = \psi(\hat{\theta}_{dk}) - \psi(\hat{\theta}_{d>K} + \textstyle\sum_{k=1}^{K} \hat{\theta}_{dk}) \tag{5.12}$$

And for the remaining aggregate mass, we have the expectations:

$$\mathbb{E}_q[\pi_{d>K}] = \frac{\hat{\theta}_{d>K}}{\hat{\theta}_{d>K} + \sum_{k=1}^{K} \hat{\theta}_{dk}} \qquad \mathbb{E}_q[\log \pi_{d>K}] = \psi(\hat{\theta}_{d>K}) - \psi(\hat{\theta}_{d>K} + \textstyle\sum_{k=1}^{K} \hat{\theta}_{dk}) \tag{5.13}$$

5.2.2 Evidence lower-bound objective function

Given the assumed factorization of $q(u, \phi, \{\pi_d, z_d\}_{d=1}^D)$ above, we now set up an optimization problem over our free parameters, just like we did for the DP mixture model in Sec. 3.3.2. The goal is to minimize KL divergence between the approximate posterior q and the true posterior (Wainwright and Jordan, 2008).

$$\underset{\hat{\tau}, \hat{\nu},\ \hat{u}, \hat{\omega},\ \hat{\theta}, \hat{r}}{\arg\min}\ \mathrm{KL}(q(u, \phi, \{\pi_d, z_d\}_{d=1}^D | \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{r}) || p(u, \phi, \{\pi_d, z_d\}_{d=1}^D | x)) \tag{5.14}$$

Computing this KL divergence directly is not possible. However, standard arguments yield an equivalent maximization problem

$$\underset{\hat{\tau}, \hat{\nu},\ \hat{u}, \hat{\omega},\ \hat{\theta}, \hat{r}}{\arg\max}\ \mathcal{L}(x, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{r}) \tag{5.15}$$

where the objective function $\mathcal{L}$ is defined as:

$$\mathcal{L}(x, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{r}) \triangleq \log p(x) - \mathrm{KL}(q(u, \phi, \pi, z|\hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{r})||p(u, \phi, z|x)) \tag{5.16}$$

$$= \mathbb{E}_{q(u,\phi,\pi,z)}\Big[\log p(x, u, \phi, \pi, z) - \log q(u, \phi, \pi, z)\Big] \tag{5.17}$$

Under our chosen exponential family forms for each factor of the approximate posterior q, the expectations that define $\mathcal{L}$ are computable as closed-form functions of the free parameters. We can tractably evaluate $\mathcal{L}$ and take derivatives with respect to the free parameters, making optimization possible.

As in Eq. 2.93 for DP mixtures, we can write the objective $\mathcal{L}$ for HDP topic models as a sum of two terms:

$$\mathcal{L}(x, \hat{r}, \hat{\eta}, \hat{\tau}, \hat{\nu}) = \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta}, \hat{u}, \hat{\omega}) \tag{5.18}$$

These terms describe distinctly interpretable pieces of the overall model: $\mathcal{L}_{\text{data}}$ gathers terms related to the observation model and $\mathcal{L}_{\text{alloc}}$ gathers terms related to the HDP topic allocation model. These terms may also be functions of the hyperparameters $\gamma, \alpha, \bar{\tau}, \bar{\nu}$, but we omit these arguments in notation for simplicity.

This term-by-term breakdown of the objective encourages a modular implementation. Given fixed assignments $\hat{r}$, the solution to finding the optimal observation model parameters $\hat{\nu}, \hat{\tau}$ must be independent of the allocation probability parameters, and vice versa. This modularization enables our implementation to implement free parameter updates once for each possible observation model or allocation model, and compose these modules to create an overall model.

Observation model term of the objective

The term $\mathcal{L}_{\text{data}}$ for HDP topic models is no different from the same term for DP mixture models. We thus refer to the original expression in Eq. 2.95, which can be evaluated given valid free parameters $\hat{r}, \hat{\tau}, \hat{\nu}$ as well as data x. This expression does not rely on any document-specific quantities, so we use the single-level data index n to identify responsibilities $\hat{r}_n$ and data x_n, rather than the corresponding two-level document-token index d, t.

Allocation model term of the objective

For the HDP topic model, we write the allocation model's contribution to the objective as:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\theta}, \hat{u}, \hat{\omega}) \triangleq \sum_{d=1}^{D} \mathbb{E}_{q(z_d)q(\pi_d)q(u)}\Big[\log \frac{p(z_d|\pi_d)}{q(z_d|\hat{r}_d)} + \log \frac{p(\pi_d|\alpha, \pi^G(u))}{q(\pi_d|\hat{\theta}_d)}\Big] \tag{5.19}$$

$$+ \sum_{k=1}^{\infty} \mathbb{E}_{q(u)}\Big[\log \frac{p(u_k|\gamma)}{q(u_k|\hat{u}, \hat{\omega})}\Big]$$

Regrouping terms, we can separate this into an entropy term for the distribution $q(z|\hat{r})$, a term which gathers any document-specific quantities, and a term for the remaining global quantities:

$$\mathcal{L}_{\text{alloc}}(\hat{r}, \hat{\eta}) = \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{HDP-doc}}(\hat{r}, \hat{\theta}, \hat{u}) + \mathcal{L}_{\text{HDP-top}}(\hat{u}, \hat{\omega}) \tag{5.20}$$

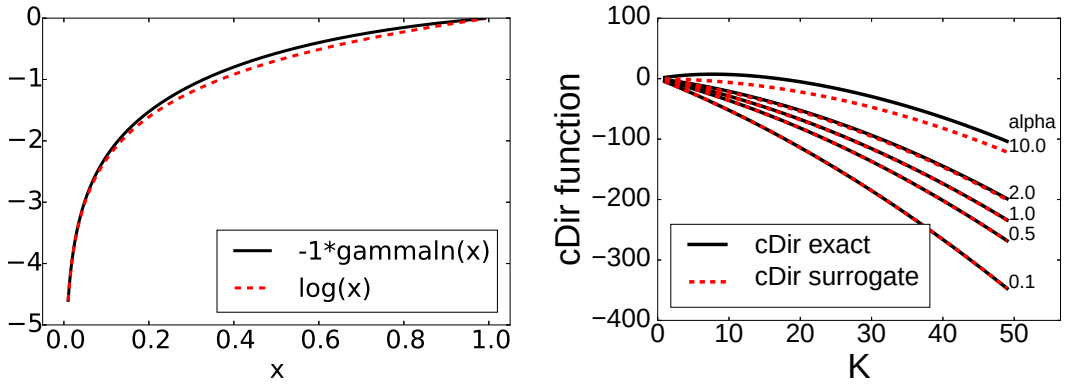

Figure 5.2: Plots of surrogate bound needed to handle non-conjugacy of HDP topic model
Left: Illustration of the bound $-\log\Gamma(x) \geq \log(x)$ for $x \in [0, 1]$, which inspired our surrogate bound. *Right:* Illustration of the exact log Dirichlet cumulant function $c_{\text{Dir}}(\alpha b_1, \alpha b_2, \ldots \alpha b_{K+1})$ (Eq. (5.22), solid black) as a function of the number of active clusters K, for various $\alpha > 0$. At each value of K, we set the vector $b = [b_1\ b_2 \ldots b_K\ b_{>K}]$ so its active entries are equal to $b_k = \frac{1}{1+\gamma}\prod_{\ell<k}\frac{\gamma}{1+\gamma}$, which is the prior mean of the global probability vector π^G, and the final entry $b_{>K}$ is set so the vector sums to one. We show along side each exact evaluation our proposed surrogate bound (Eq. (5.23), dashed red), which this plot shows to be very tight across a range of practical α values. Topic models are typically trained with $0 < \alpha < 1$ to encourage sparsity.

The specific mathematical forms of each of these terms are given in the sections below.

5.2.3 Global term of the allocation objective, using a surrogate bound

To define the global allocation term, we have gathered all terms that depend only on the global free parameters $\hat{u}$ and $\hat{\omega}$.

$$\mathcal{L}_{\text{HDP-top}}(\hat{u}, \hat{\omega}) \triangleq \mathbb{E}_{q(u)}[\log \frac{p(u_k)}{q(u_k)}] + D\mathbb{E}_{q(u)}[c_{\text{Dir}}(\alpha\pi^G(u))] \tag{5.21}$$

These expectations have no dependencies on any local free parameters $\{\hat{\theta}_d, \hat{r}_d\}_{d=1}^D$. We have included the Dirichlet cumulant function that appears in the expectation of $\mathbb{E}_q[\log \text{Dir}(\pi_d|\alpha\pi^G)]$, because this expectation is purely a function of $\hat{u}, \hat{\omega}$.

Under the chosen form of $q(u)$, we have closed-form expressions for the expectations of $\log p(u_k)$ and $\log q(u_k)$ in Eq. 5.21. However, the term $\mathbb{E}_q[c_{\text{Dir}}(\alpha\pi^G(u))]$ is problematic: there is no closed-form expression for this expectation.

To see this, recall that c_{Dir} is the cumulant function of the Dirichlet distribution. It takes two parameters: a positive scalar $\alpha > 0$ and a nonnegative vector b of length $K + 1$ that sums to one. The function is then defined as:

$$c_{\text{Dir}}(\alpha b_1, \ldots \alpha b_K, \alpha b_{K+1}) \triangleq \log\Gamma(\alpha) - \sum_{k=1}^{K+1} \log\Gamma(\alpha b_k) \tag{5.22}$$

The difficulty is that the expected value of $\log\Gamma(\alpha\pi_k^G(u))$ when u_k is a beta random variable has no known closed form. To avoid this problematic expectation of log Gamma functions, we introduce a

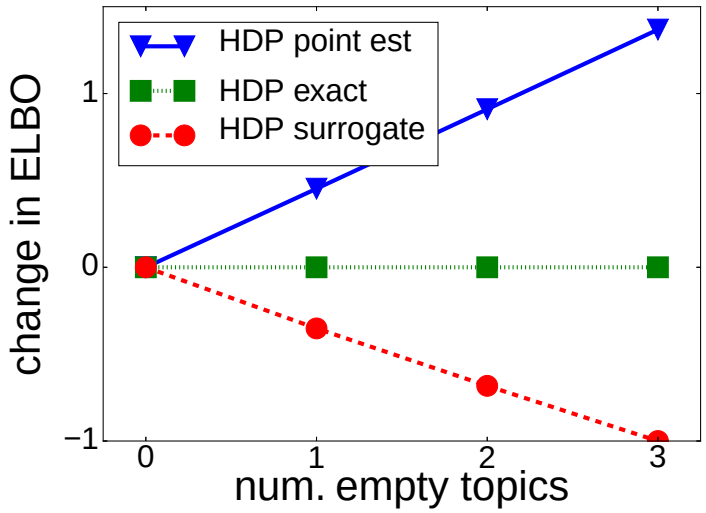

Figure 5.3: HDP model selection: point estimation vs. variational with surrogate bound
We consider several possible algorithms for estimating top-level conditional probabilities u from fixed set of observed assignments z. To assess model selection capability, we consider a fixed set of assignments z with $K = 10$ active clusters, as well as a the same assignments with $K = 11, 12, 13$ active clusters. That is, the later models are simply inserting extra clusters with no assigned observations. In this plot, for each method we show the change in its optimization objective function (the evidence lower bound, or ELBO) from the value with no extra empty topics. We seek a method whos optimization objective function prefers no empty topics to all other cases. *Algorithms:* First, we consider point estimation of $\hat{u}$ directly, which solves the optimization problem $\arg\max_{u,\hat{\theta}} \log p(z, u)$. Second, approximate posterior estimate, which seeks the parameters $\hat{u}, \hat{\omega}$ which define the distribution $q(u) = \prod_{k=1}^{K} q(u_k) = \text{Beta}(\text{mean}{=}\hat{u}_k, \text{scale}{=}\hat{\omega}_k)$ which best approximates the true (intractable) posterior $p(u|z)$. The optimization problem here is $\arg\max_{\hat{u},\hat{\omega},\hat{\theta}} \mathcal{L}(\hat{u}, \hat{\omega}, \hat{\theta}, z)$. *Conclusions:* Our new surrogate bound sensibly prefers models without empty topics, while using point estimation yields undesired preference *for* including empty topics that do not explain any data.

novel lower bound for the cumulant function c_{Dir}. Given a positive scalar $\alpha > 0$ and a nonnegative vector $[b_1, b_2, \dots b_K, b_{K+1}]$ of size $K+1$ which sums to one, we have:

$$c_{\text{Dir}}(\alpha b_1, \alpha b_2, \dots \alpha b_K,\ \alpha b_{K+1}) \geq K \log \alpha + \textstyle\sum_{k=1}^{K+1} \log b_k \tag{5.23}$$

To deal with our intractable expectation $\mathbb{E}_q[c_{\text{Dir}}(\alpha \pi^G(u))]$, we can substitute $\pi^G(u)$ in for the vector b in the above bound and apply the expansions $\pi^G_k = u_k \prod_{\ell<k}(1-u_\ell)$ and $\pi^G_{>K} = \prod_{\ell \leq K}(1-u_\ell)$. After simplifying, we have

$$c_{\text{Dir}}(\alpha \pi^G(u)) \geq K \log \alpha + \textstyle\sum_{k=1}^{K} \log u_k + \sum_{k=1}^{K}(K+1-k)[\log 1 - u_k] \tag{5.24}$$

Under our chosen beta distribution for $q(u_k)$, both expectations required by this lower bound, $E_q[\log u_k]$ and $E_q[\log 1 - u_k]$, have closed form.

Thus, we can define a *surrogate* objective term $\mathcal{L}_{\text{surrogate-HDP-top}}$ which is a tight lower bound on $\mathcal{L}_{\text{HDP-top}}$. That is, $\mathcal{L}_{\text{HDP-top}}(\hat{u}, \hat{\omega}) \geq \mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega})$ for all values of the parameters $\hat{u}, \hat{\omega}$ and all hyperparameters $\alpha > 0, \gamma > 0$. This surrogate objective term can be derived by combining the bound in Eq. 5.24 with the expanded terms in Eq. (5.21):

$$\begin{aligned}
\mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) = DK \log \alpha + \sum_{k=1}^{K} c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{u}_k \hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \\
+ \sum_{k=1}^{K} (D + 1 - \hat{u}_k \hat{\omega}_k) \mathbb{E}_q[\log u_k] \\
+ \sum_{k=1}^{K} (D(K+1-k) + \gamma - (1-\hat{u}_k)\hat{\omega}_k) \mathbb{E}_q[\log 1 - u_k]
\end{aligned} \tag{5.25}$$

5.2.4 Document-specific term of the allocation objective

The term $\mathcal{L}_{\text{HDP-doc}}$ of our overall objective $\mathcal{L}_{\text{alloc}}$ gathers any remaining terms not included in $\mathcal{L}_{\text{entropy}}$ or $\mathcal{L}_{\text{HDP-top}}$. Standard conjugate exponential family mathematics yields an expanded expression:

$$\mathcal{L}_{\text{HDP-doc}}(\hat{r}, \hat{\theta}, \hat{u}) = \sum_{d=1}^{D} \mathbb{E}_q \Big[\log p(z_d | \pi_d) + \log \frac{p(\pi_d)}{q(\pi_d)} \Big] - D \mathbb{E}_q[c_{\text{Dir}}(\alpha \pi^G(u))] \tag{5.26}$$

$$\begin{aligned}
&= \sum_{d=1}^{D} -c_{\text{Dir}}(\hat{\theta}_d) \\
&\quad + \sum_{d=1}^{D} \sum_{k=1}^{K} \Big(N_{dk}(\hat{r}) + \alpha \pi^G_k(\hat{u}) - \hat{\theta}_{dk} \Big) \mathbb{E}_{q(\pi_d | \hat{\theta}_d)}[\log \pi_{dk}] \\
&\quad + \sum_{d=1}^{D} \Big(\alpha \pi^G_{>K}(\hat{u}) - \hat{\theta}_{d>K} \Big) \mathbb{E}_{q(\pi_d | \hat{\theta}_d)}[\log \pi_{d>K}]
\end{aligned} \tag{5.27}$$

Here, we define $N_{dk}(\hat{r}) = \sum_{t=1}^{T_d} \hat{r}_{dtk}$, which is interpreted as the effective count of assignments to cluster k in document d. We have also substituted $\pi^G_k(\hat{u}) \triangleq \mathbb{E}_{q(u)}[\pi^G_k(u)]$ in the sum over active clusters, and $\pi^G_{>K}(\hat{u}) = \mathbb{E}_{q(u)}[\pi^G_{>K}(u)]$ in the last line, where we remember that scalar u_k is an

unknown random variable while $\hat{u}_k$ is a free parameter which defines its approximate posterior mean. All expectations taken with respect to $q(\pi_d)$ are defined in Eq. (5.12) and Eq. (5.13) as digamma functions of $\hat{\theta}$. Finally, the function c_{Dir} is the Dirichlet cumulant defined earlier in Eq. (5.22).

5.2.5 Assignment entropy term of the allocation objective

The entropy of the assignments is a simple non-linear function of the responsibilities:

$$\mathcal{L}_{\text{entropy}}(\hat{r}) = -\sum_{d=1}^{D}\sum_{t=1}^{T_d}\mathbb{E}_q[\log q(z_{dt})] = -\sum_{n=1}^{N}\hat{r}_{dtk}\log\hat{r}_{dtk}. \tag{5.28}$$

We emphasize that no document-specific indexing is required here. Thus, we may replace the two-level document-token indices d, t with the global data index n, which always have a one-to-one invertible relationship. Written using the global indices n for each data observation, this entropy expression is no different than the corresponding entropy term for DP mixture models.

5.3 Update steps for variational optimization

Now, given the objective functions defined above, we can write down the the constrained optimization problem for our free parameters $\hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{r}$ given an observed dataset x and hyperparameters $\mathcal{H}$:

$$\underset{\hat{\tau},\hat{\nu},\hat{u},\hat{\omega},\hat{\theta},\hat{r}}{\arg\max} \quad \mathcal{L}_{\text{data}}(x,\hat{r},\hat{\tau},\hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{HDP-doc}}(\hat{r},\hat{\theta},\hat{u}) + \mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u},\hat{\omega}) \tag{5.29}$$

where the required constraints on the local free parameters for document d are:

$$\begin{aligned} &\hat{r}_{dt} \geq 0 \text{ and } \textstyle\sum_{k=1}^{K}\hat{r}_{dtk} = 1 && \text{for } t = 1, 2, \ldots T_d \\ &\hat{\theta}_{dk} \geq 0 && \text{for } k = 1, 2, \ldots K, K+1 \end{aligned} \tag{5.30}$$

and the required constraints on global free parameters are

$$\begin{aligned} \hat{u}_k \in [0,1] \text{ and } \hat{\omega}_k \geq 0 \quad && \text{for } k = 1, 2, \ldots \\ \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} && \text{for } k = 1, 2, \ldots \end{aligned} \tag{5.31}$$

Given the internal structure of the objective, we pursue a block-coordinate ascent algorithm, which proceeds in two steps, a *local* step and a *global* step. The local step updates the free parameters $\hat{r}_d, \hat{\theta}_d$ for each document d while holding global parameters fixed. The global step updates the global free parameters given fixed (summary statistics of) local parameters.

Below, we define the optimization problem solved by each step of the block-coordinate ascent algorithm. We then describe detailed solutions which solve each problem below in closed-form in the following sections.

Key differences from coordinate ascent for DP mixtures

We stress two key differences from the DP mixture model optimization procedure of Ch. 3. These are useful to have in mind from the outset, to properly understand why the HDP training problem may be more complex than the DP training problem even beyond the fact that there are simply more latent variables.

The first key difference is the DP training algorithm had an objective function which was exactly equal to $\log p(x) - \mathrm{KL}\left(q \| p\right)$. In contrast, the HDP training objective $\mathcal{L}_{HDP}$ is a *lower bound* of this quantity. We need this surrogate bound for tractability, but its consequence is that more complex models are penalized more sharply than in the DP case.

The second key difference lies in the guarantees. The DP training algorithm guaranteed monotonic increase of the objective function $\mathcal{L}$, we can only make this guarantee for the HDP topic model under some costly assumptions: the vector $\hat{\theta}_d$ is always stored for each document.

5.3.1 Global parameter update step for observation model

Consider the optimization problem:

$$\arg\max_{\hat{\tau},\hat{\nu}} \quad \mathcal{L}_{\text{data}}(x, \hat{r}, \hat{\tau}, \hat{\nu}) \qquad \text{subject to } \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \text{ for } k = 1, 2, \ldots. \tag{5.32}$$

The optimization problem in Eq. (5.32) is the same as for the observation model in the DP mixture model, found in Eq. (3.39). This, we can apply the solution from (3.41) without complication.

$$\begin{aligned} \hat{\tau}_k^* &= S_k(x, \hat{r}) + \bar{\tau} \\ \hat{\nu}_k^* &= N_k(\hat{r}) + \bar{\nu} \end{aligned} \tag{5.33}$$

These optimal values naturally satisfy the required constraints $\hat{\nu}_k^* \in \mathbb{R}^+$ and $\hat{\tau}_k^* \in \mathcal{M}$, so that $\hat{\nu}_k^*$ and $\hat{\tau}_k^*$ remain valid parameters for the density $q(\phi_k)$.

5.3.2 Global parameter update step for allocation model

The allocation model global step solves the following optimization problem:

$$\arg\max_{\hat{u},\hat{\omega}} \quad \mathcal{L}_{\text{HDP-doc}}(\hat{r}, \hat{\theta}, \hat{u}) + \mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) \tag{5.34}$$

$$\text{subject to } \hat{u}_k \in [0, 1] \text{ for} \qquad k = 1, 2, \ldots \tag{5.35}$$

$$\hat{\omega}_k \geq 0 \text{ for} \qquad k = 1, 2, \ldots \tag{5.36}$$

Due to non-conjugacy in the generative model's specification that $\pi_d \sim \mathrm{Dir}(\alpha \pi^G(u))$, the global step update for $q(u)$ does not have closed form. However, we can evaluate the objective function of the problem in Eq. (5.34) given any valid free parameters $\hat{u}, \hat{\omega}$. This suggests a numerical optimization approach based on L-BFGS or similar modern gradient descent algorithms.

Gradient descent update for mean parameter $\hat{u}$

Writing the complete objective from the optimization problem in Eq. (5.34) as a function of $\hat{u}$, we have:

$$\begin{aligned} f(\hat{u}) = &- \sum_{k=1}^{K} c_{\text{Beta}}(\hat{u}_k\hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \\ &+ \sum_{k=1}^{K} (D + 1 - \hat{u}_k\hat{\omega}_k)[\psi(\hat{u}_k\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \sum_{k=1}^{K} (D(K+1-k) + \gamma - (1-\hat{u}_k)\hat{\omega}_k)[\psi((1-\hat{u}_k)\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \alpha\pi^G_{>K}(\hat{u})P_{>K}(\hat{\theta}) + \alpha \sum_{k=1}^{K} \pi^G_k(\hat{u})P_k(\hat{\theta}) \end{aligned} \tag{5.37}$$

where we define the summary statistics:

$$\begin{aligned} P_k(\hat{\theta}) &\triangleq \sum_{d=1}^{D} \mathbb{E}_q[\log \pi_{dk}] = \sum_{d=1}^{D} \psi(\hat{\theta}_{dk}) - \psi(\textstyle\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell}) \\ P_{>K}(\hat{\theta}) &\triangleq \sum_{d=1}^{D} \mathbb{E}_q[\log \pi_{d>K}] = \sum_{d=1}^{D} \psi(\hat{\theta}_{d>K}) - \psi(\textstyle\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell}) \end{aligned} \tag{5.38}$$

The function $f(\hat{u})$ and its gradient (which is easily computed either by hand or with autodifferentiation code packages) are all that is needed to perform optimization to find the best value $\hat{u}^*$. As before, we need only focus on the first K indices of vector $\hat{u}^*$, since the any inactive cluster $k > K$ will be optimally set to the prior mean: $\hat{u}^*_k = \frac{1}{1+\gamma}$ for $k > K$.

Heuristic update for variance parameter $\hat{\omega}$

Close inspection reveals a possible simplification: the free parameter vector $\hat{\omega}$, which determines the variance of $q(u_k)$ at each active cluster, only impacts the term $\mathcal{L}_{\text{surrogate-HDP-top}}$ defined in Eq. (5.25). After inspecting the optimal global step update for the similar $q(u)$ of the DP mixture model in Eq. (3.43), he structure of Eq. (5.25) suggests that a useful "rule-of-thumb" update for $\hat{\omega}$ is:

$$\hat{\omega}^*_k = \begin{cases} D(K+1-k) + D + 1 + \gamma & \text{for } k \leq K \\ 1 + \gamma & \text{for } k > K \end{cases} \tag{5.39}$$

Recall that the larger $\hat{\omega}_k$ is, the less variance exists in the distribution $q(u_k)$. Intuitively, because index k of vector $\pi^G(u)$ is determined by the first k indices of vector u, the estimated certainty of $q(u_k)$ should sensibly be highest for index 1 and slowly decay as the index gets larger.

Using the heuristic closed-form update for $\hat{\omega}^*_k$ above, we can save considerable cost in estimating $\hat{\omega}^*_k$ at each iteration of any algorithm while still reaping the benefits of representing a proper approximate posterior distribution for $q(u)$ rather than a point estimate.

5.3.3 Local update step

Given fixed values of the global observation parameters $\hat{\tau}, \hat{\nu}$ and allocation parameters $\hat{u}, \hat{\omega}$, the required optimization problem at each document d is to find responsibilities $\hat{r}_d$ which define $q(z_d)$ and pseudo-counts $\hat{\theta}_d$ which define $q(\pi_d)$.

$$\arg\max_{\hat{r}_d, \hat{\theta}_d} \quad \mathcal{L}_{\text{data}}(x_d, \hat{r}, \hat{\tau}, \hat{\nu}) + \mathcal{L}_{\text{entropy}}(\hat{r}) + \mathcal{L}_{\text{HDP-doc}}(\hat{r}, \hat{\theta}, \hat{u}) \tag{5.40}$$

$$\begin{aligned} \text{subject to } \hat{r}_{dt} \geq 0 \text{ and } \textstyle\sum_{k=1}^{K} \hat{r}_{dtk} = 1 \quad & \text{for } t = 1, 2, \dots T_d \\ \hat{\theta}_{dk} \geq 0 \quad & \text{for } k = 1, 2, \dots K+1 \end{aligned} \tag{5.41}$$

Unlike DP mixture models, this problem does *not* have a closed-form solution. The objective in Eq. (5.40) is non-convex and has many local optima. However, we can apply a block coordinate ascent algorithm to iteratively estimate the best $\hat{r}$ given fixed $\hat{\theta}$, then the best $\hat{\theta}$ given fixed $\hat{r}$, and so on until convergence.

Local substep for assignment responsibilities

Fixing $\hat{\theta}_d$, the objective for token a in document d reduces to an objective similar to the per-token objective $\mathcal{L}$ for DP mixtures in Eq. (3.46).

$$\mathcal{L}_{dt}(x_{dt}, \hat{r}_{dt}, \hat{\tau}, \hat{\nu}, \hat{\theta}) \triangleq \sum_{k=1}^{K} \hat{r}_{dtk} \left(W_{dtk}(x_{dt}, \hat{\tau}, \hat{\nu}, \hat{\theta}) - \log \hat{r}_{dtk} \right) \tag{5.42}$$

$$W_{dtk}(x_{dt}, \hat{\tau}, \hat{\nu}, \hat{\theta}) \triangleq \mathbb{E}_{q(\phi_k)} \Big[\log p(x_{dt} | \phi_k) \Big] + \mathbb{E}_{q(\pi_d)} \Big[\log \pi_{dk} \Big] \tag{5.43}$$

Just as with the DP mixture problem, we can use the RespFromWeights procedure to find the optimum of Eq. (5.42):

$$\hat{r}^*_{dtk} = \frac{e^{W_{dtk}}}{\sum_{\ell=1}^{K} e^{W_{dt\ell}}} \tag{5.44}$$

Local substep for doc-topic pseudo-counts

Next, we consider finding the optimal $\hat{\theta}_d$ for document d, given the requisite constraints and fixed assignments $\hat{r}_d$. Using Lagrange multiplier methods applied to the objective term $\mathcal{L}_{\text{HDP-doc}}$, we find a closed-form optimum. For each active cluster $k \leq K$, we have:

$$\hat{\theta}^*_{dk} = N_{dk}(\hat{r}_d) + \alpha \pi^G_k(\hat{u}) \tag{5.45}$$

while for the aggregate index $_{>}K$ representing all inactive clusters, we have

$$\hat{\theta}^*_{d>K} = \alpha \pi^G_{>K}(\hat{u}) \tag{5.46}$$

Algorithm 5.1 Algorithm for local step under an HDP topic model

Input:

α : positive real, document-topic smoothing scalar
$\{\pi_k^G(\hat{u})\}_{k=1}^K$: expected probability of active topic k under $q(u|\hat{u}, \hat{\omega})$
$\pi_{>K}^G(\hat{u})$: expected probability of aggregate inactive topics.
$\{\{C_{dtk}\}_{k=1}^K\}_{t=1}^{T_d}$: expected log probability of data atom x_{dt} under topic k
$C_{dtk} \triangleq \mathbb{E}_q[\log p(x_{dt}|\phi_k)]$

Output:

$\hat{r}_d$: responsibilities for doc d
$[\hat{\theta}_{d1} \dots \hat{\theta}_{dK} \hat{\theta}_{d>K}]$: topic pseudo-counts for doc d

function LocalStepForDoc($C_d, \alpha, \hat{u}$)
  for $t = 1, \dots T_d$ **do** ▷ Initialize responsibilities $\hat{r}_d$
    for $k = 1 \dots K$ **do**
      $W_{dtk} \leftarrow C_{dtk} + \log(\pi_k^G(\hat{u}))$
    $\hat{r}_{dt} \leftarrow$ RespFromWeights(W_{dt})
  while not converged **do** ▷ Iterate between updating $\hat{\theta}$ and $\hat{r}$
    for $k = 1, 2 \dots K$ **do**
      $N_{dk} \leftarrow \sum_{t=1}^{T_d} \hat{r}_{dtk}$
      $\hat{\theta}_{dk} \leftarrow N_{dk} + \alpha \pi_k^G(\hat{u})$
      $P_{dk} \leftarrow \psi(\hat{\theta}_{dk})$ ▷ Log prior probability $E[\log \pi_{dk}]$
    for $t = 1, 2 \dots T_d$ **do**
      for $k = 1, 2 \dots K$ **do**
        $W_{dtk} = C_{dtk} + P_{dk}$
      $\hat{r}_{dt} =$ RespFromWeights(W_{dt})
  $\hat{\theta}_{d>K} \leftarrow \alpha \pi_{>K}^G(\hat{u})$ ▷ Set inactive term
  return $\hat{r}_d, \hat{\theta}_d$

This algorithm solves the optimization problem defined in Eq. (5.40), producing optimal document-specific responsibilities $\hat{r}_d$ defining the assignment posterior $q(z_d|\hat{r}_d)$ and pseudocounts $\hat{\theta}_d$ defining the topic frequency posterior $q(\pi_d|\hat{\theta}_d)$.

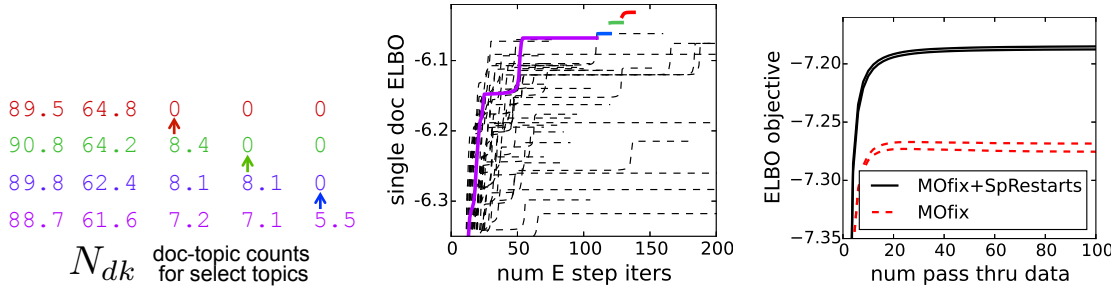

Figure 5.4: Illustration of restart proposals for HDP topic model local step. Sparsity-promoting restarts for local update steps on the Science corpus with $K = 100$. *Left:* Example fixed points of the document-topic count summary statistic N_{dk} for a single document in the Science corpus. We show only select topic indices out of all $K = 100$. *Center:* Trace of a single document's objective $\mathcal{L}$ during the local step inference for 50 random initializations (dashed lines). The solid lines show one run with sparsity-promoting moves enabled. This run climbs through the color coded fixed points in the left plot. *Right:* Trace plot of the whole-dataset objective $\mathcal{L}$ across many passes through the whole Science corpus. Using sparsity-promoting restarts yields noticeable improvements in model quality.

Iterative algorithm for joint local step update

To find an ideal joint configuration $\hat{r}_d^*, \hat{\theta}_d^*$, we iterate between the updates above until convergence. An algorithm describing the required iterations is given in Alg. 5.1. This algorithm includes our recommended heuristic initialization (discussed below) and delivers a final value for the responsibilities $\hat{r}_d^*$. From this converged value, we can easily compute the required $\hat{\theta}_d^*$.

Local step initialization. To initialize the update cycle for a document, we recommend visiting each token and updating it with initial weight $W'_{dtk} = \mathbb{E}_q[\log p(x_{dt}|\phi_k)] + \log \pi_k^G(\hat{u})$. This initialization lets the likelihood drive the initial assignments while still incorporating the current best estimate of the global topic probabilities $\pi^G(\hat{u})$. We then alternate updates to $\hat{r}$ and $\hat{\theta}$ until either a maximum number of iterations is reached (typically 100) or the maximum change of all document-topic counts N_{dk} falls below a threshold (typically 0.05).

5.3.4 Sparse restart proposals for local step

When visiting document d, the joint inference of $\hat{\theta}$ and $\hat{r}$ can be challenging due to the non-convexity of the joint inference problem in Eq. (5.40). Many local optima exist even for this single-document task, as shown Fig. 5.4. A common failure mode occurs when a few tokens are assigned to a rare "junk" topic. Reasignment of these tokens may not happen during the standard coordinate ascent updates due to a valley in the objective between keeping the current junk assignments and setting the junk topic to zero.

To more adequately escape local optima, we develop sparsity-promoting restart moves which take a final document-topic count vector $[N_{d1} \dots N_{dK}]$ produced by coordinate ascent, propose an alternative which has one entry set to zero, and accept if this improves the ELBO after further

Algorithm 5.2 Algorithm for restart proposals used in local step of inference for HDP topic model

Input: α : document-topic smoothing scalar
$\{\{C_{dtk}\}_{k=1}^{K}\}_{a=1}^{T_d}$: expected log probability of token d, t under topic k
$C_{dtk} \triangleq \mathbb{E}_q[\log p(x_{dt}|\phi_k)]$
$\hat{r}_d$: initial responsibilities for doc d
$[\hat{\theta}_{d1} \dots \hat{\theta}_{dK} \hat{\theta}_{d>K}]$: initial topic pseudo-counts for doc d

Output:
$\hat{r}_d$: responsibilities for doc d
$[\hat{\theta}_{d1} \dots \hat{\theta}_{dK} \hat{\theta}_{d>K}]$: topic pseudo-counts for doc d

function RestartProposalsForDoc($\hat{r}_d, \hat{\theta}_d$)
  for $k = 1, \dots K$ **do**
    $N_{dk} \leftarrow \sum_{t=1}^{T_d} \hat{r}_{dtk}$
  ▷ Initial usage counts.
  $\mathcal{A}_d \leftarrow \{k : N_{dk} > 0.1\}$
  for $j \in \mathcal{A}_d$ **do**
    $\hat{r}'_d \leftarrow$ Copy($\hat{r}_d$)
    for $t = 1, 2, \dots T_d$ **do**
      $\hat{r}'_{dtj} \leftarrow 0$
    ▷ Propose sparser $q(\pi_d)$ by setting count to zero
    while not converged **do**
      for $k = 1, \dots K$ **do**
        $N'_{dk} \leftarrow \sum_{t=1}^{T_d} \hat{r}'_{dtk}$
        $\hat{\theta}'_{dk} \leftarrow N'_{dk} + \alpha \pi_k^G(\hat{u})$
        $P'_{dk} \leftarrow \psi(\hat{\theta}'_{dk})$
      for $t = 1, 2, \dots T_d$ **do**
        for $k = 1, 2, \dots K$ **do**
          $W'_{dtk} \leftarrow C_{dtk} + P'_{dk}$
        $\hat{r}'_{dt} \leftarrow$ RespFromWeights(W'_{dt})
    if $\mathcal{L}_d(\hat{r}'_d, \hat{\theta}'_d) > \mathcal{L}_d(\hat{r}_d, \hat{\theta}_d)$ **then** ▷ Keep proposal if document score improves
      $\hat{r}_d \leftarrow \hat{r}'_d$
      $\hat{\theta}_d \leftarrow \hat{\theta}'_d$
  return $\hat{r}_d, \hat{\theta}_d$

Restart proposals attempt to find better local optima of the variational optimization objective function in Eq. (5.40). Based on the heuristic that the document-topic prior prefers sparsity, this procedure repeatedly proposes alternative local parameters where one currently used topic has its usage forced to zero. This procedure is guaranteed to monotonically improve the document-specific objective function.

ascent steps. In practice, the acceptance rate varies from 30-50% when trying the 25 smallest non-zero topics. We observe huge gains in the whole-dataset objective due to these restarts, as shown in Fig 5.4, while without them we sometimes even observe *decreasing* of the overall objective on repeat visits to the same document.

5.3.5 Specialization to bag-of-words datasets

Several aspects of the algorithm can be simplified or clarified for the popular use-case of training on discrete bag-of-words datasets. Let each document x_d consist of observed word tokens from a fixed vocabulary of V word types. We can represent x_d in two ways. First, as a dense list of the T_d word tokens in document d: $x_d = \{x_{dt}\}_{n=1}^{T_d}$. Each value $x_{dt} \in \{1, \dots V\}$ identifies the type of the t-th word in the document. Second, we can use a memory-saving sparse histogram representation: $x_d = \{v_{du}, c_{du}\}_{u=1}^{U_d}$, where u indexes the set of word types that appear at least once in the document, $v_{du} \in \{1, \dots V\}$ gives the integer id of word type u, and $c_{du} \geq 1$ is the count of word type v_{du} in document d. Naturally, $\sum_{u=1}^{U_d} c_{du} = \sum_{t=1}^{T_d} x_{dt} = T_d$.

Sharing parameters by word type. Naively, tracking the assignments for document d requires explicitly representing a separate K-dimensional distribution $q(z_{dt})$ for each of the T_d tokens. However, we can save memory and runtime by recognizing that given a specified document-topic approximate posterior $q(\pi_d|\hat{\theta}_d)$, the corresponding update for the assignment posterior $q(z_{dt})$ will have shared structure for any tokens of the same type. That is, if there exist two token indices s and t in document d such that $x_{ds} = v$ and $x_{dt} = v$, then by definition after a local update their responsibilities will be exactly the same: $\hat{r}_{ds} = \hat{r}_{dt}$. We can thus share parameters with no loss in representational power. Instead of T_d total responsibility vectors, we need only U_d distinct vectors, one for each unique vocabulary word appearing at least once in the document. This leads to substantial savings in required storage as well as required computation, because frequently $U_d << T_d$ for real text data due to word burstiness.

5.4 Algorithms for HDP topic model posterior estimation

5.4.1 Full-dataset variational

When the whole dataset can fit in memory, a complete algorithm for estimating the global parameters of the HDP topic model from data is given in Alg. 5.3. The algorithm alternates between the local step for each document (a subprocedure specified in Alg. 5.1) and the global step, which updates both the allocation parameters $\hat{u}, \hat{\omega}$ and observation parameters $\hat{\tau}, \hat{\nu}$. Generally, this procedure is run to convergence at a fixed point local optima. One useful criteria for judging convergence is to monitor the assignment counts N_k for each active cluster over many iterations. Convergence can be measured by checking whether the maximum absolute change in any entry of this vector drops below a small threshold.

Algorithm 5.3 Variational coordinate ascent for HDP topic model

Input:

$\{x_d\}_{d=1}^D$: dataset with D documents
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: initial global parameters of allocation model
γ, α: allocation model hyperparameters
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: updated global parameters of allocation model

function VariationalCoordAscentForHDPTopic($x, K, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}$)
  while not converged **do**
    for $d \in 1, 2, \dots D$ **do** ▷ Local step at document d
      for $t \in 1, 2, \dots T_d$ **do**
        for $k \in 1, 2, \dots K$ **do**
          $C_{dtk} \leftarrow \mathbb{E}_q[\log p(x_{dt}|\phi_k)]$
      $\hat{r}_d, \hat{\theta}_d \leftarrow$ LocalStepForDoc($C_d, \alpha, \hat{u}$)
      $\hat{r}_d, \hat{\theta}_d \leftarrow$ RestartProposalsForDoc($\hat{r}_d, \hat{\theta}_d, C_d, \alpha, \hat{u}$)
    for $k \in 1, 2, \dots K$ **do** ▷ Summary step
      $S_k \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk} s(x_{dt})$
      $N_k \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk}$
      $P_k \leftarrow \sum_{d=1}^D \psi(\theta_{dk}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$
    $P_{>K} \leftarrow \sum_{d=1}^D \psi(\theta_{d>K}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$
    for $k \in 1, 2, \dots K$ **do** ▷ Global step for observation parameters
      $\hat{\tau}_k \leftarrow S_k + \bar{\tau}$
      $\hat{\nu}_k \leftarrow N_k + \bar{\nu}$
    $\hat{u} \leftarrow \arg\max_{\hat{u}} f(\hat{u}, \hat{\omega}, P(\hat{\theta}), \alpha, \gamma)$ ▷ Global step for allocation parameters, via L-BFGS
  return $\hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}$

Full dataset algorithm for approximate posterior inference for the HDP topic model in Fig. 5.1. For the LocalStepForDoc procedure, see Alg. 5.1.

5.4.2 Stochastic variational

The application of stochastic variational inference (Hoffman et al., 2013) to our direct assignment HDP topic model optimization problem in Eq. (5.29) is almost straightforward. The natural gradient update for the observation model parameters $\hat{\tau}, \hat{\nu}$ applies without change from the DP mixture model case due to conditional conjugacy. However, the non-conjugate relationship between $q(u)$ and $q(\pi_d)$ does not allow the direct application of the natural gradient update. Nevertheless, we develop a modified stochastic algorithm given in Alg. 5.4.

5.4.3 Memoized variational

A memoized algorithm for the HDP topic model variational optimization problem is in Alg. 5.5.

Algorithm 5.4 Stochastic variational coordinate ascent for HDP topic model

Input:

$\{x_d\}_{d=1}^D$: dataset with D documents
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: initial global parameters of allocation model
γ, α: allocation model hyperparameters
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: updated global parameters of allocation model

function StochasticVariationalForHDPTopic($x, K, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}$)
  for iteration $i \in 1, 2, \ldots$ **do**
    $\mathcal{D}_i \leftarrow$ SampleWithoutReplacement($\{1, 2, \ldots D\}, D/B$)
    for $d \in \mathcal{D}_i$ **do** ▷ Local step at current batch
      for $a \in 1, 2, \ldots A_d$ **do**
        for $k \in 1, 2, \ldots K$ **do**
          $C_{dtk} \leftarrow \mathbb{E}_q[\log p(x_{dt}|\phi_k)]$
      $\hat{r}_d, \hat{\theta}_d \leftarrow$ LocalStepForDoc($C_d, \alpha, \hat{u}$)
      $\hat{r}_d, \hat{\theta}_d \leftarrow$ RestartProposalsForDoc($\hat{r}_d, \hat{\theta}_d, C_d, \alpha, \hat{u}$)

    for $k \in 1, 2, \ldots K$ **do** ▷ Summary step at current batch
      $S_k \leftarrow \sum_{d \in \mathcal{D}_i} \sum_{t=1}^{T_d} \hat{r}_{dak} s(x_{da})$
      $N_k \leftarrow \sum_{d \in \mathcal{D}_i} \sum_{t=1}^{T_d} \hat{r}_{dak}$
      $P_k \leftarrow \sum_{d \in \mathcal{D}_i} \psi(\theta_{dk}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$
    $P_{>K} \leftarrow \sum_{d \in \mathcal{D}_i} \psi(\theta_{d>K}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$

    $\xi_i \leftarrow (\delta + i)^{-\kappa}$ ▷ Update learning rate
    for $k \in 1, 2, \ldots K$ **do**
      $\hat{\tau}_k \leftarrow (1 - \xi_i)\hat{\tau}_k + \xi_i \left(\bar{\tau} + \frac{D}{|\mathcal{D}_i|} S_{tk} \right)$ ▷ Global step for observation parameters
      $\hat{\nu}_k \leftarrow (1 - \xi_i)\hat{\nu}_k + \xi_i \left(\bar{\nu} + \frac{D}{|\mathcal{D}_i|} N_{tk} \right)$
    $\hat{u}_i \leftarrow \arg\max_{\hat{u}} f(\hat{u}, \hat{\omega}, \frac{D}{|\mathcal{D}_i|} P(\hat{\theta}), \alpha, \gamma)$ ▷ Global step for allocation parameters

  return $\hat{\tau}, \hat{\nu}, \ \hat{u}, \hat{\omega}$

Stochastic variational algorithm for approximate posterior inference for the HDP topic model in Fig. 5.1. For the LocalStepForDoc, see Alg. 5.1.

Algorithm 5.5 Memoized variational coordinate ascent for HDP topic model

Input:

$\{x_d\}_{d=1}^D$: dataset with D documents
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: initial global parameters of allocation model
γ, α: allocation model hyperparameters
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: updated global parameters of allocation model

function MemoizedVariationalForHDPTopic($x, K, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}$)
  $S^G, N^G, T^G \leftarrow 0$ ▷ Initialize global statistics
  for batch $b \in 1, 2, \dots B$ **do**
    $S_b, N_b, T_b \leftarrow 0$ ▷ Initialize batch statistics
  for lap $\ell \in 1, 2, \dots$ **do**
    for batch $b \in$ Shuffle($\{1, 2, \dots B\}$) **do**
      for doc $d \in \mathcal{D}_b$ **do** ▷ Local step at current batch
        for $t \in 1, 2, \dots T_d$ **do**
          for $k \in 1, 2, \dots K$ **do**
            $C_{dtk} \leftarrow \mathbb{E}_q[\log p(x_{dt}|\phi_k)]$
        $\hat{r}_d, \hat{\theta}_d \leftarrow$ LocalStepForDoc($C_d, \alpha, \hat{u}$)
        $\hat{r}_d, \hat{\theta}_d \leftarrow$ RestartProposalsForDoc($\hat{r}_d, \hat{\theta}_d, C_d, \alpha, \hat{u}$)

      $S^G \leftarrow S^G - S_b$ ▷ Decrement previous batch statistics
      $N^G \leftarrow N^G - N_b$
      $T^G \leftarrow T^G - T_b$

      for $k \in 1, 2, \dots K$ **do** ▷ Summary step at current batch
        $S_{bk} \leftarrow \sum_{d \in \mathcal{D}_b} \sum_{a=1}^{A_d} \hat{r}_{dtk} s(x_{da})$
        $N_{bk} \leftarrow \sum_{d \in \mathcal{D}_b} \sum_{a=1}^{A_d} \hat{r}_{dtk}$
        $P_{bk} \leftarrow \sum_{d \in \mathcal{D}_b} \psi(\theta_{dk}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$
      $P_{b>K} \leftarrow \sum_{d \in \mathcal{D}_b} \psi(\theta_{d>K}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$

      $S^G \leftarrow S^G + S_b$ ▷ Increment new batch statistics
      $N^G \leftarrow N^G + N_b$
      $T^G \leftarrow T^G + T_b$

      for $k \in 1, 2, \dots K$ **do**
        $\hat{\tau}_k \leftarrow \bar{\tau} + S_k^G$ ▷ Global step for observation parameters
        $\hat{\nu}_k \leftarrow \bar{\nu} + N_k^G$
      $\hat{u} \leftarrow \arg\max_{\hat{u}} f(\hat{u}, \hat{\omega}, P^G, \alpha, \gamma)$ ▷ Global step for allocation parameters, via L-BFGS
  return $\hat{\tau}, \hat{\nu}, \ \hat{u}, \hat{\omega}$

For the LocalStepForDoc, see Alg. 5.1. Memoized algorithm for approximate posterior inference for the HDP topic model in Fig. 5.1.

Memoized algorithm and monotonicity guarantee

In general, the objective function $\mathcal{L}$ is guaranteed to monotonically increase after any global step, but *not* after the local step. This occurs because the local step requires jointly estimating the assignment responsibilities $\hat{r}_d$ and document-topic probability pseudo-counts $\hat{\theta}_d$ *from scratch* when visiting a document d. Because this local step problem is non-convex even given fixed global parameters, we cannot guarantee that on subsequent visits to a document d we will always improve the objective. Instead, it may be the case that even after restart proposals, the fixed point $\hat{r}_d, \hat{\theta}_d$ leads to a drop in the $\mathcal{L}$ score. We find that this is rare, but it can happen.

Without a procedure to always deliver a guaranteed global optima to the local step problem, the only way to *guarantee* monotonic increase of $\mathcal{L}$ is to warm-start the local step iterations at a previous value of $\hat{r}_d$ or $\hat{\theta}_d$ stored from a previous visit to the document d. However, this requires tremendous memory. For example, we'd need to store the value of $\hat{\theta}_{dk}$ for all K active topics and D documents. With thousands of topics and millions of documents, this becomes infeasible.

Memoized evaluation of the objective

As in DP mixture models, after completing the first epoch or lap of memoized inference in Alg. 5.5, the global statistics N^G, S^G, T^G accurately represent every document in the dataset and thus we can compute the score of the whole objective $\mathcal{L}$ at any point beyond lap $\ell = 1$. We do need several auxiliary statistics, defined below.

Entropy statistic for ELBO computation. Like in DP mixture models, we require tracking the assignment entropy of each active topic in each batch. Let H_{bk} define the sum of assignment entropies within batch b for topic k. Then given the whole-dataset aggregated entropy values $H_k^G = \sum_b H_{bk}$, we can easily evaluate $\mathcal{L}_{\text{entropy}}$ from Eq. (5.28) as:

$$\mathcal{L}_{\text{entropy}}(\hat{r}) = \sum_{k=1}^{K} H_k^G(\hat{r}), \qquad H_k^G = \sum_{b=1}^{B} H_{bk}, \qquad H_{bk}(\hat{r}) \triangleq -\sum_{d \in \mathcal{D}_b} \sum_t \hat{r}_{dtk} \log \hat{r}_{dtk}. \tag{5.47}$$

Document-specific statistics for ELBO computation. The document-specific term of the allocation model $\mathcal{L}_{\text{HDP-doc}}$ in Eq. (5.26) can be rewritten using the statistic vector Q defined below:

$$\mathcal{L}_{\text{HDP-doc}}(\hat{r}, \hat{\theta}, \hat{u}) = Q_0^G(\hat{\theta}) + Q_{>K}^G(\hat{\theta}) + \sum_{k=1}^{K} Q_k^G(\hat{r}, \hat{\theta}) + \sum_{k=1}^{K} \alpha \pi_k^G(\hat{u}) P_k(\hat{\theta}) \tag{5.48}$$

$$Q_0^G \triangleq \sum_b Q_{b0}, \qquad Q_{b0} = \sum_{d \in \mathcal{D}_b} \log \Gamma(\textstyle\sum_{\ell=1}^{K+1} \hat{\theta}_{d\ell})$$

$$\forall k \in \{1, 2, \dots K\} \quad Q_k^G \triangleq \sum_b Q_{bk}, \qquad Q_{bk} = \sum_{d \in \mathcal{D}_b} -\log \Gamma(\hat{\theta}_{dk}) + (N_{dk}(\hat{r}) - \hat{\theta}_{dk})[\psi(\hat{\theta}_{dk}) - \psi(\hat{\theta}_{d\cdot})]$$

$$Q_{>K}^G \triangleq \sum_b Q_{b>K}, \qquad Q_{b>K} = \sum_{d \in \mathcal{D}_b} -\log \Gamma(\hat{\theta}_{d>K}) - \hat{\theta}_{d>K}[\psi(\hat{\theta}_{d>K}) - \psi(\hat{\theta}_{d\cdot})]$$

Thus, with modest storage for Q_b (a $K+2$ dimensional vector) and H_b (a K dimensional vector) required at each batch b, we can compute the whole dataset objective $\mathcal{L}$ exactly despite processing data one batch at a time.

5.5 Variational algorithms with proposal moves that adapt the number of clusters

We now develop a framework for merge, birth, and delete proposal moves for the HDP topic model, building on our earlier efforts for the DP mixture model. Previous efforts for MCMC sampling inference using transition operators that explicitly modify the number of clusters include the split-merge proposal methods of Wang and Blei (2012b), as well as the more recent split-merge sampler by Chang and Fisher III (2014). Among previous methods that adaptively add or remove clusters within a variational framework, two lines of work stand out. First, Wang and Blei (2012a) used a Gibbs sampler inner-loop for the local step, which could add or remove clusters but will generally be slower to make large changes that well-designed explicit proposals. Second, Bryant and Sudderth (2012) offer an approach similar to ours: explicit split and merge proposals which are either accepted or rejected based on improvements to the optimization problem's objective function. However, by using stochastic variational inference, they cannot make accept or reject decisions which are consistent with the entire dataset. Our approach can simultaneously make bigger changes than the Gibbs-within-variational inference methods while offering guarantees of improved model quality not possible with stochastic methods.

5.5.1 Merge proposals

Each merge proposal transforms a current variational posterior $\hat{r}, \hat{\theta}, \hat{u}, \hat{\omega}, \hat{\tau}, \hat{\nu}$ into a candidate with one fewer cluster by combining two chosen topics k_A, k_B into a single *merged* topic. These moves can eliminate redundant topics and improve the final model's interpretability. Accepted merge proposals also make subsequent iterations of our algorithm faster by reducing the number of active topics, since all steps of inference scale linearly with K.

Merge proposal construction

Just like with DP mixture models, given a pair of clusters k_A, k_B to merge, we create the candidate state in two steps. First, we create the local parameters $\hat{r}', \hat{\theta}'$ and their accompanying sufficient statistics. Second, we instantiate values for the candidate global parameters, which are optimal under our objective given the proposed local parameters. A fully-constructed proposal can then be checked by the objective $\mathcal{L}$ to decide acceptance or rejection.

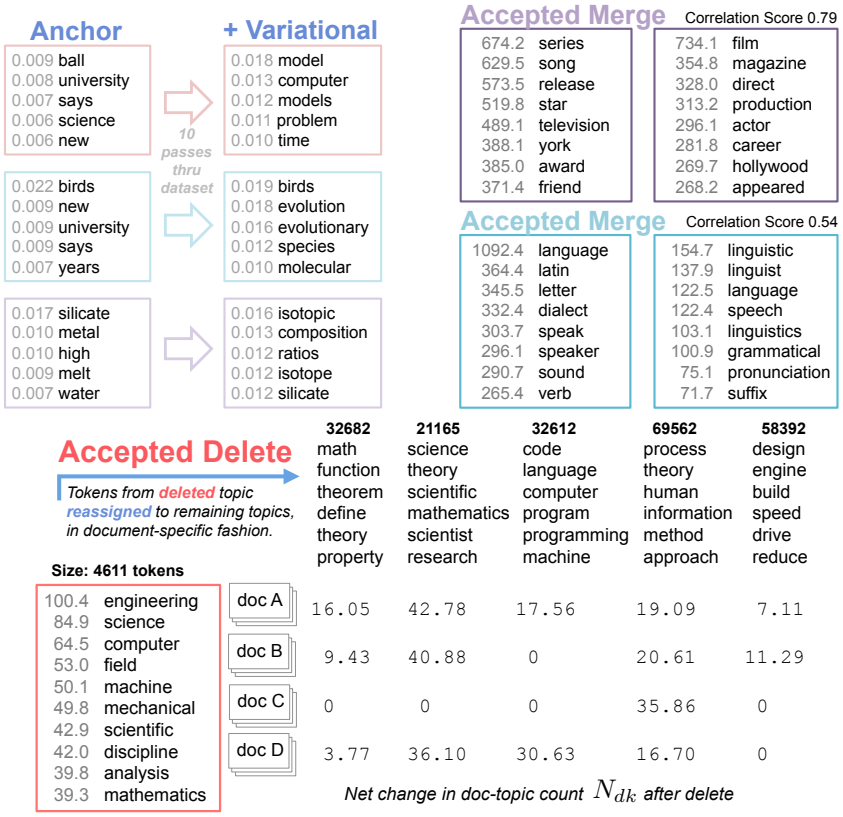

Figure 5.5: Practical examples of merges and deletes on topic models
Top Left: Anchor topics (Arora et al., 2013) can be improved significantly by variational updates. *Top Right:* Topic pairs accepted by merge moves during run on Wikipedia. Combining each pair into one topic improves our objective $\mathcal{L}$, saves space, and removes redundancy. *Bottom Row:* Accepted delete move during run on Wikipedia. Red topic is rarely used and lacks semantic focus. Removing it and reassigning its mass to remaining topics improves $\mathcal{L}$ and interpretability.

Local construction. We use the deterministic addition in Eq. (4.1) for creating $\hat{r}'$ from $\hat{r}$. Similarly, we have a deterministic addition for creating $\hat{\theta}'$ from $\hat{\theta}$. For each document d, we have:

$$\text{for } k = 1, 2, \dots K-1 : \hat{\theta}'_{dk} = \begin{cases} \hat{\theta}_{dk_A} + \hat{\theta}_{dk_B} & \text{if } k = k_A \\ \hat{\theta}_{dk} & \text{else if } k < k_B \\ \hat{\theta}_{dk+1} & \text{else if } k \geq k_B \end{cases} \tag{5.49}$$

We also keep the pseudo-count value for the inactive topics $\hat{\theta}_{d>K}$ the same.

Summary statistic construction. As in DP mixture models, the statistics $N_k(\hat{r}')$ and $S_k(\hat{r}', x)$ can be constructed easily given their original values, as in Eq. (4.2). We also must compute the entropy of the assignment posterior $H(\hat{r}')$, which again is the same computation as the DP mixture model.

The summary statistics of $\hat{\theta}$, such as the aggregate log probabilities $P(\hat{\theta})$ used for the global step and the terms $Q(\hat{\theta})$ needed for the evaluation of $\mathcal{L}_{\text{HDP-doc}}$, need to be computed specifically for each merge pair k_A, k_B, because these are non-linear functions of $\hat{\theta}$.

Global construction. As in DP mixtures, given the summary statistics for the merged local parameters, we can create proposed global parameters simply by executing the appropriate global parameter optimization steps.

Selecting candidate cluster pairs to try merging

Past work (Bryant and Sudderth, 2012; Hughes et al., 2015a) suggests that one viable way to select candidate pairs is to use the empirical correlation of two candidate topics across all document-specific count vectors $\{N_d\}_{d=1}^D$.

$$\text{score}(k_A, k_B) = \text{Corr}(N_{:k_A}, N_{:k_B}), \quad -1 < \text{score} < 1. \tag{5.50}$$

Large scores identify topic pairs frequently used in the same documents, which may be a useful signal for potential merges.

While useful, we now recommend a version of the selection score used for DP mixture models, which uses information about the change in $\mathcal{L}_{\text{data}}$ under a proposal to suggest which terms to track.

5.5.2 Delete proposals

Delete moves provide a more powerful alternative to merges for removing rarely used "junk" topics. For an illustration of an accepted delete proposal in the context of training a topic model on Wikipedia data, see Fig. 5.5. After identifying a candidate topic with small mass to delete, we reassign *all* its tokens to the remaining topics. This move can succeed when a merge would fail because each document's tokens can be reassigned in a customized, document-specific fashion, as shown in Fig. 5.5.

Our first work on deletes for topic models was published in an AISTATS '15 conference paper (Hughes et al., 2015a). We reproduce the ideas from that paper on delete proposals here. First, we outline how a delete move would work if we could afford explicitly updating all documents in the dataset. Next, we describe how deletes work in our memoized framework, where we use a *heuristic* delete proposal in the sense that it did not construct valid parameters $\hat{r}', \hat{\theta}'$ which represented the full dataset, or evaluate a whole-dataset objective function. Nevertheless, we find decent performance in practice.

Whole-dataset delete construction

Delete moves remove some topic, indexed by j, from a current set of parameters and sufficient statistics of size $K+1$. For simplicity, this explanation assumes j is last in index order, but in fact j can be at any position. The move constructs new parameters and new sufficient statistics S', N' of

size K, where any mass assigned to topic j has been reallocated among the other topics. Here, unlike the merge move, we have no one-step rule for constructing the candidate local parameters. Instead, we use a heuristic initialization followed by refinement coordinate ascent updates. We initialize sufficient statistics by simply removing any entries associated with topic j.

$$\begin{aligned} \text{Original: } N &= [N_1 \quad \dots \quad N_K \; N_j] \\ \text{Candidate Init: } N' &= [N_1 \quad \dots \quad N_K] \end{aligned} \tag{5.51}$$

Given these initial summaries, we take a global step to create K candidate global parameters. The main paper's Fig. 1 reviews the big picture for how summary statistics lead to global parameter updates. After creating the candidate global parameters, we realize that $\hat{\tau}'$ will have exactly the same first K topics as the original model. For $\hat{\rho}', \hat{\omega}'$, the resulting $\mathbb{E}[\beta]$ will be similar, too. Next, a local step reassigns *all* tokens (including those ignored) among the K remaining topics. After one more global step, we have a viable candidate model q' representing the whole dataset. This model can be compared to the original, and kept if the objective improves.

Memoized delete construction

For large datasets, it is infeasible to perform several local/global update cycles for all documents just to evaluate one candidate move. A more scalable delete move is possible because we assume junk topic j has only a small subset of documents with appreciable mass, while most documents assign $N_{dj} \approx 0$. Thus, only the small set satisfying $N_{dj} > \epsilon$ need to be edited explicitly, where we set $\epsilon = 0.01$.

The memoized delete move happens in three steps. First, we gather all documents satisfying this threshold test into a target dataset during a standard pass of the dataset. Second, we construct the delete candidate model q' from the target set, performing the simple construction described above while holding the non-target sufficient statistics fixed. That is, for each additive sufficient statistic vector N, S, T in the previous model, we create candidates N', S', T' that satisfy the following relation:

$$N'_k = N_k - N_k^{before} + N_k^{after}, k \in \{1 \dots K\} \tag{5.52}$$

Here, N_k^{before} is the statistic for topic k on the target set before removing j, and N_k^{after} is the computed statistic on the target set after removing j and performing the several updates.

For the specific case of the token count statistic N' on the target set, we know that $N_\epsilon + \sum_{k=1}^K N'_k = N_j + \sum_{k=1}^K N_k$ where N_ϵ represents the small mass assigned to j from documents that did not pass the threshold test. If accepted, sufficient statistic vector N' will soon accurately reflect all data (including the small discarded mass) after a complete pass of local and global steps at all batches.

To determine acceptance, we evaluate the objective $\mathcal{L}(\cdot)$ using candidate global parameters $\hat{u}', \hat{\omega}', \hat{\tau}', \hat{\nu}'$, which are obtained via direct updates from N', S', P'. For the local arguments to $\mathcal{L}(\cdot)$, we use the inferred parameters $\hat{r}_d, \hat{\theta}_d$ from documents in the target set.

If the candidate model improves this objective, we accept it. After accepting, we need to adjust all stored batch-specific summaries to reflect the new model. Otherwise, our new aggregate summaries will not be consistent with the sum of stored batch summaries, and subsequent incremental updates will be invalid. We thus edit the stored statistics for each batch to reflect the final state of the target-set documents from that batch.

Immediately after a delete move, we do not have the required ELBO summaries to exactly compute the bound after visiting the next batch. However, after completing a complete lap through all batches, the relevant summaries will be refreshed and the ELBO computable.

Selecting topics to delete

Delete move costs scale with the number of documents in the target set. We specify a maximum cap for the total documents we can afford to process as a target set as 500. Any topic occuring in fewer than 500 documents is eligible for deletion. We select among this eligible set as many topics as possible until the total cap is reached, and build the target set as the union of all documents passing the threshold test for any selected topic. This allows potentially *multiple* topics to be deleted in one pass through the data, each one considered independently, while never exceeding the specified cap on target set size.

5.5.3 Birth proposals

HDP topic models have two sets of local parameters that come from every proposal: assignments $\hat{r}'$, which has one vector per data atom, and probability pseudo-counts $\hat{\theta}'$, which has one vector per document d. As in DP mixtures, we can easily handle create coherent summaries $N(\hat{r}), S(\hat{r})$ of assignments across batches with different truncation levels by inserting zeros. However, each batch requires a summary $P(\hat{\theta})$ which is a non-linear function of the pseudo-count local parameters:

$$P_{dk}(\hat{\theta}) = \mathbb{E}_q[\log \pi_{dk}] = \psi(\hat{\theta}_{dk}) - \psi(\hat{\theta}_{d\cdot}), \qquad P_{bk} = \sum_{d \in \mathcal{D}_b} P_{dk} \tag{5.53}$$

Each entry P_{bk} can be interpreted as an aggregated log probability. If topic k is popular across batch b, the value of P_{bk} will be a small magnitude negative number, while as topic k becomes more rare, the value of P_{bk} will grow toward negative infinity.

To build intuition for birth proposals with an HDP allocation model, we consider an example for the d-th document. Below, we show possible before and after values for four interrelated variational parameter vectors: the expected global topic probabilities π^G, aggregated assignment counts $N(\hat{r}_d)$, variational parameters $\hat{\theta}_d$, and log probabilities P_d. We assume the original model has 3 active clusters, and the proposal splits the third cluster into two new active clusters, which are placed last

in stick-breaking order.

$$\begin{aligned}
\text{BEFORE}: \quad \pi^G &= \texttt{[0.4, 0.3, 0.2]}, \texttt{0.1} \qquad (5.54)\\
N(\hat{r}_d) &= \texttt{[10.0, 50.0, 5.0]}\\
\hat{\theta}_d &= \texttt{[10.4, 50.3, 5.2]}, \texttt{0.1}\\
P(\theta_d) &= \texttt{[-1.889, -0.274, -2.633]}\\
U(\theta_d) &= \texttt{[0.000, 0.000, -14.606]}\\
\text{AFTER}: \quad \pi^{'G} &= \texttt{[0.4, 0.3, 0.02, 0.09, 0.09]}, \texttt{0.1}\\
N(\hat{r}'_d) &= \texttt{[10.0, 50.0, 0.00, 3.00, 2.00]}\\
\hat{\theta}'_d &= \texttt{[10.4, 50.3, 0.02, 3.09, 2.09]}, \texttt{0.1}\\
P(\theta'_d) &= \texttt{[-1.889, -0.274, -54.727, -3.224, -3.703]}\\
U(\theta'_d) &= \texttt{[0.000, 0.000,   0.000, 0.000, -14.606]} \qquad (5.55)
\end{aligned}$$

The first step of constructing this proposal is creating a temporary value for probability vector π^G. Our current variational state defines an approximate posterior $q(u)$ which implies a distribution on π^G. We begin by setting π^G to its expected value under this posterior. This gives us a concrete vector of size K, with one entry for each of the active clusters. Remember that we also have one extra entry that aggregates the mass of all inactive clusters. Next, we create an expanded vector $\pi^{'G}$ of size $K + J'$ by copying π^G but redistributing the original target cluster mass uniformly among the new J' clusters. The proposed value for $\pi^{'G}$ leaves only a small fraction ϵ of mass at the target cluster, and keeps the inactive mass unchanged.

Next, we provide this specific $\pi^{'G}$ as input to the Bregman k-means++ and restricted local step process, which delivers specific values for assignments $\hat{r}'_d$ at each data unit and their associated per-document summaries $N(\hat{r}'_d)$. Given specific values of $N(\hat{r}'_d)$ and $\pi^{'G}$, the local parameter vector $\hat{\theta}'_d$ has a closed-form optimal value via the local-step formula: $\hat{\theta}'_{dk} = N'_{dk} + \alpha\pi^{'G}$. In the example above, we assumed $\alpha = 1$. Both summary vectors P' and U' are easily computed given $\hat{\theta}'$.

We may run this proposal process at every document d inside a batch b. This yields the summaries N'_b, S'_b, P'_b, U'_b which then naturally can be aggregated and used to construct the global parameters via Step 2 of Fig. 4.2.

Multi-batch summary tracking for HDP topic models. With more than once batch $B > 1$, we need to effectively track the whole-dataset summaries P^G, U^G that represents batches at different truncation levels. We cannot simply insert zeros to the vector P^G, because these vectors represent log probabilities, not counts. In fact, there is no finite constant that we can insert in the log probability domain to represent unassigned mass.

Rather than perform an operation to expand vectors to one consistent truncation level, we instead embrace the idea that different truncation levels can exist at different data units or batches simultaneously. We assign all data units in batch b to the same truncation K_b. We find we can represent this via coherent whole-dataset statistics P^G and U^G, each a vector of size $K = \max_b K_b$.

The vector P^G represents active log probabilities, and U^G tracks the inactive log probability mass. They are formally defined as:

$$P^G = \sum_{b:K_b \geq k} P_{bk}(\hat{\theta}) = \sum_{d:K_d \geq k} \psi(\hat{\theta}_{dk}) - \psi(\hat{\theta}_{d:}) \tag{5.56}$$

$$U^G = \sum_{b:K_b = k} U_{bk}(\hat{\theta}) = \sum_{d:K_d = k} \psi(\hat{\theta}_{d,>k}) - \psi(\hat{\theta}_{d:}) \tag{5.57}$$

The aggregated vector $U_{:}$ will have non-zero entries at each unique truncation level existing in some current batch. After a proposal at batch b that transforms P_{bk} into P'_{bk}, we can create the new whole-dataset vector $P'_{:}$ using the following updates for each possible cluster index $k \in 1, \ldots K + J'$:

$$P'_{:k} = \begin{cases} P_{:k} - P_{bk} + P'_{bk} & \text{if } k = k_{\text{target}} \\ P_{:k} & \text{else if } k \leq K \\ P'_{bk} & \text{else if } k > K \end{cases} \qquad U'_{:k} = \begin{cases} U_{:k} - U_{bk} + U'_{bk} & \text{if } k = k_{\text{target}} \\ U_{:k} & \text{else if } k \leq K \\ U'_{bk} & \text{else if } k > K \end{cases} \tag{5.58}$$

Together, the proposal vectors P'^G, U'^G accurately represent the truncation status of every batch in the entire dataset. As usual, these vectors are sufficient statistics for a global update for allocation model parameters.

5.6 Experimental results

Our experiments compare inference methods for fitting HDP topic models. For our new HDP objective, we study stochastic with fixed K (SOfix), memoized with fixed K (MOfix), and memoized with deletes and merges (MOdm). For baselines, we consider the collapsed sampler (Gibbs) of Teh et al. (2006), the stochastic CRF method (crfSOfix) of Wang et al. (2011), and the stochastic split-merge method (SOsm) of Bryant and Sudderth (2012). For each method, we perform several runs from various initial K values.

For each run, we measure its predictive power via a heldout document completion task, as in Bryant and Sudderth (2012). Each model is summarized by a point-estimate of the topic-word probabilities ϕ. For each heldout document d we randomly split its word tokens into two halves: x'_d, x''_d. We use the first half to infer a point-estimate of π_d, then estimate log-likelihood of each token in the second half x''_d.

$$\text{heldout-lik}(x|\phi) = \frac{\sum_{d \in \mathcal{D}_{test}} \log p(x''_d | \pi_d, \phi)}{\sum_{d \in \mathcal{D}_{test}} |x''_d|} \tag{5.59}$$

Hyperparameters. In all runs, we set $\gamma = 10$, $\alpha = 0.5$ and topic-word pseudocount $\bar{\tau} = 0.1$. Stochastic runs use the learning rate decay recommended in Bryant and Sudderth (2012): $\kappa = 0.5, \delta = 1$.

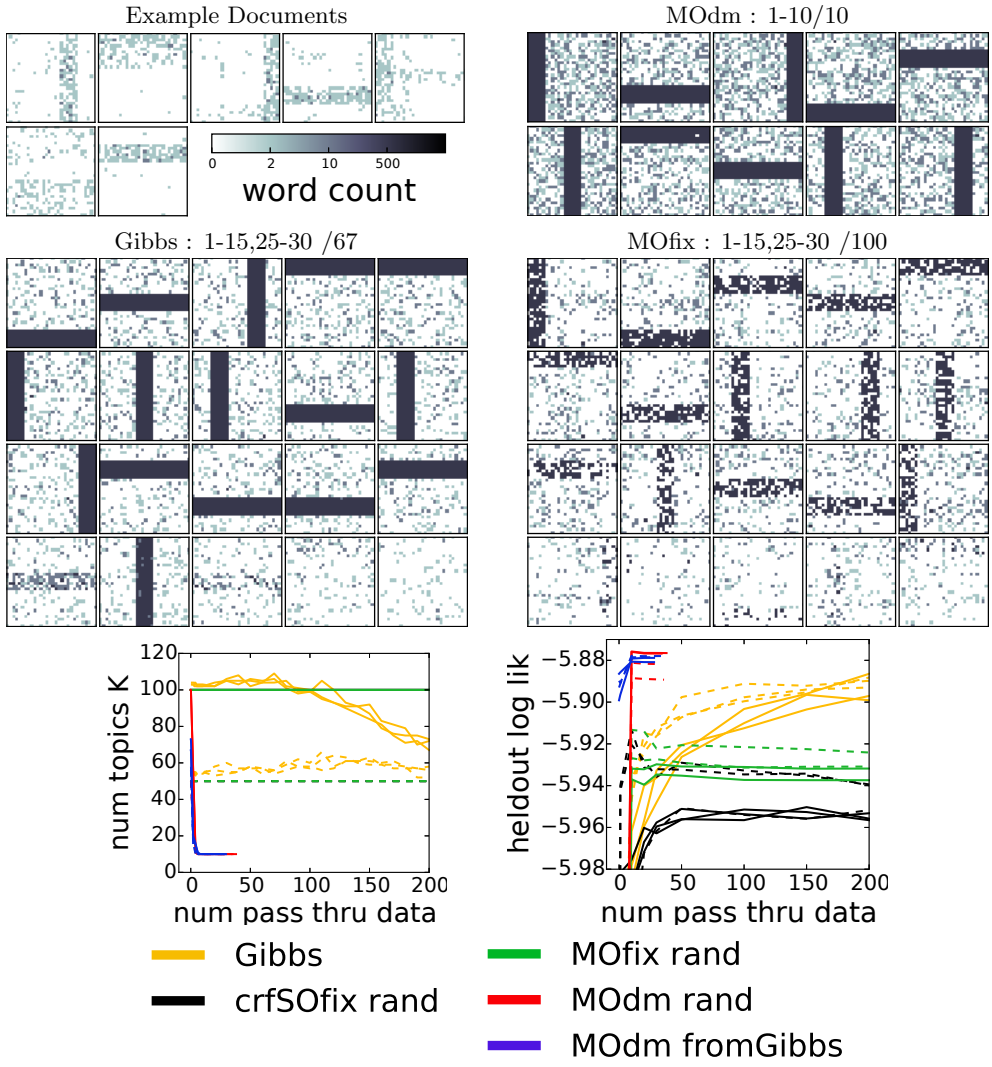

Figure 5.6: Comparison of HDP topic model inference methods on toy bars dataset. Comparison of inference methods on toy bars dataset from Sec. 5.6.1. *Top row:* Word count images for 7 example documents and the final 10 estimated topics from MOdm. Each image shows all 900 vocabulary types arranged in square grid. *Middle row:* Final estimated topics from Gibbs sampler and fixed truncation memoized (MOfix). We rank each algorithm's final set of topics from most to least probable in terms of global appearance probability π_k^G, and show the topics ranked 1-15 and 25-30. *Bottom row:* Trace plots of the number of topics K and heldout likelihood during training. Line style indicates number of initial topics: dashed is $K = 50$, solid is $K = 100$.

5.6.1 Toy bars dataset

We study a variant of the toy bars dataset of Griffiths and Steyvers (2004), shown in Fig. 5.6. There are 10 ideal bar topics, 5 horizontal and 5 vertical. The bars are noisier than the original and cover a larger vocabulary (900 words). We generate 1000 documents for training and 100 more for heldout test. Each one has 200 tokens drawn from 1-3 topics.

Fig. 5.6 shows many runs of all algorithms on this benchmark. Variational methods initialized with 50 or 100 topics get stuck rapidly, while the Gibbs sampler finds a redundant set of the ideal topics and is unable to effectively merge down to the ideal 10.

In contrast, our MOdm method uses merges and deletes to rapidly recover the 10 ideal bars after only a few laps. Without these moves, MOfix runs remain stuck at suboptimal fragments of bars. Furthermore, our MOdm method initialized with the sampler's final topics (fromGibbs) easily recovers the ideal bars.

5.6.2 Academic and news articles

Next, we apply all methods to papers from the NIPS conference, articles from Wikipedia, and articles from the journal Science (Paisley et al., 2011), with 80%-20% train-test splits. Online methods process each training set in 20 batches. Trace plots in Fig. 5.7 compare predictive power and model complexity as more data is processed. We summarize conclusions below.

Anchor topics are good; variational is better. Using the anchor word method (Arora et al., 2013) for initial topic-word parameters yields better predictions than random initialization (`rand`). However, our methods can still make big, useful changes from this starting point. See Fig. 5.5 for some examples.

Deletes and merges make big, useful changes. Across all 3 datasets in Fig. 5.7, merges and deletes remove many topics. On Wikipedia, we reduce 200 topics to under 100 while improving predictions. Similar gains occur from the final result of the Gibbs sampler.

Competitors get stuck or improve slowly. The Gibbs sampler needs many laps to make quality predictions. The CRF method gets stuck quickly, while our methods (using the direct assignment representation) do better from similar initializations. The stochastic split-merge method (SOsm) grows to a prescribed maximum number of topics but fails to make better predictions. This indicates problems with heuristic acceptance rules, and motivates our moves governed by exact evaluation of a whole-dataset objective.

Next, we analyze the New York Times Annotated Corpus: 1.8 million articles from 1987 to 2007. We withhold 800 documents and divide the remainder into 200 batches (9084 documents per batch). Fig. 5.7 shows the predictive performance of the more-scalable methods.

For this large-scale task, our direct assignment representation is more efficient than the CRF code released by Wang et al. (2011). With $K = 200$ topics, our memoized algorithm with merge and delete moves (MOdm) completes 8 laps through the 1.8 million documents in the amount of time the CRF code completes a single lap. No deletes or merges are accepted from any MOdm run,

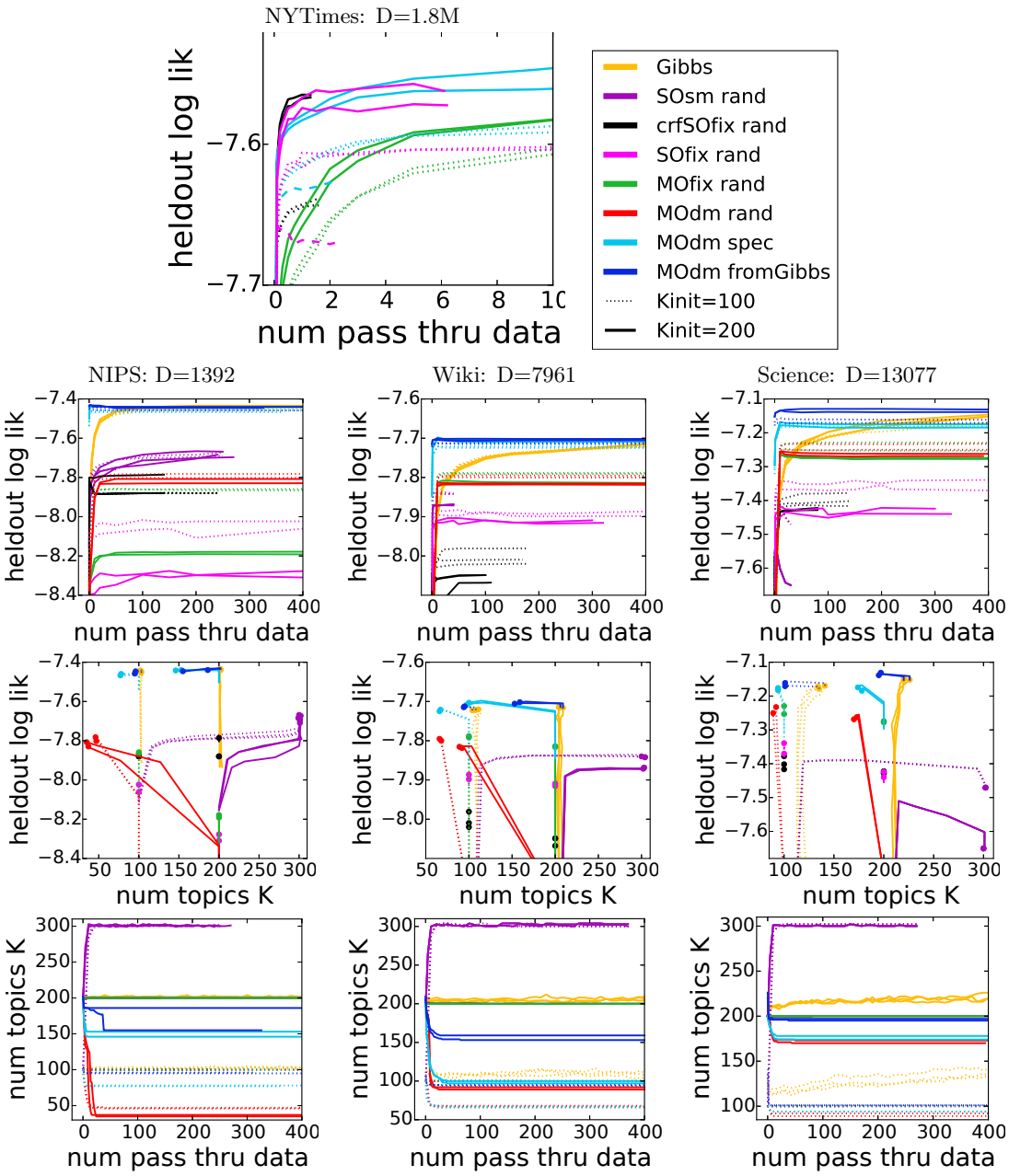

Figure 5.7: HDP topic model results on NIPS, Wikipedia, Science, and NYTimes datasets Trace plots of predictive performance and number of topics K from Sec. 5.6.2.

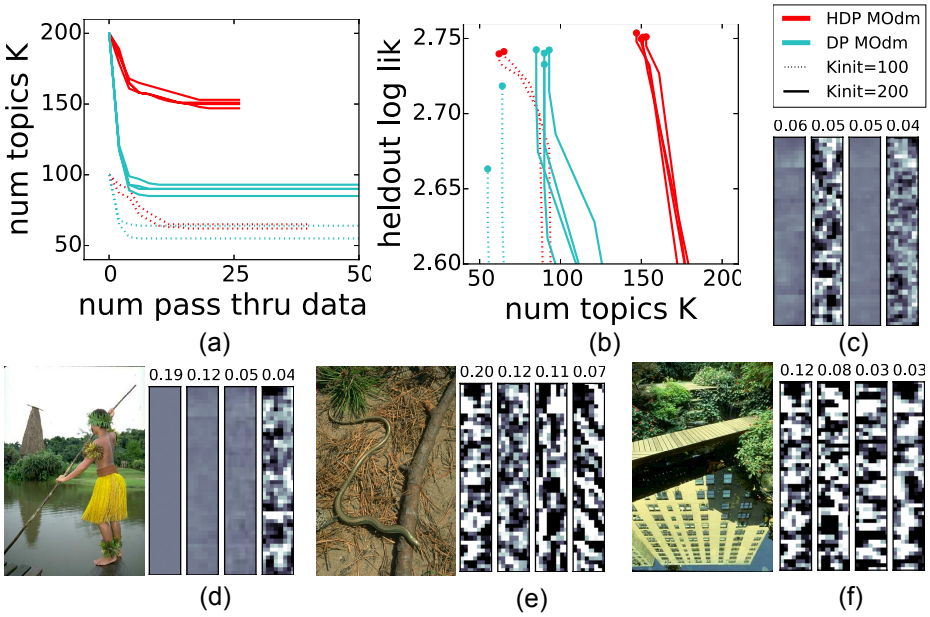

Figure 5.8: Comparison of DP mixtures and HDP admixtures on 3.5M image patches (a-b) Trace plots of number of topics and heldout likelihood. (c) Patches from the top 4 estimated DP clusters. Each column shows 6 stacked 8×8 patches sampled from one cluster. (d-f) Patches from 4 top-ranked HDP clusters for select test images from BSDS500 (Arbelaez et al., 2011). See Sec. 5.6.3 for details.

likely because 1.8M documents require more than a few hundred topics. However, the acceptance rate of sparsity-promoting restarts is 75%. With a more efficient, parallelized implementation, we believe our variational approach will enable reliable large-scale learning of topic models with larger K.

5.6.3 Image patch modeling

Finally, we study 8×8 patches from grayscale natural images as in Zoran and Weiss (2012). We train on 3.5 million patches from 400 images, comparing HDP admixtures to Dirichlet process (DP) mixtures using a zero-mean Gaussian likelihood. The HDP model captures within-image patch similarity via image-specific mixture component frequencies. Both methods are evaluated on 50 heldout images scored via Eq. (5.59).

Fig. 5.8 shows merges and deletes removing junk topics while improving predictions, justifying the generality of these moves. Further, the HDP earns better prediction scores than the DP mixture. We illustrate this success by plotting sample patches from the top 4 topics (ranked by topic weight π) for several heldout images. The HDP adapts topic weights to each image, favoring smooth patches for some images (d) and textured patches for others (e-f). The less-flexible DP must use the same weights for all images (c).

5.7 Discussion

Variational inference for the HDP topic model faces numerous challenges not present in the DP mixture model: the non-conjugacy in the model requires our new surrogate optimization objective, while the presence of multiple random variables $\hat{r}_d, \pi_d$ in every document d leads to harsh local optima problems even when the global parameters are known. We have developed a full-dataset inference algorithm which can be scaled up with both memoized and stochastic techniques. Our birth, merge, and delete proposals offer ways to escape some bad local optima and deliver more reliable clustering results.

Several issues remain open for the topic model, including the best way to initialize multinomial topic models and removing some of the slow information propagation between document-level frequency posteriors $q(\pi_d)$ and the overall top-level frequency posterior $q(\pi^G(u))$. We hope our work is a first step toward reliable large-scale training of topic models.

Chapter 6

Scalable variational inference for HDP hidden Markov models

The hidden Markov model (HMM) (Rabiner, 1989) has long been a fundamental building block of modern unsupervised learning for sequential data, such as the motion capture segmentation task of Fig. 1.4 or the chromatin segmentation task of Fig. 1.5. Here, we consider a Bayesian nonparametric version of the hidden Markov model, with a hierarchical prior which shares statistical strength across the transition parameters π_k specific to each cluster k.

Beal et al. (2001) introduced an early version of this model called the "infinite HMM", which was later formalized with the hierarchical Dirichlet process as the HDP-HMM by Teh et al. (2006). Later, Fox et al. (2011) developed the "sticky" parameterization of the HDP-HMM, which encourages models to have larger self-transition probabilities to better match empirical data. Recently, (Johnson and Willsky, 2014) developed a stochastic variational inference algorithm for HDP-HMMs, as well as more complex models with. However, while scalable, this approach did not investigate proposal moves that adapt the number of clusters and used a point estimate for some top-level allocation model parameters.

In this chapter, we expand on earlier work published with collaborators William Stephenson and Erik Sudderth in the NIPS 2015 conference (Hughes et al., 2015b). In particular, the contributions of this work are:

Contribution 1: New optimization problem for the HDP-HMM with model selection capability. The sticky parameterization of the HDP-HMM results in a required additional surrogate bound beyond that used for the HDP topic model in Ch. 5. Previous work (Johnson and Willsky, 2014) used a point estimation strategy for π^G (equivalently for u) which is problematic for model selection. In contrast, by placing proper approximate posterior distributions on all random variables, our surrogate bound yields effective model selection as we demonstrate in experiments.

Contribution 2: Scalable memoized algorithm for the HDP-HMM. Previously, only stochastic algorithms existed for this model. Our memoized approach offers a useful alternative which avoids the sensitivity to learning rate schedules required by the stochastic approach.

Contribution 3: Birth, merge, and delete proposals to escape local optima. We show that adaptive proposals which add or remove clusters offer improvements in both model quality (as measured by our objective function) and in application-specific metrics like Hamming distance. Our extensive experiments show promising results across speaker diarization, motion capture, and epigenomic segmentation tasks.

Roadmap. We first specify the model, focusing on the interesting model features relative to the HDP topic model: the sequential structure of $p(z|\pi)$ and the sticky parameterization of the hierarchy of transition probabilities. Next, we set up the variational optimization problem and derive complete expressions for the objective function. Later, we give complete descriptions of coordinate ascent updates as well as both full-dataset and memoized algorithms. We follow with experimental results applying these training algorithms on several datasets, and conclude with discussion of open problems of interest to future work.

6.1 Hierarchical Dirichlet Process Hidden Markov Models

We wish to jointly model data from D separate sequences, where sequence d has data $x_d = [x_{d1}, x_{d2}, \ \ldots \ , x_{dT_d}]$ and observation x_{dt} is a vector representing some measurement at timestep t. We assume that these timesteps occur at regular intervals. For example, vector x_{dt} could be the spectrogram of an instant of audio, or the sensed joint positions of a human subject during a 100ms interval of the motion capture application illustrated in Fig. 1.4.

The HDP-HMM explains this data by assigning each observation x_{dt} to a single hidden state z_{dt}. The chosen state comes from a countably infinite set of possible cluster labels $k \in \{1, 2, \ldots\}$. However, the states at neighboring timesteps are not drawn independently. Instead, we assume that the whole sequence $z_d = [z_{d1} z_{d2} \ldots z_{dt} \ldots z_{dT_d}]$ is generated via first-order Markovian dynamics. These are parameterized by initial state probabilities π_0 and transition probabilities $\{\pi_k\}_{k=1}^{\infty}$. These global parameters themselves have a common prior distribution, which means they are related hierarchically. The entire graphical model of directed dependence relationships between hidden variables is shown in Fig. 6.1.

6.1.1 Generative model for each sequence.

To generate each sequence, we require two sets of global variables. First, the global cluster shape parameters $\{\phi_k\}_{k=1}^{\infty}$ of the observation model, which we assume come from their common prior in Sec. 2.1.3. Second, the Markovian transition parameters, whose generative story we provide in Sec. 6.1.2.

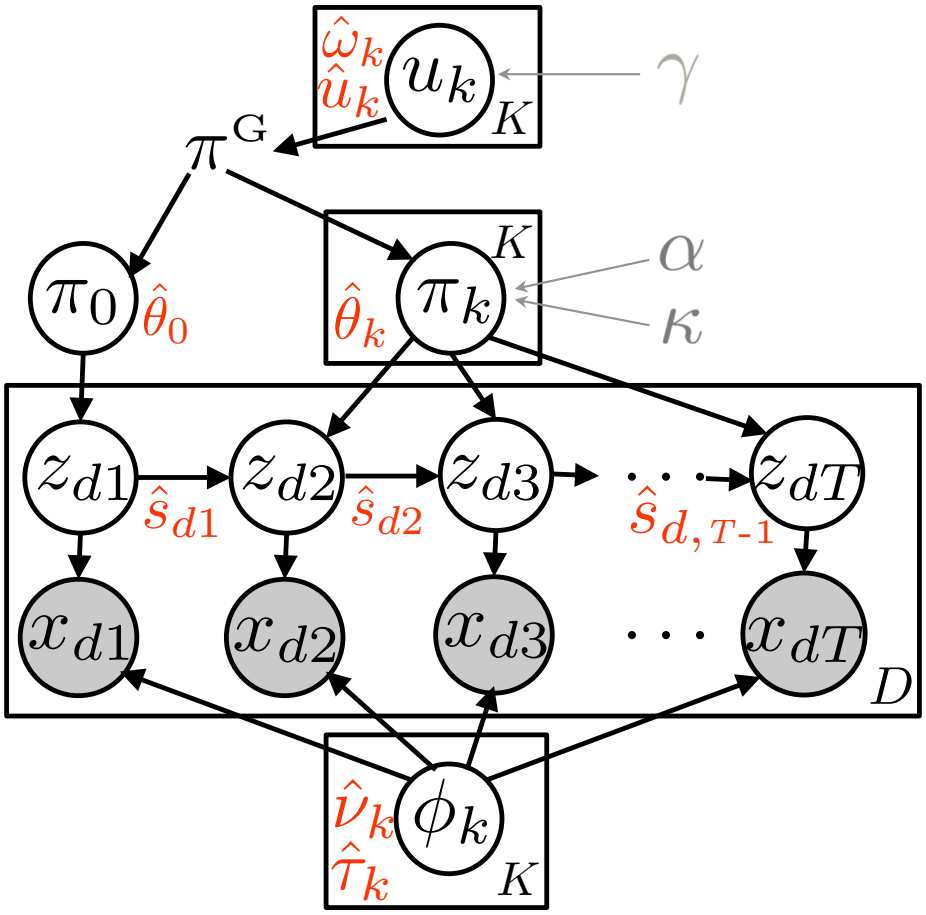

Figure 6.1: Directed graphical representation of the HDP hidden Markov model (HDP-HMM). The diagram shows the fundamental random variables (circled nodes), hyperparameters (gray), and variational free parameters (red) of the hierarchical Dirichlet process (HDP) hidden Markov model (HDP-HMM). This Bayesian nonparametric model generates a countably infinite number of clusters under the prior, of which some number $K \leq N$ are assigned to data. This model has the same global parameters for cluster conditional probabilities u and shape parameters ϕ as the mixture model, as well as a additional transition probabilities π_k and starting-state probabilities π_0. The conditional probabilities u are *deterministically* mapped to top-level global cluster probabilities π^G via the invertible stick-breaking transformation. The dataset x consists of N total observations divided into D sequences, and within a sequence we assume Markovian structure exists among the assigned label sequence $z_{d1}, z_{d2}, \ldots z_{dT}$. Our variational representation captures this with pair-wise joint probabilities $\hat{s}_{dt}$ for each timstep.

Given these global parameters, we allocate the state sequence z_d by first drawing its first timestep's assignment from the initial distribution π_0, and then recursively using the transition distribution specified by each chosen cluster at timestep $t-1$ to determine the cluster at timestep t:

$$z_{d1} \sim \text{Cat}_\infty(\pi_0), \qquad z_{d2}|z_{d1} \sim \text{Cat}_\infty(\pi_{z_{d1}}), \qquad z_{dt}|z_{dt-1} \sim \text{Cat}_\infty(\pi_{z_{d,t-1}}). \tag{6.1}$$

Given the assigned cluster label z_{dt} at timestep t, we then draw the observed data x_{dt} independently

of any other variable in the sequence using a cluster-specific likelihood:

$$x_{dt}|z_{dt} = k \sim \mathrm{L}(x_{dt}|\phi_k) \tag{6.2}$$

where L is our general exponential family likelihood distribution from Sec. 2.1.3.

First-order autoregressive models. As a small extension, we can allow the observation at timestep t to depend on the observation from timestep $t-1$. This first-order auto-regressive behavior is formally defined as:

$$x_{dt}|z_{dt} = k \sim \mathcal{N}(x_{dt}|A_k x_{dt-1}, \Sigma_k) \tag{6.3}$$

Such autoregressive likelihoods belong to the exponential family, admit conjugate priors, and are amenable to all our inference techniques. See Fox (2009) for a detailed discussion of these auto-regressive models, as well as MCMC training algorithms for these models. Bayesian posterior analysis of such autoregressive processes goes back at least to Quintana and West (1987). While the graphical model in Fig. 6.1 does not depict auto-regressive behavior explicitly, our inference procedure easily supports this extension.

6.1.2 Hierarchical prior on transition probabilities via the HDP.

Under the HDP-HMM prior and posterior, the number of clusters or states is unbounded. Each cluster k has its own infinite transition probability vector $\pi_k = [\pi_{k1}\ \pi_{k2}\ldots]$, which must be non-negative and sum-to-one. Without careful regularization, the sheer number of parameters in the infinite transition matrix π makes it very possible to overfit to training datasets. We impose such regularization by having each transition vector π_k share a common prior.

As in the HDP topic model, let π^G define a global probability vector over the infinite state space. We can write this quantity as finite vector representing K active topics and the aggregate mass of all inactive topics with index larger than K: $\pi^G = [\pi^G_1 \pi^G_2 \ldots \pi^G_K \pi^G_{>K}]$. Then, given this common set of probabilities, we generate the transition vector π_k for cluster k as:

$$\pi_k \sim \mathrm{Dir}(\alpha\pi^G_1, \alpha\pi^G_2, \ldots \alpha\pi^G_K, \alpha\pi^G_{>K}) \tag{6.4}$$

This regularizes the vector π_k by setting its mean to π^G but allowing some flexibility governed by the concentration $\alpha > 0$. Any setting of $\alpha < K$ will typically encourage the vector π_k to be sparse, with only a few entries with mass significantly greater than zero. Typically, we set $\alpha \approx 0.5$.

A similar prior is given to the starting state distribution π_0:

$$\pi_0 \sim \mathrm{Dir}(\alpha_0\pi^G_1, \alpha_0\pi^G_2, \ldots \alpha_0\pi^G_K, \alpha_0\pi^G_{>K}) \tag{6.5}$$

We generally set $\alpha_0 >> \alpha$ because very few starting states are observed and thus a smaller variance (greater concentration) is needed. Typically, we have $\alpha_0 \approx 10$.

This hierarchical relationship between the top-level probabilities π^G and the cluster-level probabilities π_k is formally a realization of the *hierarchical Dirichlet process* (Teh et al., 2006). The HDP

prior specification is complete by defining the stick-breaking prior on π^G. Given independent stick-breaking weights $u_k \sim \text{Beta}(1, \gamma)$, for each cluster, we use the stick-breaking transform of Eq. (3.4) to completely determine the K active entries of π^G:

$$\pi_1^G(u) = u_1, \qquad \pi_k^G(u) = u_k \prod_{\ell=1}^{k-1}(1 - u_\ell). \tag{6.6}$$

6.1.3 Sticky self-transition bias.

In many applications of clustering for time-series or sequential data, we expect assigned clusters to persist for many timesteps. For example, a human may walk for several minutes before switching to another activity, or the genome may exhibit long inactive segments in between transcription sites. The "sticky" parameterization of the HDP-HMM introduced by Fox et al. (2011) places extra mass on each cluster's self-transition probability π_{kk} compared to the conventional prior:

$$[\pi_{k1} \ \ldots \ \pi_{k>K}] \sim \text{Dir}(\alpha\pi_1^G, \ldots, \alpha\pi_{k-1}^G, \ \alpha\pi_k^G + \kappa, \ldots \alpha\pi_{>K}^G) \tag{6.7}$$

Here, the scalar hyperparameter $\kappa > 0$ controls the degree of self-transition bias. Choosing $\kappa >> 1$ allows the model to place much higher probability on long periods of high self-transitions in observed sequences. Alternatives that use explicit duration distributions or related semi-Markovian models have been thoroughly discussed in Johnson and Willsky (2014), but we find the sticky parameterization to be effective while also affordable.

Global and local variables

By inspection of Fig. 6.1, the only local (data-attached) hidden variables in this model are the sequence of assignments z_d for every sequence d. We recognize several global parameters for the HDP-HMM. First, the cluster shape parameters $\{\phi_k\}_{k=1}^K$ remain the obvious global parameters of the observation model, just like in the HDP topic model. Second, the allocation model includes several global parameters: the top-level conditional probabilities $\{u_k\}_{k=1}^K$ as well as the Markovian dynamics probabilities: starting vector π_0 and transition vectors $\{\pi_k\}_{k=1}^K$.

6.2 Posterior inference as a variational optimization problem

6.2.1 Mean-field approximate posterior

Our ideal goal in posterior inference is to estimate the joint distribution $p(u, \pi, \phi, \{z\}_{d=1}^D | x_{d=1}^D)$ given D observed sequences. However, because this joint posterior is intractable we appeal to mean-field variational methods to find the closest member of the factorized family:

$$q(u, \pi, \phi, z) = \prod_{k=1}^{\infty} q(u_k) \cdot q(\pi_0) \cdot \prod_{k=1}^{\infty} q(\pi_k) \cdot \prod_{k=1}^{\infty} q(\phi_k) \cdot \prod_{d=1}^{D} q(z_d) \tag{6.8}$$

$$q(\phi) = \prod_{k=1}^{\infty} \mathrm{P}(\phi_k | \hat{\tau}_k, \hat{\nu}_k) \tag{6.9}$$

$$q(u) = \prod_{k=1}^{\infty} \mathrm{Beta}(u_k | \hat{u}_k \hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \tag{6.10}$$

$$q(\pi_0) = \mathrm{Dir}(\pi_0 | \hat{\theta}_{01}, \hat{\theta}_{02}, \ldots \hat{\theta}_{0K}, \hat{\theta}_{0>K}) \tag{6.11}$$

$$q(\pi_k) = \mathrm{Dir}(\pi_k | \hat{\theta}_{k1}, \hat{\theta}_{k2}, \ldots \hat{\theta}_{kK}, \hat{\theta}_{k>K}) \tag{6.12}$$

$$q(z) = \prod_{d=1}^{D} \mathrm{Cat}_{\infty}(z_{d1} | \hat{r}_{d1}) \prod_{t=2}^{T_d} \mathrm{Cat}_{\infty}(z_{dt} | \hat{s}_{z_{dt-1}}) \tag{6.13}$$

Our chosen factorization for q is similar to Johnson and Willsky (2014), but includes a proper approximate posterior for $q(u)$ rather than a point-estimate. Just like our earlier methods for the DP mixture and HDP topic model, we do not require any truncation assumption beyond that for $q(z)$ detailed below.

Approximate posterior factor for each state sequence

Our choice for the assignment sequence approximate posterior $q(z)$ is *not* a full mean-field approach, but rather keeps the Markov structure of each sequence d:

$$q(z_d) \triangleq \left[\prod_{k=1}^{\infty} \hat{r}_{d1k}^{\delta_k(z_{d1})} \right] \prod_{t=1}^{T_d-1} \prod_{k=1}^{\infty} \prod_{\ell=1}^{\infty} \left[\frac{\hat{s}_{ntk\ell}}{\hat{r}_{ntk}} \right]^{\delta_k(z_{nt})\delta_\ell(z_{n,t+1})} \tag{6.14}$$

This factor is defined by two related free parameters. First, free parameter $\hat{s}_{dt}$ defines the joint assignment probabilities for each pair of assignments $z_{dt}, z_{d,t+1}$ at adjacent timesteps. Formally, $\hat{s}_{dtk\ell} \triangleq q(z_{d,t+1} = \ell, z_{dt} = k)$. Thus, the parameter $\hat{s}_{dt}$ has infinitely many rows and columns, but its entries are all non-negative and must sum to one.

Next, the parameter vector $\hat{r}_{dt}$ defines the marginal probability of assignment at each timestep t. That is: $\hat{r}_{dtk} \triangleq q(z_{nt} = k)$. To form a valid probability distribution for the whole sequence, $q(z_d)$, both $\hat{r}_d$ and $\hat{s}_d$ vector must obey several constraints. First, at each timestep $t \in 1, 2, \ldots T_d - 1$ the adjacent pair joint probabilities must obey a sum-to-one constraint and non-negativity constraint:

$$\sum_{k=1}^{\infty} \sum_{\ell=1}^{\infty} \hat{s}_{dtk\ell} = 1, \qquad \hat{s}_{dtk\ell} \geq 0 \text{ for all } (k, \ell) \in \{1, 2, \ldots\} \times \{1, 2, \ldots\} \tag{6.15}$$

Second, the marginal vector $\hat{r}_{dt}$ at each timestep $t = 1 \ldots T_d$ must obey similar constraints, as well a direct constraint that entry k of $\hat{r}_{dt}$ equals the sum over row k in the joint probability matrix $\hat{s}_{dt}$:

$$\sum_{k=1}^{\infty} \hat{r}_{dtk} = 1, \qquad \hat{r}_{dtk} \geq 0 \text{ for all } k \in 1, 2, \ldots \tag{6.16}$$

$$\hat{r}_{dtk} = \sum_{\ell=1}^{\infty} \hat{s}_{ntk\ell} \text{ for all } k \in 1, 2, \ldots$$

Truncation to K active clusters.

Just like in HDP topic models and DP mixture models, we enforce the assumption that only the first K active clusters have any non-zero probability. All remaining inactive clusters with index $k > K$ must have zero mass and thus need not be explicitly represented. Thus, at each timestep t we need to specify the $K \times K$ matrix $\hat{s}_{dt}$ and the K-length vector $\hat{r}_{dt}$.

Approximate posterior $q(\phi_k)$ for global cluster shape

Like every other model we've discussed, each cluster k in our countably infinite set is given an independent posterior factor $q(\phi_k)$ for its shape parameter, which has form given by the conjugate prior density P and two global free parameters: pseudo-count $\hat{\nu}_k$ and shape parameter $\hat{\tau}_k$. For more information, see Sec. 2.1.3.

Approximate posterior $q(u_k)$ for global cluster frequencies

As in HDP topic models, each cluster k in our countably infinite set has an independent posterior factor $q(u_k)$. We give $q(u_k)$ a Beta distribution, parameterized so $\hat{u}_k$ gives the mean value of u_k and $\hat{\omega}_k$ gives the variance, just as in the HDP topic model from Ch. 5, especially Sec. 5.2.1.

Approximate posterior $q(\pi_k)$ for transition probabilities.

Each cluster k has a global parameter π_k indicating how probable a transition from k to any other cluster is. We learn an approximate posterior for π_k which is Dirichlet with $K+1$ parameters $\hat{\theta}_{k1}, \ldots \hat{\theta}_{kK}, \hat{\theta}_{k>K}$. We interpret each $\hat{\theta}_{k\ell}$ as a non-negative pseudo-count of how often we will transition from k to state ℓ. The starting state probability vector π_0 also has a similar $K+1$-length vector named $\hat{\theta}_0$.

Under this chosen approximate posterior, we have the expectations:

$$\mathbb{E}_q[\pi_{k\ell}] = \frac{\hat{\theta}_{k\ell}}{\sum_{j=1}^{K+1} \hat{\theta}_{kj}}, \qquad \mathbb{E}_q[\log \pi_{k\ell}] = \psi(\hat{\theta}_{k\ell}) - \psi(\textstyle\sum_{j=1}^{K+1} \hat{\theta}_{kj}) \tag{6.17}$$

6.2.2 Evidence lower-bound objective function

Given the assumed factorization over the global random variables $\phi, u, \{\pi_k\}_{k=0}^{\infty}$ and the local random variables z, we now set up an optimization problem to find the free parameters whose implied factorized posterior minimizes the KL divergence to the true posterior.

$$\underset{\hat{\tau},\hat{\nu},\ \hat{u},\hat{\omega},\ \hat{\theta},\ \hat{s}}{\arg\min}\ \mathrm{KL}\left(q\Big(\phi, u, \{\pi_k\}_{k=0}^{K}, \{z_d\}_{d=1}^{D} \mid \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}\Big) \Big\| p\Big(\phi, u, \{\pi_k\}_{k=0}^{K}, \{z_d\}_{d=1}^{D} \mid x\Big)\right) \tag{6.18}$$

Following the standard logic used in deriving the tractable optimization problems in Ch. 3 and Ch. 5, we have an equivalent optimization problem:

$$\underset{\hat{\tau},\hat{\nu},\ \hat{u},\hat{\omega},\ \hat{\theta},\ \hat{s}}{\arg\max}\ \mathcal{L}(x, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) \tag{6.19}$$

$$\mathcal{L}(x, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) \triangleq \log p(x) - \mathrm{KL}\left(q(\phi, u, \pi, z|\hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) \middle\| p(\phi, u, \pi, z|x)\right) \tag{6.20}$$

$$= \mathbb{E}_{q(\phi,u,\pi,z)}\Big[\log p(x, \phi, u, \pi, z) - \log q(\phi, u, \pi, z)\Big] \tag{6.21}$$

Under our chosen factorized posterior q, the expectation that defines $\mathcal{L}$ in the last line is computable via closed-form functions of the free parameters. This makes optimization possible.

Remembering that the marginal responsibilities $\hat{r}$ can be found as a deterministic function of the pairwise joint responsibilities $\hat{s}$, we can write this objective as a sum of two terms:

$$\mathcal{L}(x, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) = \mathcal{L}_{\text{data}}(x, \hat{\tau}, \hat{\nu}, \hat{r}(\hat{s})) + \mathcal{L}_{\text{alloc}}(\hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) \tag{6.22}$$

The data term $\mathcal{L}_{\text{data}}$ is computed just as it was for HDP topic models and DP mixture models in Eq. (2.95). We need not reproduce it here.

We will focus on evaluating the allocation model term. This term itself decomposes into three interpretable pieces:

$$\mathcal{L}_{\text{alloc}}(\hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}) = \mathcal{L}_{\text{entropy}}(\hat{s}) + \mathcal{L}_{\text{HDP-trans}}(\hat{s}, \hat{\theta}, \hat{u}) + \mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) \tag{6.23}$$

We now briefly define each term via expectations.

Entropy term of the assignment posterior $q(z)$

The entropy term is a simple function of the joint timestep probabilities $\hat{s}$.

$$\mathcal{L}_{\text{entropy}}(\hat{s}) \triangleq -\textstyle\sum_{d=1}^{D} \mathbb{E}_q\left[\log q(z_d)\right] \tag{6.24}$$

$$= \sum_{d=1}^{D} \sum_{k=0}^{K} \sum_{\ell=1}^{K} H_{dk\ell}(\hat{s}),$$

where for each state index $k = \{0, 1, \ldots K\}$ and receiving state index $\ell = \{1, 2, \ldots K, K+1\}$ we have:

$$H_{dk\ell}(\hat{s}_d) = \begin{cases} -\hat{r}_{dt1k} \log \hat{r}_{dt1\ell}, & \text{if } k = 0 \\ -\sum_{t=1}^{T_d - 1} \hat{s}_{dtk\ell} \log \frac{\hat{s}_{dtk\ell}}{\sum_{j=1}^{K} \hat{s}_{dtkj}} & \text{if } k = 1, 2, \ldots K \end{cases} \tag{6.25}$$

Transition-specific term

The HDP-HMM has a term similar to $\mathcal{L}_{\text{alloc}}$ which aggregates all pieces of the objective that are functions of $\hat{\theta}$. This term is:

$$\mathcal{L}_{\text{HDP-trans}}(\hat{s}, \hat{\theta}, \hat{u}) = \mathbb{E}_q\Big[\log p(z|\pi) + \sum_{k=0}^{K} \log \frac{p(\pi_k)}{q(\pi_k)}\Big] - \sum_{k=0}^{K} \mathbb{E}[c_{\text{Dir}}(\alpha_k \pi^G(u))] \tag{6.26}$$

$$= \sum_{k=0}^{K} -c_{\text{Dir}}(\hat{\theta}_{k1} \dots \hat{\theta}_{kK}\ \hat{\theta}_{k>K})$$

$$+ \sum_{k=0}^{K} \sum_{\ell=1}^{K+1} \Big(M_{k\ell}(\hat{s}) + \alpha_k \pi_k^G(\hat{u}) + \kappa\delta_k(\ell) - \hat{\theta}_{k\ell}\Big)\mathbb{E}_q[\log \pi_{k\ell}]$$

where c_{Dir} is the cumulant function of a Dirichlet distribution in Eq. (5.22), the expectation of $\log \pi_{k\ell}$ is given in Eq. (6.17), and we define the transition count summary statistic $M_{k\ell}$ to given the total number of expected transitions from cluster k to cluster ℓ:

$$M_{k\ell}(\hat{s}) = \begin{cases} \sum_{d=1}^{D} \hat{r}_{dt1\ell} & \text{if } k = 0 \\ \sum_{d=1}^{D} \sum_{t=1}^{T_d - 1} \hat{s}_{dtk\ell} & \text{if } k \in \{1, 2, \dots K\} \end{cases} \tag{6.27}$$

Surrogate lower bound for top-level term.

As in the HDP topic model, computing the expected value of $\mathbb{E}_{q(u)}[c_{\text{Dir}}(\pi^G(u))]$ under the chosen beta distribution form for $q(u)$ has no known closed form. We thus use the surrogate bound from Eq. (5.24) to write this term as a tractable function of u. Substituting this lower bound in and assuming a non-sticky HDP prior with $\kappa = 0$, we have:

$$\mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) \leq \mathbb{E}_q\left[\log \frac{p(u)}{q(u)}\right] + \sum_{k=0}^{K} \mathbb{E}_q\left[c_{\text{Dir}}(\alpha_k \pi^G(u))\right] \tag{6.28}$$

$$\mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) = K \log \alpha_0 + K^2 \log \alpha$$

$$+ \sum_{k=1}^{K} c_{\text{Beta}}(1, \gamma) - c_{\text{Beta}}(\hat{u}_k\hat{\omega}_k, (1 - \hat{u}_k)\hat{\omega}_k)$$

$$+ \sum_{k=1}^{K} \Big((K{+}1) + 1 - \hat{u}_k\hat{\omega}_k\Big)\mathbb{E}_q[\log u_k]$$

$$+ \sum_{k=1}^{K} \Big((K{+}1)(K{+}1{-}k) + \gamma - (1 - \hat{u}_k)\hat{\omega}_k\Big)\mathbb{E}_q[\log 1 - u_k]$$

This surrogate function can be evaluated in closed-form given $\hat{u}, \hat{\omega}$ free parameters and the hyper-parameters α, α_0, γ. For sticky values $\kappa > 0$, we need an alternative expression, which is derived in the next section.

6.2.3 Surrogate objective for sticky HDP-HMM

To consider the sticky parameterization of the HDP prior on the transition parameters $\{\pi_k\}_{k=0}^K$ given the global probabilities $\pi^G(u)$, a small change to the top-level surrogate term is required.

Previously, we had:

$$c_{\text{Dir}}(\alpha\pi^G(u)) \geq K\log\alpha + \sum_{\ell=1}^{K+1}\log\pi_\ell^G(u) \tag{6.29}$$

Now, when we compute the cumulant function of the vector $\alpha\pi^G(u)$ plus an extra point mass κ on the k-th entry of the vector. In the supplement of (Hughes et al., 2015b), we establish the following bound for any $\kappa > 0, \alpha > 0$:

$$c_D(\alpha\pi^G(u) + \kappa\delta_k) \geq K\log\alpha - \log(\alpha+\kappa) + \log(\alpha\pi_k^G(u)+\kappa) + \textstyle\sum_{\ell=1}^{K+1}{}_{\ell\neq k}\log(\pi_\ell^G(u)). \tag{6.30}$$

The term $\log(\alpha\pi_k^G(u)+\kappa)$ still has no tractable expectation under $q(u)$, so we apply a further bound given the concavity of the log function:

$$\log(\alpha\pi_k^G(u)+\kappa) \geq \pi_k^G(u)\log(\alpha+\kappa) + (1-\pi_k^G(u))\log\kappa. \tag{6.31}$$

$$= \log\kappa + \Big(\log(\alpha+\kappa) - \log\kappa\Big)\pi_k^G(u) \tag{6.32}$$

Combining Eqs. (6.30) and (6.31), then unpacking the vector $\pi^G(u)$ into a pure function of the u variables, and computing the whole expectation of $\sum_{k=0}^K c_{\text{Dir}}(\alpha_k\pi^G(u)+\kappa\delta_k)$, we have:

$$\mathcal{L}_{\text{surrogate-HDP-sticky-top}}(\hat{u},\hat{\omega}) \leq \mathbb{E}_q\left[\log\frac{p(u)}{q(u)}\right] + \mathbb{E}_q\left[c_{\text{Dir}}(\alpha_0\pi^G(u))\right] + \sum_{k=1}^K \mathbb{E}_q\left[c_{\text{Dir}}(\alpha_k\pi^G(u)+\kappa\delta_k)\right]$$

$$\begin{aligned}\mathcal{L}_{\text{surrogate-HDP-sticky-top}}(\hat{u},\hat{\omega}) &= K\log\alpha_0 + K^2\log\alpha + K\Big(\log(\kappa) - \log(\alpha+\kappa)\Big) \\ &+ \sum_{k=1}^K c_{\text{Beta}}(1,\gamma) - c_{\text{Beta}}(\hat{u}_k\hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \\ &+ \Big(\log(\alpha+\kappa) - \log(\kappa)\Big)\textstyle\sum_{k=1}^K \pi_k^G(\hat{u}) \\ &+ \sum_{k=1}^K \Big(K+1-\hat{u}_k\hat{\omega}_k\Big)\mathbb{E}_q[\log u_k] \\ &+ \sum_{k=1}^K \Big(K(K+1-k) + \gamma - (1-\hat{u}_k)\hat{\omega}_k\Big)\mathbb{E}_q[\log 1-u_k]\end{aligned} \tag{6.33}$$

Directly cross-referencing the non-sticky surrogate term in Eq. (6.28) and the sticky surrogate term in Eq. (6.33) shows that not much has changed and the overall complexity of evaluating the surrogate top-level objective remains the same.

6.3 Update steps for variational optimization

Using the objective functions defined above, we can write down the the constrained optimization problem for our free parameters $\hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}, \hat{s}$ given an observed dataset x and hyperparameters $\mathcal{H}$:

$$\underset{\hat{\tau},\hat{\nu},\hat{u},\hat{\omega},\hat{\theta},\hat{s}}{\arg\max} \quad \mathcal{L}_{\text{data}}(x,\hat{\tau},\hat{\nu},\hat{r}(\hat{s})) + \mathcal{L}_{\text{entropy}}(\hat{s}) + \mathcal{L}_{\text{HDP-trans}}(\hat{s},\hat{\theta},\hat{u}) + \mathcal{L}_{\text{surrogate-HDP-sticky-top}}(\hat{u},\hat{\omega}) \tag{6.34}$$

where the required constraints on the local free parameters for document d are:

$$\hat{s}_{dt} \geq 0 \text{ and } \sum_{k=1}^{K} \sum_{\ell=1}^{K} \hat{s}_{dtk\ell} = 1 \qquad \text{for } t = 1, 2, \ldots T_d \tag{6.35}$$

and the required constraints on global free parameters are

$$\begin{aligned} \hat{u}_k \in [0,1] \text{ and } \hat{\omega}_k \geq 0 \qquad & \text{for } k = 1, 2, \ldots \\ \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \qquad & \text{for } k = 1, 2, \ldots \\ \hat{\theta}_{jk} \geq 0 \qquad & \text{for } j, k \in \{0, 1, 2, \ldots\} \times \{1, 2, \ldots\} \end{aligned} \tag{6.36}$$

As with earlier models, we pursue a block-coordinate ascent algorithm, which proceeds in two steps, a *local* step and a *global* step. The local step updates the free parameters $\hat{s}_d$ for each sequence d while holding global parameters fixed. The global step updates the global free parameters given fixed (summary statistics of) local parameters. When applied iteratively, these steps are guaranteed to monotonically improve the whole objective $\mathcal{L}$ until it converges to a local optima. We emphasize again that due to the required surrogate bound term, which is $\mathcal{L}_{\text{surrogate-HDP-top}}$ when $\kappa = 0$ or $\mathcal{L}_{\text{surrogate-HDP-sticky-top}}$ for $\kappa > 0$, the objective function $\mathcal{L}$ we are optimizing is a strict lower bound of $\log p(x) - \text{KL}(q||p)$.

Below, we define the optimization problem solved by each step of the block-coordinate ascent algorithm. We then describe detailed solutions which solve each problem below in closed-form in the following sections.

6.3.1 Global parameter update step for observation model

Consider the optimization problem:

$$\arg\max_{\hat{\tau}, \hat{\nu}} \quad \mathcal{L}_{\text{data}}(x, \hat{\tau}, \hat{\nu}, \hat{r}) \qquad \text{subject to } \hat{\nu}_k \geq 0 \text{ and } \hat{\tau}_k \in \mathcal{M} \text{ for } k = 1, 2, \ldots. \tag{6.37}$$

Where we remember that for the HDP-HMM, the responsibilities $\hat{r}$ at each timestep are a deterministic function of the pair-wise responsibilities $\hat{s}$. The optimization problem in Eq. (5.32) is the same as for the observation model in the DP mixture model, found in Eq. (3.39). This, we can apply the solution from Eq. (3.41) directly.

$$\begin{aligned} \hat{\tau}_k^* &= S_k(x, \hat{r}) + \bar{\tau} \\ \hat{\nu}_k^* &= N_k(\hat{r}) + \bar{\nu} \end{aligned} \tag{6.38}$$

6.3.2 Global step for allocation model

The allocation model global step involves two substeps: (1) finding the optimal top-level parameters $\hat{u}, \hat{\omega}$, and (2) finding the optimal transition parameters $\hat{\theta}$. Below, we describe our solutions to each of these subproblems, and an approach to the complete problem.

Global step for transition parameters

We now consider finding the optimal $\hat{\theta}$ parameters for the problem: Consider the optimization problem

$$\arg\max_{\hat{\theta}} \quad \mathcal{L}_{\text{HDP-trans}}(\hat{s}, \hat{\theta}, \hat{u}) \tag{6.39}$$

$$\text{subject to } \hat{\theta}_{jk} \geq 0 \text{ for} \qquad j = 0, 1, 2, \ldots K, \quad k = 1, 2, \ldots K+1$$

Using standard exponential family mathematics, we find a closed-form solution for any value of $\kappa \geq 0$:

$$\hat{\theta}^*_{jk} = M_{jk}(\hat{s}) + \alpha \pi^G_k(\hat{u}) + \kappa \delta_j(k) \tag{6.40}$$

which naturally shares much in common with the update for document-level $\hat{\theta}$ parameters from the HDP topic model, but differs in the use of the pair-wise count statistics $M_{jk}(\hat{s})$ and the sticky hyperparameter κ.

Global step for top-level parameters

Consider the optimization problem

$$\arg\max_{\hat{u}, \hat{\omega}} \quad \mathcal{L}_{\text{HDP-trans}}(\hat{s}, \hat{\theta}, \hat{u}) + \mathcal{L}_{\text{surrogate-HDP-top}}(\hat{u}, \hat{\omega}) \tag{6.41}$$

$$\text{subject to } \hat{u}_k \in [0, 1] \text{ for} \qquad k = 1, 2, \ldots \tag{6.42}$$

$$\hat{\omega}_k \geq 0 \text{ for} \qquad k = 1, 2, \ldots \tag{6.43}$$

Due to non-conjugacy, we cannot update $\hat{u}, \hat{\omega}$ in closed form. However, we can apply the same strategies as in the HDPTopicModel. First, we have a heuristic update for $\hat{\omega}$, which follows the same derivation pattern as the similar update in Eq. (5.39)

$$\hat{\omega}_k = \begin{cases} (K+1)(K+2-k) + 1 + \gamma & \text{if } k \leq K \text{ and } \kappa = 0 \\ K(K+2-k) + 1 + \gamma & \text{if } k \leq K \text{ and } \kappa > 0 \\ 1 + \gamma & \text{if } k > K \end{cases} \tag{6.44}$$

Second, we have a numerical optimization update for $\hat{u}$. The objective changes slightly depending on whether we have a non-sticky model with $\kappa = 0$ or a sticky state transition model with $\kappa > 0$. For both cases, we provide the exact objective needed below. These functions can be provided to any numerical optimization technique to find an optimal $\hat{u}^*$ within the constraint set.

Function for $\hat{u}$ with non-sticky model $\kappa = 0$:

$$\begin{aligned} f(\hat{u}, \hat{\omega}, T(\hat{\theta}), \alpha, \alpha_0, \gamma) = &- \sum_{k=1}^{K} c_{\text{Beta}}(\hat{u}_k \hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \\ &+ \sum_{k=1}^{K} (K{+}1 + 1 - \hat{u}_k\hat{\omega}_k)[\psi(\hat{u}_k\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \sum_{k=1}^{K} ((K{+}1)(K{+}1{-}k) + \gamma - (1-\hat{u}_k)\hat{\omega}_k)[\psi((1-\hat{u}_k)\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \sum_{j=0}^{K} \alpha_j \sum_{k=1}^{K+1} \pi_k^G(\hat{u}) P_{jk}(\hat{\theta}) \end{aligned} \tag{6.45}$$

where we define the $K+1$-dimensional log probability statistic P_j for the starting state ($j = 0$) as well as each active state $1 \leq j \leq K$:

$$P_{jk}(\hat{\theta}) \triangleq \mathbb{E}_q[\log \pi_{jk}] = \psi(\hat{\theta}_{jk}) - \psi(\textstyle\sum_{\ell=1}^{K+1} \hat{\theta}_{j\ell}) \tag{6.46}$$

Function for $\hat{u}$ with sticky model $\kappa > 0$:

$$\begin{aligned} f(\hat{u}, \hat{\omega}, P(\hat{\theta}), \alpha, \alpha_0, \kappa, \gamma) = &- \sum_{k=1}^{K} c_{\text{Beta}}(\hat{u}_k \hat{\omega}_k, (1-\hat{u}_k)\hat{\omega}_k) \\ &+ \sum_{k=1}^{K} (K + 1 - \hat{u}_k\hat{\omega}_k)[\psi(\hat{u}_k\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \sum_{k=1}^{K} (K(K{+}1{-}k) + \gamma - (1-\hat{u}_k)\hat{\omega}_k)[\psi((1-\hat{u}_k)\hat{\omega}_k) - \psi(\hat{\omega}_k)] \\ &+ \sum_{j=0}^{K} \alpha_j \sum_{k=1}^{K+1} \pi_k^G(\hat{u}) P_{jk}(\hat{\theta}) \\ &+ (\log(\alpha+\kappa) - \log\kappa) \left(\textstyle\sum_{k=1}^{K} \pi_k^G(\hat{u}) \right) \end{aligned} \tag{6.47}$$

Complete global step for allocation model

Updating $\hat{u}$ requires the statistics $P(\hat{\theta})$, while updating $\hat{\theta}$ requires the quantity $\pi^G(\hat{u})$. Because of the interdependence of these updates, we recommend that if possible they be iterated several times during the complete *global step* to improve convergence. That is, given updated values of the assignments $\hat{s}$, we first update $\hat{\theta}$ via Eq. (6.40), and then alternate between $\hat{u}$ and $\hat{\theta}$ until convergence.

6.3.3 Local step to update assigned state sequence

The update for assignment resposibility parameters $\hat{s}_d$ for each sequence d can be processed independently. However, within a sequence d we need to find a jointly optimal value across all timesteps $\{\hat{s}_{dt}\}_{t=1}^{T_d}$ via dynamic programming (Beal, 2003). The forward-backward algorithm (Rabiner, 1989) takes two input parameters. First, for the d-th sequence we compute a matrix C_d which has T_d rows

and K columns. Each entry defines the expected likelihood of the data x_{dt} at timestep t under cluster k: $C_{dtk} = \mathbb{E}_q[\log p(x_{dt} \mid \phi_k)]$. Second, we require the log transition probability vector $\{P_j(\hat{\theta})\}_{j=0}^K$ for each possible active cluster as well as the starting state ($j = 0$). Given these values, the algorithm produces the optimal probabilities $\hat{s}_d$ for the sequence under the objective in Eq. (6.48).

$$\begin{aligned} \arg\max_{\hat{s}_d} \quad & \mathcal{L}_{\text{data}}(x_d, \hat{\tau}, \hat{\nu}, \hat{r}_d(\hat{s}_d)) + \mathcal{L}_{\text{entropy}}(\hat{s}_d) + \mathcal{L}_{\text{HDP-trans}}(\hat{s}_d, \hat{\theta}, \hat{u}) \\ & \text{subject to } \sum_{k,\ell} \hat{s}_{dtk\ell} = 1 \text{ and } \hat{s}_{dt} \geq 0 \text{ for } t = 1, 2, \dots T_d \end{aligned} \tag{6.48}$$

From $\hat{s}_d$, we can easily compute the marginal responsibilities $\hat{r}_d$ via summation.

Runtime cost. The dynamic programming required here has cost $\mathcal{O}(T_d K^2)$ for the d-th sequence, where K is the number of active clusters and T_d is the number of timesteps in the sequence. For efficiency, multiple sequences can be processed in parallel.

6.4 Variational algorithms with fixed number of topics

6.4.1 Full-dataset algorithm

A complete full dataset coordinate ascent algorithm is given in Alg. 5.3. It alternates between visiting all D sequences in the local step and then updating all the global free parameters in the global step. The runtime cost is dominated by the local step, where each dynamic programming has cost $\mathcal{O}(T_d K^2)$.

6.4.2 Memoized variational algorithm

The full dataset algorithm can be easily extended to process one small batch of sequences $\mathcal{D}_b$ during each local step instead of all D sequences. Assuming that all D sequences have been divided into B total batches before iterations begin, we can apply the memoized algorithm in Alg. 5.5.

Memoized algorithm and monotonicity guarantee

The memoized algorithm for training the HDP-HMM in Alg. 6.2 is guaranteed to monotonically increase the objective function $\mathcal{L}$ after both the global step and the local step. This is because unlike HDP topic models, which have multiple local variables, the HDP-HMM has only the local pairwise responsibilities $\hat{s}$. The forward-backward algorithm delivers *optimal* values for the local parameter $\hat{s}$ under a fixed set of global parameters. Thus, the local step is guaranteed to improve the objective $\mathcal{L}$.

Memoized evaluation of the objective

As in DP mixture models or HDP topic models, we can compute the objective score $\mathcal{L}$ for the current configuration of global parameters and global statistics of local parameters *after* visiting

Algorithm 6.1 Variational coordinate ascent for HDP-HMM

Input:

$\{x_d\}_{d=1}^D$: dataset with D sequences, each of length T_d
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: initial global parameters of allocation model defining $q(u)$
$\{\{\hat{\theta}_{jk}\}_{k=1}^{K+1}\}_{j=0}^K$: initial global parameters defining Markov transition probabilities
γ, α, α_0: allocation model hyperparameters
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: updated global parameters of allocation model
$\{\{\hat{\theta}_{jk}\}_{k=1}^{K+1}\}_{j=0}^K$: updated global parameters defining Markov transition probabilities.

function VariationalCoordAscentForHDPHMM($x, K, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}$)
  while not converged **do**
    for $d \in 1, 2, \ldots D$ **do** ▷ Local step at document d
      for $t \in 1, 2, \ldots T_d$ **do**
        for $k \in 1, 2, \ldots K$ **do**
          $C_{dak} \leftarrow \mathbb{E}_q[\log p(x_{da}|\phi_k)]$
      $\hat{s}_d, \hat{r}_d \leftarrow$ FwdBwdAlgForSeq($C_d(\hat{\tau}, \hat{\nu}), P(\hat{\theta})$)

    for $k \in 1, 2, \ldots K$ **do** ▷ Summary step
      $S_k \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk} s(x_{dt})$
      $N_k \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk}$
      $M_{0k} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{d1k}$
      for $j \in 1, 2, \ldots K$ **do**
        $M_{jk} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{s}_{dtjk}$

    for $k \in 1, 2, \ldots K$ **do** ▷ Global step for observation parameters
      $\hat{\tau}_k \leftarrow S_k + \bar{\tau}$
      $\hat{\nu}_k \leftarrow N_k + \bar{\nu}$

    while not converged **do**
      for $j \in 0, 1, 2, \ldots K$ **do** ▷ Global step for transition parameters
        for $k \in 1, 2, \ldots K+1$ **do**
          $\hat{\theta}_{jk} \leftarrow M_{jk} + \alpha_j \pi_k^G(\hat{u}) + \delta_k \kappa$
        for $k \in 1, 2, \ldots K+1$ **do**
          $P_{jk} \leftarrow \psi(\hat{\theta}_{jk}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{j\ell})$
      $\hat{u} \leftarrow \arg\max_{\hat{u}} f(\hat{u}, \hat{\omega}, P(\hat{\theta}), \alpha, \gamma, \kappa)$ ▷ Global step, via L-BFGS
  return $\hat{\tau}, \hat{\nu},\ \hat{u}, \hat{\omega},\ \{\hat{\theta}\}_{j=0}^K$

Full-dataset algorithm for approximate posterior inference for the HDP hidden Markov model shown in Fig. 6.1.

every batch at least once. This corresponds to completing the first epoch or lap of memoized inference. For this computation, the only auxiliary statistic we need is the entropy statistic of the pair-wise responsibilities $\hat{s}$.

Entropy statistic for ELBO computation. At batch b, let H_{bjk} define the sum of assignment entropies related to the transition between clusters j and k across all sequences in the batch:

$$\mathcal{L}_{\text{entropy}}(\hat{s}) = \sum_{j=0}^{K}\sum_{k=1}^{K} H^G_{jk}(\hat{s}), \quad H^G_{jk} = \sum_{b=1}^{B} H_{bjk}, \quad H_{bjk}(\hat{s}) \triangleq \sum_{d \in \mathcal{D}_b} H_{djk}(\hat{s}) \tag{6.49}$$

where the scalar entropy statistic H_{djk} for the assignment posterior $q(z_d)$ of sequence d is defined in Eq. (6.25). We compute this scalar for each possible source cluster $j \in \{0, 1, 2, \dots K\}$, including the starting state label $j = 0$, and for every possible destination cluster $k \in \{1, 2, \dots K\}$. Again, our assumed truncation directly prevents any transition to clusters beyond index K. These infinitely many inactive clusters all have zero probability mass and thus zero entropy.

6.5 Variational algorithms with proposal moves that adapt the number of clusters

6.5.1 Merge proposals

Merge proposals try to find a less redundant but equally expressive model. Each proposal takes a pair of existing states $i < j$ and constructs a candidate model where data from state j is reassigned to state i. Conceptually this reassignment gives a new value $\hat{s}'$, but instead statistics M', S' can be directly computed and used in a global update for candidate parameters $\hat{\tau}', \hat{\nu}', \hat{u}', \hat{\omega}', \hat{\theta}'$.

$$S'_i = S_i + S_j, \quad M'_{:i} = M_{:i} + M_{:j}, \quad M'_{i:} = M_{i:} + M_{j:}, \quad M'_{ii} = M_{ii} + M_{jj} + M_{ji} + M_{ij}.$$

While most terms in $\mathcal{L}$ are linear functions of our cached sufficient statistics, the entropy $\mathcal{L}_{\text{entropy}}$ is not. Thus for each candidate merge pair (i, j), we use $\mathcal{O}(K)$ storage and computation to track column $H'_{:i}$ and row $H'_{i:}$ of the corresponding merged entropy matrix H'. Because all terms in the H' matrix of Eq. (6.24) are non-negative, we can lower-bound $\mathcal{L}_{\text{entropy}}$ by summing a subset of H'.

This allows us to make coherent accept or reject decisions for multiple pairs, while guaranteeing that we never accept a merge unless the resulting configuration of global and local parameters would yield an improved objective function value. The only caveat is that, like DP mixtures or HDP topic models, we can only accept merge pairs i, j where both cluster indices have not been involved in any previous accepted merges during that pass. Because many entries of H' are near-zero, our bound is very tight, and in practice enables us to scalably merge many redundant state pairs in each lap through the data.

Selecting candidate merge pairs.

Before a single pass through the dataset, we examine all pairs of states and keep those pairs i, j which improve a score function similar to that used for DP mixture models in Sec. 4.1.4. That is, we use all terms of the objective $\mathcal{L}$ for which a candidate value for pair i, j can be constructed directly from global sufficient statistics and global parameters.

6.5.2 Birth proposals

Our birth moves for the HDP-HMM can create many new states at once while maintaining the monotonic increase of the whole-dataset objective, $\mathcal{L}$. Each proposal happens within the local step by trying to improve $q(z_d)$ for a single sequence d. Given current assignments $\hat{s}_d$ with truncation K, the move proposes new assignments $\hat{s}'_d$ that include the K existing states and some new states with index $k > K$. This procedure is detailed below. Next, we combine the proposed value of $\hat{s}'_d$ with the existing summary statistics for other sequences to obtain valid candidate summary statistics M'^G, N'^G, S'^G for the whole dataset. Given these global summaries, we can compute candidate global parameters $\hat{u}', \hat{\omega}', \hat{\theta}', \hat{\tau}', \hat{\nu}'$ via the standard global coordinate ascent step updates. Finally, we can compute $\mathcal{L}'$ for the candidate value, and compare it to the original configuration's objective function score $\mathcal{L}$. If $\mathcal{L}$ improves under the proposal, we accept and use the expanded set of states for all remaining updates in the current lap.

Constructing proposed assignments $\hat{s}$

The proposal for expanding the pair-wise assignments $\hat{s}'_d$ for sequence d with new states can flexibly take any form, from very naïve to very data-driven. For data with "sticky" state persistence, we recommend randomly choosing one interval $[t, t+\delta]$ of the current sequence to reassign when creating $\hat{s}'_d$, leaving other timesteps fixed. We split this interval into two contiguous blocks (one may be empty), each completely assigned to a new state. This search finds the cut point that maximizes the observation-model objective $\mathcal{L}_{\text{data}}$. Other proposals such as sub-cluster splits Chang and Fisher III (2014) could be easily incorporated in our variational algorithm, but we find this simple interval-based proposal to be fast and effective.

6.5.3 Delete Proposals

Our proposal to delete a rarely-used state j begins by dropping row j and column j from M to create M', and dropping S_j from S to create S'. Using a *target dataset* of sequences with non-trivial mass on state j, $x' = \{x_n : \sum_{t=1}^{T_n} \hat{r}_{ntj} > 0.01\}$, we run global and local parameter updates to reassign observations from former state j in a data-driven way. Rather than verifying on only the target dataset as in Hughes et al. (2015a), we accept or reject the delete proposal via the whole-dataset bound $\mathcal{L}$. To control computation, we only propose deleting states used in 10 or fewer sequences.

Candidate selection. Deletes are more flexible than merges at reassigning mass but require expensive local steps on the target data. To keep costs affordable, before each lap we find a group of L states whose unified target set is at most 10 sequences, prioritizing states that have not been attempted before. We then collect the unified target set and try all L deletes one at a time. If no group can be found ($L = 0$), no move is performed.

6.6 Experimental Results

In this section, we present results from several experiments comparing various training algorithms for the HDP-HMM. We consider two variants of our scalable memoized algorithm with adaptive proposals: one with all three proposal moves (birth, merge, and delete) and one with only proposal moves that remove clusters (merge, delete).

As a baseline, we consider two scalable algorithms that also optimize the same objective $\mathcal{L}$: a fixed-truncation memoized algorithm and a fixed-truncation stochastic algorithm. We further compare to the blocked Gibbs sampler for the HDP-HMM (Fox et al., 2011) that was previously shown to mix faster than slice samplers (Van Gael et al., 2008). These baselines (including the blocked sampler) maintain a fixed number of states K, though some states may have usage fall to zero as training proceeds. We start all fixed-K methods (including the sampler) from matched initializations, so comparisons are fair. See the original paper (Hughes et al., 2015b) and its supplement for futher discussion and all details needed to reproduce these experiments. Our publically released Python source code can be found online[1].

Learning rate schedule. For all datasets, we set the learning rate ρ_t for stochastic variational at update iteration t to $\rho_t = (1+t)^{0.51}$. This is a fairly aggressive schedule, recommended in past work by Bryant and Sudderth (2012). Future work could tune this specifically for each dataset, but we chose to simplify the comparisons here. Note that under this setting, SVI reaches noticeably better objective scores than the fixed-truncation memoized algorithm on the chromatin experiments, but performs worse on the larger-scale motion capture experiments.

Initialization. Across all experiments, we used the same procedure to initialize algorithms given the provided data sequences and a specific number of clusters K. We call this procedure "random-contiguous-blocks", since it selects subwindows of data sequences at random and uses these to create the global likelihood parameters (via the standard global step).

We found this worked well with all datasets here. To generalize to a new dataset, though, it is often advantageous to define the window length used as longer or shorter than our default, since some datasets have much longer segments than others. Also, higher-dimensional datasets can benefit from using more data in initialization. Of course, if the dataset has very little self-transition (lots of fast-switching states), another type of initialization may be preferred.

6.6.1 Toy Data

In Fig. 6.2, we study 32 toy data sequences generated from 8 Gaussian states with sticky transitions. This simple toy dataset was suggested by previous work on scalable hidden Markov models (Foti et al., 2014) as a simple check of algorithm quality. From an abundant initialization with 50

[1] http://www.bitbucket.org/michaelchughes/bnpy-dev/

states, the sampler and non-adaptive variational methods require hundreds of laps to remove redundant states, especially under a non-sticky model ($\kappa = 0$). In contrast, our adaptive methods reach the ideal of zero Hamming distance within a few dozen laps regardless of stickiness, suggesting less sensitivity to hyperparameters.

This toy dataset has 32 sequences divided into $B = 8$ batches. Each sequence has length $T = 1000$. Each observation is a 2D real vector $x_{nt} \in \mathbb{R}^2$. We use a full-covariance Gaussian for likelihood L and a corresponding Wishart distribution for the prior P. When we show segmentations in Fig. 6.2, we always show segmentations for sequences 1, 3, 5, and 7, which together contain at least 50 timesteps representing each of the 8 true states.

In Fig. 6.2, we show trace plots that compare the progress of various algorithms under two settings: non-sticky dynamics ($\kappa = 0$) and sticky dynamics ($\kappa = 50$). Comparing non-sticky and sticky models, we see that the sticky model generally encourages faster convergence for all algorithms. In particular, for the Gibbs sampler of Fox et al. (2011), the non-sticky sampler takes thousands of laps through the dataset for Hamming distance to drop near zero, but it does reach that configuration, as illustrated by the segmentations in the bottom row. In contrast, the sticky sampler reaches this ideal by around 200 laps according to the trace plots. Thus, the performance of the sampler can be quite sensitive to the value of the provided sticky hyperparameter. We thank an anonymous reviewer for suggesting this detailed analysis.

Note that across Fig. 6.2 in both sticky and non-sticky cases, our adaptive algorithms with birth/merge/delete proposals eliminate the redundant states more quickly than non-adaptive competitors. Our proposal moves enable fast convergence regardless of the hyperparameter values, suggested that algorithms with greater power to escape local optima can avoid some sensitivity exhibited by more limited methods.

6.6.2 Speaker Diarization

We study 21 unrelated audio recordings of meetings with an unknown number of speakers from the NIST 2007 speaker diarization challenge NIST (2007). The sticky HDP-HMM previously achieved state-of-the-art diarization performance Fox et al. (2011) using a sampler that required hours of computation. We ran methods from 10 matched initializations with 25 states and $\kappa = 100$, computing Hamming distance on non-speech segments as in the standard DER metric. Fig. 6.3 shows that within minutes, our algorithms consistently find segmentations better aligned to true speaker labels.

There are $N = 21$ sequences, which have no overlap in terms of common speakers. Thus, we process each one independently. This makes "memoized" inference equivalent to full-dataset inference because there is only one batch. It also makes stochastic inference irrelevant, so we only compare to the sampler as a baseline method. We use a full covariance Gaussian likelihood with corresponding Wishart prior.

For Hamming distance computations on this dataset, we utilize the provided annotations of each sequence into "background" (non-speech) and "foreground" (speech) states. We only count

timesteps labeled as foreground in the distance computation, and ignore any assignments to timesteps labelled background. Our dataset clearly marks background labels with negative integer labels, while foreground states have non-negative labels $\{0, 1, ...\}$.

6.6.3 Motion capture dataset.

Labelled 6-sequence motion capture dataset. Next, we consider the task of unsupervised discovery of activity types from motion capture sensor data captured over time. Fox et al. (2014) presented a benchmark dataset with 6 total sequences, each labelled at every timestep as one of 12 exercise types, such as jogging and jumping-jacks. The raw sensor data as well as the labels provided by a human annotator are illustrated in Fig. 1.4. Each sequence has 12 joint angles (wrist, knee, etc.) captured at 0.1 second intervals. To model this data, we use a first order auto-regressive (AR) Gaussian likelihood with the corresponding matrix-Normal-Wishart conjugate prior. For online algorithms, we process each of the 6 sequences as its own batch.

Fig. 6.4 shows that non-adaptive methods struggle even when initialized abundantly with 30 (dashed lines) or 60 (solid) states, while our adaptive methods reach better values of the objective $\mathcal{L}$ and cleaner many-to-one alignment to true exercises. It seems the 30 state models are slightly preferred (especially for the sampler). However, for our adaptive models with deletes and merges (red curves) and with births (purple), the number of states in the initialization does not seem to matter too much.

Large 124-sequence motion capture dataset. Next, we apply scalable methods to the 124 sequence dataset of Fox et al. (2014). Again, we use first-order auto-regressive (AR) likelihoods with the corresponding conjugate priors. For this larger dataset, we process all 124-sequences as 20 distinct batches, each containing about 6 sequences.

Fig. 6.5 shows deletes and merges making consistent reductions from abundant initializations and births growing from $K = 1$. Fig. 6.5 also shows estimated segmentations for 10 representative sequences, along with skeleton illustrations for the 10 most-used states in this subset. These segmentations align well with held-out text descriptions of the raw sensor sequences, which are found on the source website. We lack ground truth exercise labels at each timestep, but these whole-sequence labels provide a proxy for judging the resulting segmentation. Scripts for visualizing the skeleton trace of a specific data segment can be found in an open-source code repository available online[2].

For non-adaptive methods on the 124-sequence dataset, we compare each algorithm initialized from abundant initializations of 100 (dashed) and 200 (solid) states. For the stochastic method (SVI, yellow curves), under all initializations we see a rapid drop in the number of states used during the first lap. To explain this, remember that with 20 batches for 124 sequences, each batch will have around 6 sequences. From the segmentation figure, it is clear that state usage patterns have lots of variety across sequences, which each sequence only using a handful of states. The aggressive learning rate we use in the first lap will tend to severely downweight any initial states not used in the first few

[2] http://github.com/michaelchughes/mocap6dataset/

batches, which explains the rapid drop. In contrast, the memoized method (blue) is designed to use global information for each parameter update, not just the current batch. We further enforce this by delaying the first global update until at least 50 sequences are seen. This makes the memoized results a large improvement on the stochastic results for this dataset.

Furthermore, using delete and merge moves only (red) shows that we can reduce down to about 30 states and reach even higher levels of performance. Similarly, starting from 1 state with birth moves (purple), we can grow to nearly comparable levels of performance. We hope to answer why the purple curves do not quite reach the performance of the red curves in future work. Regardless, the set of adaptive methods reach high objective scores much more consistently than non-adaptive methods.

6.6.4 Chromatin epigenomic dataset

Finally, we study segmenting the human genome by the appearance patterns of regulatory proteins Hoffman et al. (2012). This problem of chromatin segmentation was motivated earlier in Ch. 1 and illustrated in Fig. 1.5.

We observe 41 binary signals from Ernst and Kellis (2010) at 200bp intervals throughout a white blood cell line (CD4T). Each binary value indicates the presence or absence of an acetylation or methylation that controls gene expression. Such binary observations are naturally explained by a Bernoulli likelihood with a conjugate Beta prior. We set our prior to $\text{Beta}(0.1, 0.3)$, which favors clusters with extreme probabilities (nearly 0 or nearly 1).

We divide the whole epigenome into 173 sequences (one per batch) with total size $T = 15.4$ million. Some efforts choose to process each chromosome as one very long sequence, but we elected to test the ability of our algorithms to handle many batches. To split up each chromosome, we searched for intervals with at least 50 consecutive all-zero observations, which are somewhat common. We picked division points in the middle of these empty segments to use to split up into more manageable size sequences, while avoiding artifacts at the starts of each sequence as much as possible. In the end, we obtained 173 sequences, ranging in size from $T = 10000$ to $T = 200,000$ timesteps.

Fig. 6.6 shows our adaptive proposal optimization algorithms can grow from 1 state to 70 states and compete favorably with non-adaptive competitors even on genome-scale datasets. Even when fixed truncation algorithms are given 100 initial clusters, they do not achieve ELBO scores as high as the nearly 75 clusters discovered by our method.

Fig. 6.6 also demonstrates that parallel processing of distinct sequences in the local step can produce useful speed improvements. Using 64 cores working in parallel on a $K = 50$ cluster problem, we can complete the fixed-truncation local step 25x times faster than a serial implementation. This lowers the time required to complete a local step on the entire dataset (173 sequences totalling 15 million observations) from over an hour to less than 2 minutes. This practical speed gain makes large-scale analysis with our HDP-HMM optimization algorithms possible.

6.7 Discussion

Extension of our previous methods to the HDP hidden Markov model required further technical innovation to derive a surrogate optimization problem for the sticky HDP as well as several innovations to make the proposal moves (birth, merge, delete) apply to sequential data.

Among open issues, we note that the primary obstacle to large-scale scalability of these methods is the $O(K^2 T_d)$ of the local step at each sequence d. This quadratic dependence on the number of active clusters K makes training with many clusters very slow even with the parallelization we have developed. Finding slightly different models or inference assumptions which would improve this scaling to nearly-linear in K could make it possible to train effectively on millions of sequences and thousands of hidden states or clusters.

Algorithm 6.2 Memoized variational algorithm for the HDP-HMM

Input:

$\{x_d\}_{d=1}^D$: dataset with D sequences, each of length T_d
K: truncation level
$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: initial global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: initial global parameters of allocation model defining $q(u)$
$\{\{\hat{\theta}_{jk}\}_{k=1}^{K+1}\}_{j=0}^K$: initial global parameters defining Markov transition probabilities
γ, α, α_0: allocation model hyperparameters
$\bar{\tau}, \bar{\nu}$: observation model prior hyperparameters

Output:

$\{\hat{\tau}_k, \hat{\nu}_k\}_{k=1}^K$: updated global parameters of observation model
$\{\hat{u}_k, \hat{\omega}_k\}_{k=1}^K$: updated global parameters of allocation model
$\{\{\hat{\theta}_{jk}\}_{k=1}^{K+1}\}_{j=0}^K$: updated global parameters defining Markov transition probabilities.

function MemoizedCoordAscentForHDPHMM($x, K, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}, \hat{\theta}$)
  for lap $l \in 1, 2, \ldots$ **do**
    for batch $b \in$ Shuffle($\{1, 2, \ldots B\}$) **do** ▷ Local step at batch b
      for $d \in \mathcal{D}_b$ **do**
        $\hat{s}_d, \hat{r}_d \leftarrow$ FwdBwdAlgForSeq($C_d(\hat{\tau}, \hat{\nu}), P(\hat{\theta})$)

      $S^G \leftarrow S^G - S_b$ ▷ Decrement global statistics
      $N^G \leftarrow N^G - N_b$
      $M^G \leftarrow M^G - M_b$

      for $k \in 1, 2, \ldots K$ **do** ▷ Summary step at batch b
        $S_{bk} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk} s(x_{dt})$
        $N_{bk} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{dtk}$
        $M_{b0k} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{r}_{d1k}$
        for $j \in 1, 2, \ldots K$ **do**
          $M_{bjk} \leftarrow \sum_{d=1}^D \sum_{t=1}^{T_d} \hat{s}_{dtjk}$

      $S^G \leftarrow S^G + S_b$ ▷ Increment global statistics
      $N^G \leftarrow N^G + N_b$
      $M^G \leftarrow M^G + M_b$

      for $k \in 1, 2, \ldots K$ **do** ▷ Global step for observation parameters
        $\hat{\tau}_k \leftarrow S_k + \bar{\tau}$
        $\hat{\nu}_k \leftarrow N_k + \bar{\nu}$

      while not converged **do**
        for $j \in 0, 1, 2, \ldots K$ **do**
          for $k \in 1, 2, \ldots K+1$ **do** ▷ Update global transition probabilities $q(\pi_j)$
            $\hat{\theta}_{jk} \leftarrow M_{jk} + \alpha_j \pi_k^G(\hat{u}) + \delta_k \kappa$
          for $k \in 1, 2, \ldots K+1$ **do**
            $P_{jk} \leftarrow \psi(\hat{\theta}_{jk}) - \psi(\sum_{\ell=1}^{K+1} \hat{\theta}_{j\ell})$
        $\hat{u} \leftarrow \arg\max_{\hat{u}} f(\hat{u}, \hat{\omega}, P(\hat{\theta}), \alpha, \gamma, \kappa)$ ▷ Update global top probabilities $q(\pi^G)$
  return $\hat{\tau}, \hat{\nu},\ \hat{u}, \hat{\omega},\ \{\hat{\theta}\}_{j=0}^K$

Memoized coordinate ascent algorithm for approximate posterior inference for the HDP-HMM.

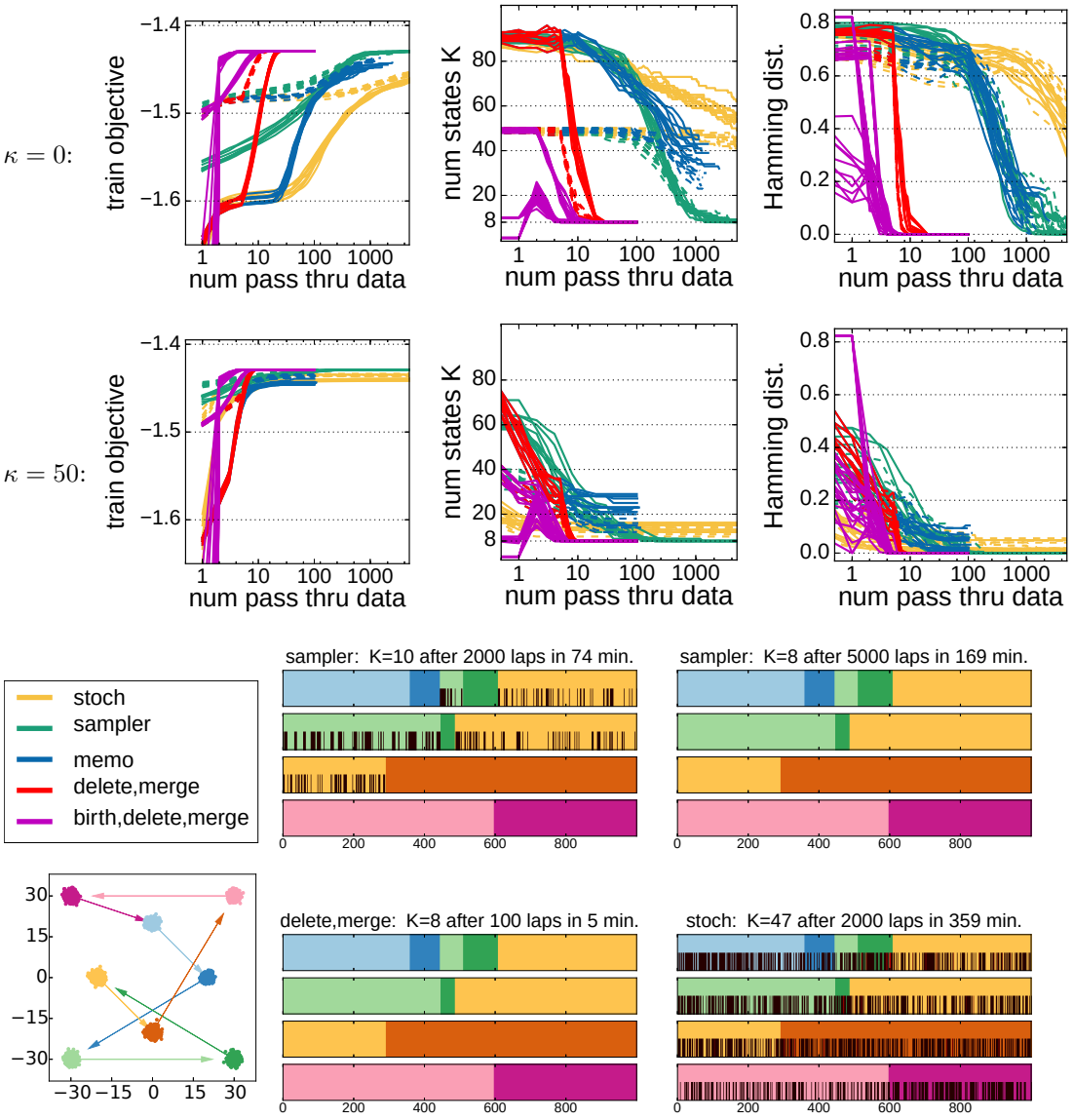

Figure 6.2: Toy data HDP-HMM algorithm comparison, using sticky and non-sticky model. Algorithm comparison on toy dataset, using a non-sticky state transition model with $\kappa = 0$ (*top row*) and sticky model with $\kappa = 50$ (*middle row*). *Left column:* Objective function $\mathcal{L}$ as more training data is seen. *Middle column:* Number of effective states K (states with $N_k \geq 1$) as more data is seen. *Right column:* Hamming distance between aligned segmentations and the ground truth segmentation. All non-birth algorithms are initialized using a common set of over-complete segmentations, with either $K = 50$ (dashed lines) or $K = 100$ states (solid lines). The sticky model encourages faster convergence for all algorithms. The non-sticky sampler takes thousands of laps through the dataset for Hamming distance to drop near zero, but it does reach that configuration, as illustrated in the segmentations in the bottom rows. In contrast, the sticky sampler reaches this ideal by around 200 laps according to the middle trace plots. Both sticky and non-sticky models clearly prefer the ideal segmentation, but algorithm convergence is sensitive to the sticky hyperparameter, especially the fixed-truncation variational algorithms. In contrast, our adaptive methods reach ideal configurations within 20 laps regardless of whether the model is sticky or not.

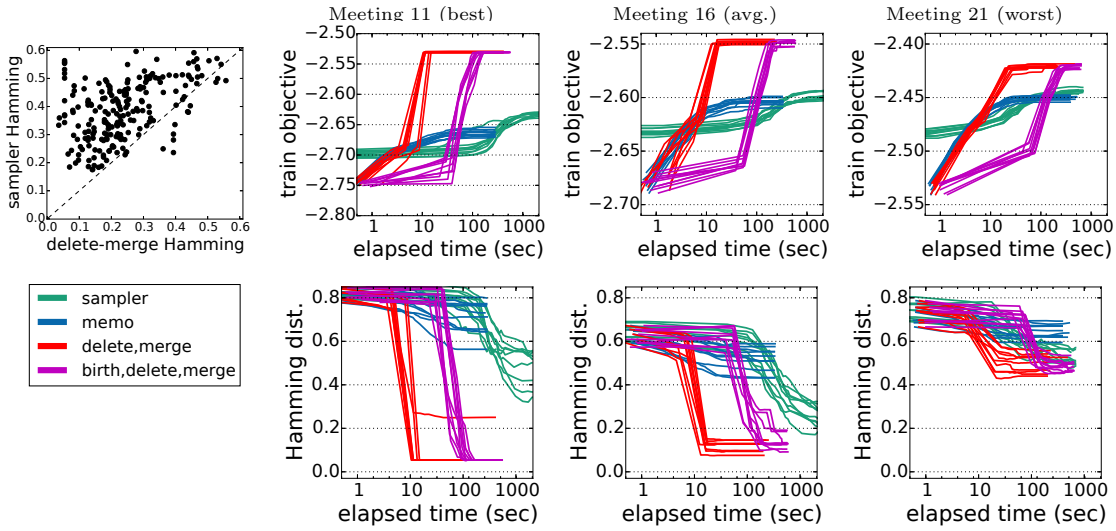

Figure 6.3: Comparison of HDP-HMM algorithms on 21 speaker diarization sequences. Method comparison on speaker diarization from common $K = 25$ initializations (Sec. 6.6.2). *Left:* Scatterplot of final Hamming distance for our adaptive method and the sampler. Across 21 meetings (each with 10 initializations shown as individual dots) our method finds segmentations closer to ground truth. *Right:* Traces of objective $\mathcal{L}$ and Hamming distance for meetings representative of good, average, and poor performance.

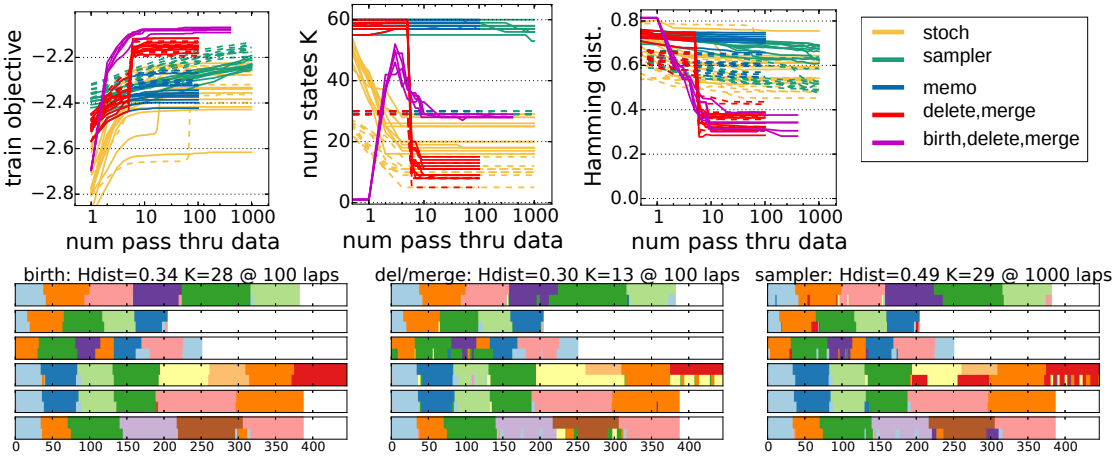

Figure 6.4: Comparison of HDP-HMM algorithms on 6 motion capture sequences. Comparison on 6 motion capture streams (Sec. 6.6.3). *Top:* Our adaptive methods reach better $\mathcal{L}$ values and lower distance from true exercise labels. *Bottom:* Segmentations from the best runs of birth/merge/delete (left), only deletes and merges from 30 initial states (middle), and the sampler (right). Each sequence shows true labels (top half) and estimates (bottom half) colored by the true state with highest overlap (many-to-one).

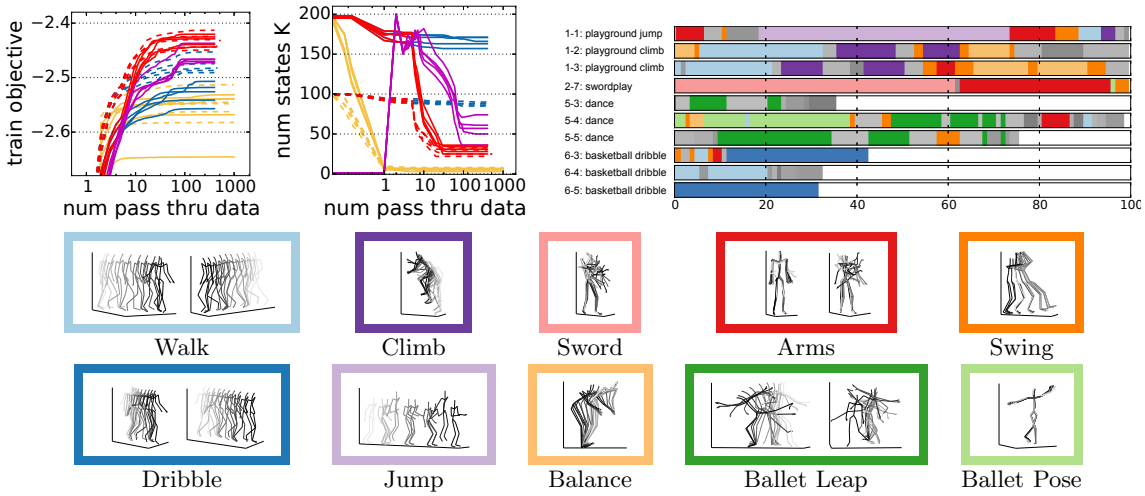

Figure 6.5: Comparison of HDP-HMM algorithms on 124 motion capture sequences. Study of 124 motion capture sequences (Sec. 6.6.3). *Top Left:* Objective $\mathcal{L}$ and state count K as more data is seen. Solid lines have 200 initial states; dashed 100. *Top Right:* Final segmentation of 10 select sequences by our method, with id numbers and descriptions from `mocap.cs.cmu.edu`. The 10 most used states are shown in color, the rest with gray. *Bottom:* Time-lapse skeletons assigned to each highlighted state.

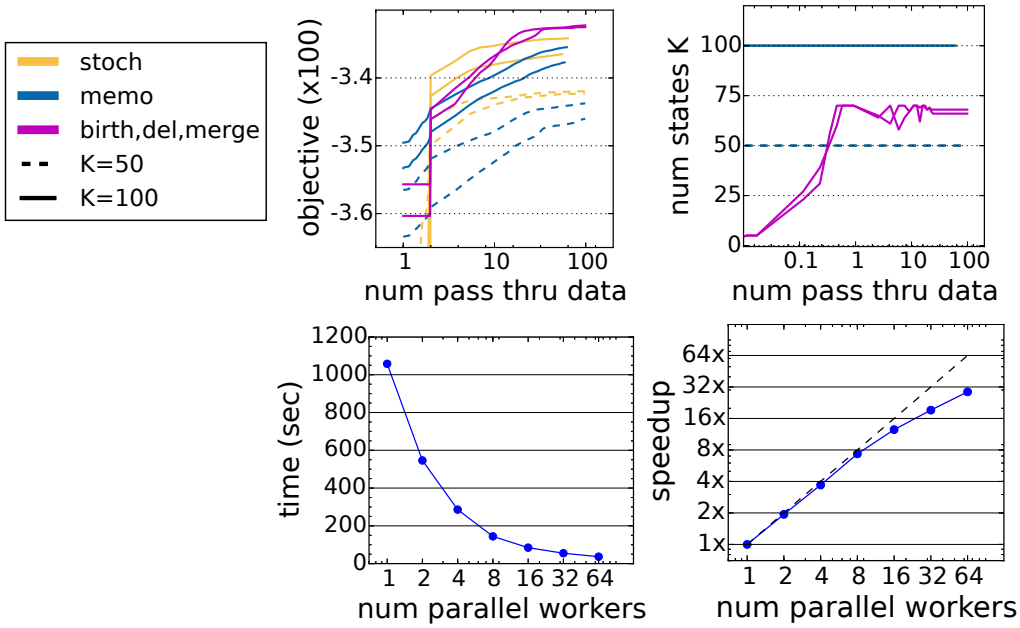

Figure 6.6: Comparison of HDP-HMM algorithms for chromatin segmentation of human genome. Segmentation of human epigenome: 15 million observations across 173 sequences (Sec. 6.6.4). *Top Row:* Adaptive runs started at 1 state grow to 70 states within one lap and reach better $\mathcal{L}$ scores than 100-state non-adaptive methods. Each run takes several days. *Bottom Row:* Wallclock times and speedup factors for a parallelized fixed-truncation local step on 1/3 of this dataset. 64 workers complete a local step with $K = 50$ states in under one minute.

Chapter 7

Sparse variational posteriors for cluster and topic assignments

In this chapter, our overall goal is to improve scalability of training algorithms by reducing the runtime cost of the most expensive step in our earlier coordinate ascent optimization algorithms: the local step. Just as our earlier memoized algorithms helped us scale to large dataset sizes N, we now wish to improve scaling with larger numbers of clusters or topics K.

Towards this end, we investigate alternative posterior approximations for the local assignment variables z in mixture models and topic models. Our previous efforts for these models in Ch. 3 and Ch. 5 used a conventional approximate posterior $q(z)$ which explicitly represented the probability of each data token being assigned to each of the K active clusters. Here, we propose a *constrained* family of sparse variational distributions that allow at most $L < K$ non-zero entries in the learned resposibility parameter vector for each local assignment, where the user-specified threshold L trades off speed for accuracy. Setting $L = K$ recovers the dense representation of our earlier conventional approaches, while the other extreme of setting $L = 1$ yields a "hard" or "winner-take-all" assignments. We will show that moderate values such as $L \approx 4$ can produce significant speed gains compared to dense $L = K$ conventional representations when $K >> 100$, while still producing similar-quality heldout predictions. Our approach fits into any variational algorithm for parametric or nonparametric models regardless of whether global parameters are inferred by point estimates (as in EM) or given full approximate posteriors. Furthermore, our approach easily integrates into existing frameworks for large-scale streaming data analysis (Hoffman et al., 2013; Broderick et al., 2013), and is easy to parallelize.

At the time of publication, the work in this chapter is under review for a machine learning conference. To summarize, the core contributions are:

Contribution 1: New variational algorithm for mixture models with user-specified L-sparse assignments. Our approach is the first to use $O(L)$ memory to represent the assignment

posterior $q(z_n)$ for data token n. Hard assignment algorithms with $L = 1$ have long been known in the literature, but no procedure to coherently optimize an L-sparse $q(z_n)$ for $1 < L < N$ was previously known to our knowledge. Algorithms like sparse EM (Neal and Hinton, 1998) allow the update to $q(z_n)$ to process fewer than K indices of the corresponding responsibility vector during an update, but must represent all K entries in memory. In contrast, our approach requires less storage and less processing time, especially for computing summary statistics.

Contribution 2: New variational algorithm for topic models with user-specified L-sparse assignments. We present a similar algorithm for topic models, for which hard assignments via MCMC sampling had been the most prominent previous way to attain $L = 1$ sparsity in $q(z_n)$ (Mimno et al., 2012).

Contribution 3: Experimental evidence that moderate L values offer good speed-accuracy tradeoff. We often find it favorable to choose *moderate* values of L larger than the winner-take-all baseline but much smaller than the total number of active clusters K. Our later experiments show that the hard assignment condition $L = 1$ often leads to pathological behavior, especially in topic models.

7.1 Local step algorithms for L-sparse mixture models

In this section, we focus our attention on the local step of the coordinate ascent algorithm for the finite mixture model presented in Ch. 2. We begin by reviewing the conventional local step update that produces dense responsibility vectors, which was first presented in Sec. 2.6.1. Later, we introduce our new sparse variational approximate posterior $q(z_n)$, which adds an additional constraint to the conventional dense formulation and derive the appropriate coordinate ascent update for this new approximation. We emphasize that while we've chosen to present the material here for the finite mixture model for simplicity, the parallels to the local step for DP mixture models in Ch. 3 are straight-forward.

7.1.1 Local step with conventional dense responsibilities.

The local step of inference computes a new assignment vector $\hat{r}_n$ for each observation n which optimizes $\mathcal{L}$ while holding global parameters fixed. Under either the approximate posterior treatment of global parameters in Eq. (2.93) or ML estimates of Eq. (2.73), the optimal updates have closed form, which we derive by expanding the expectations that define $\mathcal{L}$ and dropping terms independent of $\hat{r}_n$. After simplification, we have

$$\mathcal{L}_n(x_n, \hat{r}_n) = \sum_{k=1}^{K} \hat{r}_{nk} W_{nk}(x_n) - \hat{r}_{nk} \log \hat{r}_{nk}, \tag{7.1}$$

$$W_{nk}(x_n, \hat{\theta}, \hat{\tau}, \hat{\nu}) \triangleq \mathbb{E}_q[\log \pi_k] + \mathbb{E}_q[\log p(x_n | \phi_k)].$$

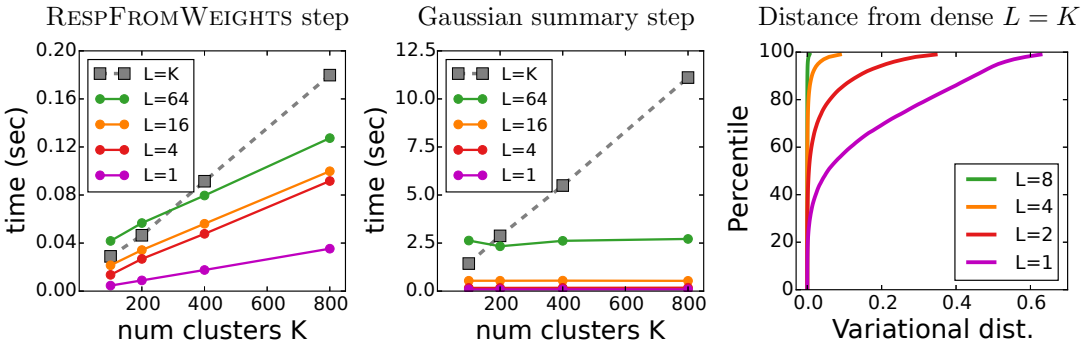

Figure 7.1: Speed and accuracy of L-sparse assignment posteriors for training mixture models Impact of sparsity-level L on different substeps of estimating local assignments for a mixture model. L defines the number of non-zero entries in the posterior responsibility vector $\hat{r}_n$ for each observation n. For these experiments, we use a minibatch of $N = 36{,}000$ 8x8 image patches and a pretrained mixture model with zero-mean Gaussian likelihood, as described in Sec. 7.2.1. *Left:* Comparison of methods from Alg. 2.3 for computing optimal responsibilities $\hat{r}_n$ given fixed log posterior weights W_{nk} for each cluster. When $L = K$ we use DenseRespFromWeights while for $L < K$ we find TopLRespFromWeights is much faster. *Center:* Given fixed assignments $\hat{r}$, the summary step computes the per-cluster sufficient statistics $\{N_k, S_k\}_{k=1}^K$ defined in Eq. (7.3). Here, we measure the time required for this computation for our minibatch of 8x8 patch data, where the observation statistic $s(x_n) = x_n x_n^T$ is 64x64. *Right:* Comparison of variational distance between optimal dense responsibilities and those from TopLRespFromWeights across various L values. We show the cumulative density function across all patches in our minibatch, using the pretrained $K = 200$ model published online by Zoran and Weiss (2012). Moderate values $L \approx 8$ are almost indistinguishable from the dense solution.

We interpret $W_{nk} \in \mathbb{R}$ as the log posterior weight that cluster k has for observation n. Larger values imply that cluster k is more likely to be assigned to observation n. For ML or MAP learning, the expectations defining W_{nk} are replaced with point estimates.

Our goal is to find the cluster responsibility vector $\hat{r}_n$ that optimizes $\mathcal{L}_n$ in Eq. (7.1), subject to the constraint that the entries of $\hat{r}_n$ are non-negative and sum to one:

$$\hat{r}_n^* = \arg\max_{\hat{r}_n} \mathcal{L}_n(x_n, \hat{r}_n) \text{ s.t. } \hat{r}_n \geq 0, \textstyle\sum_k \hat{r}_{nk} = 1. \tag{7.2}$$

The optimal solution is simple: exponentiate each weight and then normalize the resulting vector. The function DenseRespFromWeights in Alg. 7.1 details the required steps. The runtime cost is $O(K)$, dominated by K required evaluations of the exp function.

Summary statistics. Given fixed assignments $\hat{r}$, the global step computes the optimal values of the global free parameters under $\mathcal{L}$. Whether doing point estimation or approximate posterior inference, this update requires only two finite-dimensional sufficient statistics of $\hat{r}$, rather than the complete $\hat{r}$ matrix. For each cluster k, we must compute the expected count $N_k \in \mathbb{R}^+$ of its assigned observations and the expected data statistic vector S_k. These summary statistics were first introduced in Eq. (2.96), and are defined again here for convenience:

$$N_k(\hat{r}) = \textstyle\sum_{n=1}^N \hat{r}_{nk}, \quad S_k(x, \hat{r}) = \sum_{n=1}^N \hat{r}_{nk} s(x_n). \tag{7.3}$$

Algorithm 7.1 Update for dense responsibilities given log posterior weights for mixture model.

Input: $[W_{n1} \dots W_{nK}]$: log posterior weights.
Output: $[\hat{r}_{n1} \dots \hat{r}_{nK}]$: resp. values for each cluster

function DENSERESPFROMWEIGHTS(W_n)
  for $k \in 1, \dots K$ **do**
    $\hat{r}_{nk} = e^{W_{nk}}$
  $s_n = \sum_{k=1}^{K} \hat{r}_{nk}$
  for $k \in 1, \dots K$ **do**
    $\hat{r}_{nk} = \hat{r}_{nk} / s_n$
  return $\hat{r}_n$

DENSERESPFROMWEIGHTS is the conventional method for solving the optimization problem in Eq. (7.2). It delivers a dense vector $\hat{r}_n$ of K non-zero entries, satisfying non-negativity and sum-to-one constraints. The runtime cost requires K evaluations of the exp function, K summations, and K divisions. For large K, can be expensive, especially because the resulting vector $\hat{r}_n$ often has a vast majority of entries indistiguishable from zero.

Algorithm 7.2 Update for L-sparse responsibilities given log posterior weights for mixture model.

Input: $[W_{n1} \dots W_{nK}]$: log posterior weights.
Output: $\{\hat{r}_{n\ell}, i_{n\ell}\}_{\ell=1}^{L}$: top L values and indices

function TOPLRESPFROMWEIGHTS(W_n, L)
  $i_{n1}, \dots i_{nL} =$ SELECTTOPL(W_n)
  for $\ell \in 1, \dots L$ **do**
    $\hat{r}_{n\ell} = e^{W_{n i_{n\ell}}}$
  $s_n = \sum_{\ell=1}^{L} \hat{r}_{n\ell}$
  for $\ell \in 1, \dots L$ **do**
    $\hat{r}_{n\ell} = \hat{r}_{n\ell} / s_n$
  return $\hat{r}_n, i_n$

TOPLRESPFROMWEIGHTS optimizes the same objective as RESPFROMWEIGHTS, subject to the additional constraint that at most L clusters can have non-zero posterior probability in $\hat{r}_n$. The L-sparse optimization problem this procedure solves is in Eq. (7.4). To tractably solve this problem, an $O(K)$ introspective selection algorithm (Musser, 1997) first identifies the indices of the L largest values of the weight vector. After this, we can find the optimum with L evaluations of the exp function, L summations, and L divisions. This cost can be substantially less than the cost of dense procedure RESPFROMWEIGHTS, as shown in Fig. 7.1.

These sums have cost linear in the number of observations N. The required work is $O(NK)$ for the count vector and $O(NKD)$ for the data vector, where D is the dimension of the data statistic vector: $s(x_n) \in \mathbb{R}^D$.

7.1.2 Local step with L-sparse responsibilities.

To speed up the local steps above, we recognize that much of the runtime cost comes from representing $\hat{r}_n$ as a dense vector. Although there are K clusters, for any observation n only a few entries in $\hat{r}_n$ will have appreciable mass while the vast majority are close to zero. We thus introduce an additional constraint to our optimization problem: that at most L entries are non-zero, where

$1 \leq L \leq K$. The formal problem is:

$$\hat{r}_n^* = \text{argmax}_{\hat{r}_n} \mathcal{L}_n(x_n, \hat{r}_n) \quad (7.4)$$
$$\text{s.t. } \hat{r}_n \geq 0, \ \sum_{k=1}^K \hat{r}_{nk} = 1, \ \sum_{k=1}^K 1(\hat{r}_{nk} > 0) = L.$$

This constrained optimization problem has a simple solution, given by the function TopLRespFromWeights in Alg. 7.1. First, we identify the indices of the top L values of the weight vector W_n in descending order. Let $i_{n1}, \ldots, i_{nL}$ denote these top-ranked cluster indices, each one a distinct value in $\{1, 2, \ldots, K\}$. Given this active set of clusters, we simply exponentiate and normalize only at these indices.

$$\hat{r}_{nk}^* = \begin{cases} e^{W_{nk}} / \sum_{\ell=1}^L e^{W_{ni_{n\ell}}} & \text{if } k \in i_{n1}, \ldots, i_{nL} \\ 0 & \text{otherwise.} \end{cases} \quad (7.5)$$

As shown in Alg. 7.1, we can represent this solution as an L-sparse vector, with L real values $\hat{r}_{n1}, \ldots, \hat{r}_{nL}$ and L integer indices $i_{n1}, \ldots, i_{nL}$. Solutions may not be unique if the posterior weights W_n contain duplicate values. We handle these ties arbitrarily, since swapping duplicate indices leaves the objective unchanged.

Proof of optimality. We offer a proof by contradiction that TopLRespFromWeights solves the optimization problem in Eq. (7.4). Suppose that $\hat{r}'_n$ is optimal, but there exists a pair of clusters j, k such j has larger weight but is not included in the active set while k is. This means $W_{nj} > W_{nk}$, but $\hat{r}'_{nj} = 0$ and $\hat{r}'_{nk} > 0$. Consider the alternative $\hat{r}_n^*$ which is equal to vector $\hat{r}'_n$ but with entries j and k swapped. After substituting into Eq. (7.1) and simplifying, we find the objective function value increases under our alternative: $\mathcal{L}(x_n, \hat{r}_n^*) - \mathcal{L}(x_n, \hat{r}'_n) = \hat{r}'_{nk} \cdot (W_{nj} - W_{nk}) > 0$. Thus, the optimal solution must include the largest L clusters by weight in its active set.

Runtime cost. ComparingDenseRespFromWeights and TopLRespFromWeights side-by-side yields pertinent insights to the runtime cost. The former requires K exponentiations, K additions, and K divisions to turn weights into responsibilities. In contrast, given the active set of cluster indices i_n our procedure requires only L of each operation. Furthermore, finding the active indices i_n can be done in $O(K)$ via selection algorithms. Thus, for $L \ll K$ we find TopLRespFromWeights to be much faster, as shown empirically in Fig. 7.1.

Selection algorithms (Blum et al., 1973; Musser, 1997) are designed to find the top L values in descending order within an array of size K. These methods use divide-and-conquer strategies to recursively partition the input array into two blocks, one with values above a pivot and the other below. Musser (1997) introduced a selection procedure which uses *introspection* to smartly choose pivot values and thus guarantee $O(K)$ worst-case runtime. This procedure is implemented within the C++ standard library as `nth_element`, which we use for SelectTopL in practice. This function operates in-place on the provided array, rearranging its values so that the first L entries are all bigger than the remainder. Importantly, there is no internal sorting within either partition. We use `nth_element` to find the top indices, not values, by defining a custom comparator.

Advantages of $L-$sparse responsibilities. The sparsity parameter L provides a practioner with a natural way to tradeoff between execution speed and training accuracy. When $L = K$, we recover the original problem in Eq. (7.2), while $L = 1$ leads to assigning each observation to exactly one cluster, as in k-means or *maximization expectation* (ME, Kurihara and Welling (2009)) algorithms. We call this $L = 1$ case "hard" assignment or "winner-take-all" assignment. Our focus is on modest values of $1 < L \ll K$. As shown in Fig. 7.1, when the number of clusters K measures in the hundreds or larger, with moderate L values we find that TopLRespFromWeights is significantly faster than DenseRespFromWeights. We find that the dense method's required K exponentiations dominates the cost of the $O(K)$ introspective selection procedure.

With $L-$sparse responsibilities, computing the summary statistics in Eq. (7.3) scales linearly with L rather than K. This gain is noticeable for Gaussian mixture models with unknown covariances, which require a $D \times D$ sufficient statistic $s(x_n) = x_n x_n^T$. Fig. 7.1 shows that the cost of a typical minibatch summary step for image patch modeling with a published model with $K = 200$ total clusters drops from over 10 seconds for the dense $L = K$ procedure to less than a second for $L = 4$.

In addition to the speed gains in Fig. 7.1, we also expect that sparsity may improve the overall convergence rate of training across many global and local steps. Finally, we hope that $L-$sparse responsibilities are more interpretable due to the lack of small-but-non-zero values.

7.1.3 Related work.

Using "hard" cluster assignments instead of "soft" probabilities is well-known way to balance accuracy for speed. K-means and its Bayesian nonparametric extension DP-means (Kulis and Jordan, 2012) justify $L = 1$ sparsity via small-variance asymptotics. Viterbi training (Juang and Rabiner, 1990), and maximization-expectation algorithms (Kurihara and Welling, 2009) both use $L = 1$ hard assignments. However, we expect $L = 1$ to be too coarse for many applications.

Few published methods allow any tuning of the sparsity-level L. Neal and Hinton (1998) introduced sparse EM, a method intended for datasets where all local parameters fit into main memory. The algorithm maintains a dense parameter vector $\hat{r}_n$ for each observation n, but only edits a subset of this vector during each local step. The edited subset may consist of the L largest entries or all entries above some threshold. Any inactive entries are "frozen" to small but non-zero values, and newly edited entries are normalized such that the whole vector $\hat{r}_n$ preserves its sum-to-one constraint. This approach is effective for small datasets (Ng and McLachlan, 2004), but the large memory requirement limits scalability. In contrast, our approach stores only L values at each observation.

More recently, several efforts have used MCMC samplers to approximate the local step within a larger variational algorithm (Mimno et al., 2012; Wang and Blei, 2012a). They estimate an approximate assignment posterior by averaging over many samples, where each sample is an $L = 1$ hard assignment. The number of finite samples S is a crucial choice which balances accuracy and speed. We think selecting an L value via our approach is more intuitive than choosing an S for the same problem. Furthermore, our method provides an exact, monotonically increasing way to optimize $\mathcal{L}$ while sampling improves $\mathcal{L}$ only in expectation.

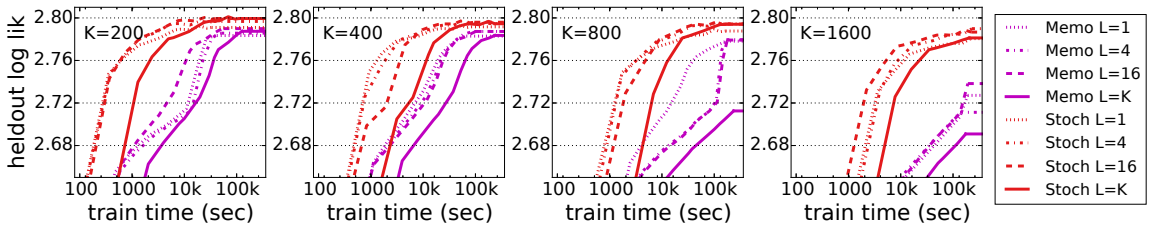

Figure 7.2: L-sparse mixture model results on millions of image patches.
Impact of sparsity on training zero-mean Gaussian mixture model on 3.6 million 8x8 pixel image patches using stochastic and memoized variational inference. Training data comes from 400 total images processed 4 images at a time. Each panel shows heldout prediction scores over time for several training runs at a fixed number of clusters K. In all cases, runs with small L values match or exceed the prediction quality of the dense baseline $L = K$ in far less time. Moderate $L = 4$ or $L = 16$ can outperform $L = 1$ hard assignments, as shown in the $K = 200$ and $K = 1600$ panel.

7.1.4 Integration with scalable and adaptive proposal algorithms

Our proposed L-sparse assignment algorithm for mixture models easily fits into the scalable variational algorithms we have discussed in detail in previous chapters. Both stochastic variational (Hoffman et al., 2013) and memoized variational algorithms (Hughes and Sudderth, 2013) from Ch. 3 can use TopLRespFromWeights as a drop-in replacement for DenseRespFromWeights without any other required changes except a summary step which takes advantage of the sparsity to gain speed.

7.2 Experimental results with L-sparse mixture models

7.2.1 Mixture models for image patches

We now consider applying mixture models to natural images, inspired by Zoran and Weiss (2012). We train a model for 8x8 image patches taken from overlapping regular grids of stride 4 pixels. Each observation is a vector $x_n \in \mathbb{R}^{64}$, preprocessed to remove its mean. We then apply a mixture model with concentration $\alpha = 10$, using a zero-mean, full-covariance Gaussian likelihood function with corresponding conjugate Wishart prior.

To evaluate, we track the log-likelihood score of heldout observations x' under our trained model:

$$\text{score}(x'_n) = \log \sum_{k=1}^{K} \hat{\pi}_k \mathcal{N}(x'_n | 0, \hat{\Sigma}_k). \tag{7.6}$$

Here, $\hat{\pi}_k = \mathbb{E}_q[\pi_k]$ and $\hat{\Sigma}_k = \mathbb{E}_q[\Sigma_k]$ are point estimates computed from our trained global parameters using standard formulas. The function $\mathcal{N}$ is the probability density function of a multivariate normal.

Fig. 7.2 compares stochastic and memoized implementations of our algorithm on 3.6 million patches from 400 images. The algorithms process 100 minibatches each with $N = 36816$ patches. First, we confirm clear speed gains. With $K = 800$ clusters, we can process all 3.6 million patches in about 8100 seconds with $L = 4$, while the dense procedure takes 27000 seconds (over 7 hours).

Second, these speed gains do not sacrifice prediction quality. Across many values of K, we see that sparse runs with $L < K$ reach similar heldout scores as the dense baselines, and often much better values as shown in the $K = 800$ or $K = 1600$ panels.

7.3 Local step algorithms for L-sparse topic models

Next, we apply our sparse approximate posterior idea to topic models. Again, for simplicity we focus on a finite topic model known as latent Dirichlet allocation (Blei et al., 2003), but extensions to our Bayesian nonparametric HDP topic model from Ch. 5 are quite straightforward.

7.3.1 Mean field for the LDA Topic Model

Our sparsity-level constraint naturally applies to topic models, which are hierarchical mixtures applied to data from D documents, $x_1, \dots x_D$ containing words from a finite vocabulary of size V. The Latent Dirichlet Allocation (LDA) topic model (Blei et al., 2003) generates a document's observations from a mixture model with common topics $\{\phi\}_{k=1}^K$ but document-specific frequencies π_d. This is the finite version of the infinite model presented in Fig. 5.1, though with a different, non-hierarchical prior for $p(\pi^G)$.

Each topic $\phi_k \sim \text{Dir}_V(\bar{\lambda})$, where ϕ_{kv} is the probability of type v under topic k. The document-specific frequences π_d are drawn from a symmetric Dirichlet $\text{Dir}_K(\frac{\alpha}{K} \dots \frac{\alpha}{K})$, where $\alpha > 0$ is a scalar. Tokens are generated:

$$z_{dn} \sim \text{Cat}_K(\pi_d), \quad x_{dn} \sim \text{Cat}_V(\phi_{z_{dn}}). \tag{7.7}$$

The goal of posterior inference is to estimate the common topics as well as the frequencies and assignments in any document. The standard mean-field approximate posterior over these quantities is specified by:

$$\begin{aligned} q(z_d) &= \textstyle\prod_{n=1}^{N_d} \text{Cat}_K(z_{dn}|\hat{r}_{dn1}, \dots \hat{r}_{dnK}), \\ q(\pi_d) &= \text{Dir}_K(\pi_d|\hat{\theta}_{d1}, \dots \hat{\theta}_{dK}), \\ q(\phi) &= \textstyle\prod_{k=1}^{K} \text{Dir}_V(\phi_k|\hat{\lambda}_{k1}, \dots \hat{\lambda}_{kV}). \end{aligned} \tag{7.8}$$

Under this factorized approximate posterior, we can again set up a variational optimization objective:

$$\begin{aligned} \mathcal{L}(x, \hat{r}, \hat{\theta}, \hat{\lambda}) &= \log p(x) - \text{KL}(q||p) \\ &= \mathbb{E}_q[\log p(x, z, \pi, \phi) - \log q(z, \pi, \phi)] \end{aligned} \tag{7.9}$$

7.3.2 Local step of LDA Training Algorithm

Here, we derive an interative update algorithm for estimating the assignment factor $q(z_d)$ and the frequencies factor $q(\pi_d)$ for a document d. Alg. 5.1 gives the conventional local step algorithm, while Alg. 7.4 gives our novel algorithm for $L-$sparse assignment posteriors.

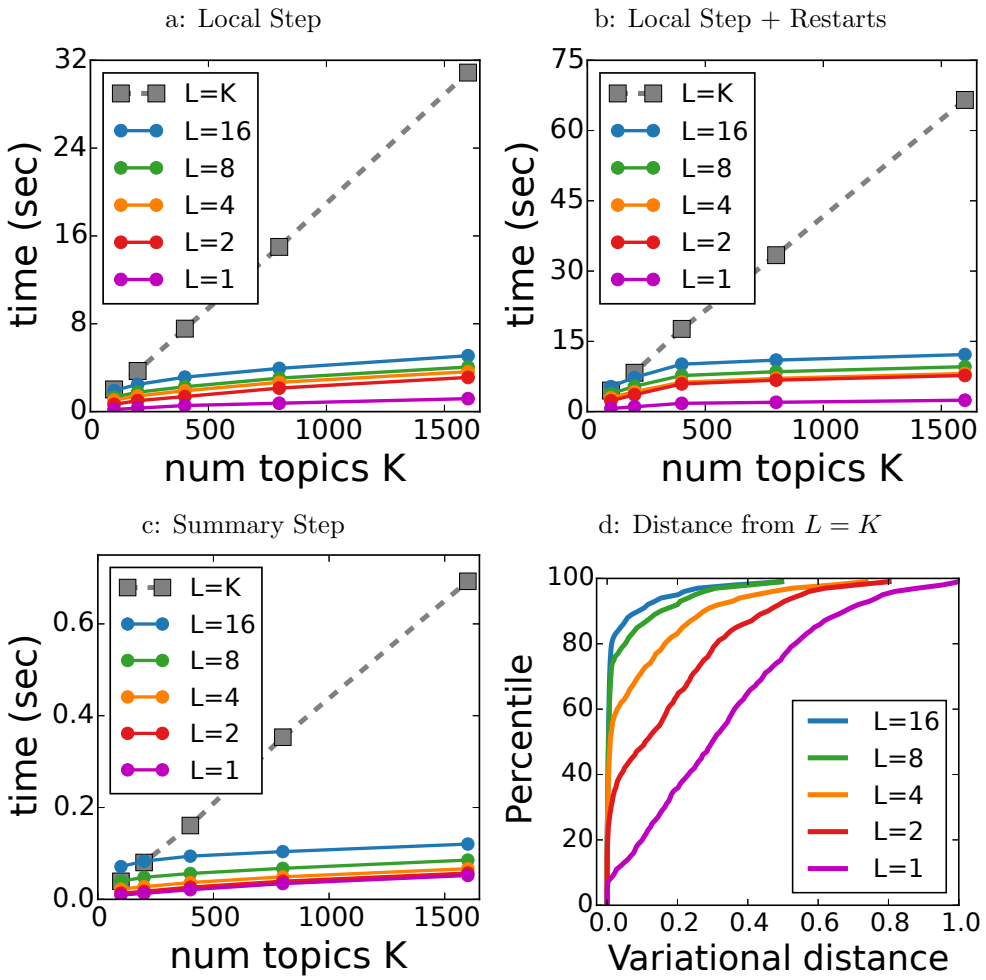

Figure 7.3: Speed and accuracy of L-sparse assignment posteriors for training topic models Influence of sparsity-level L on the different substeps of training a topic model from one mini-batch of 1000 NY Times articles. L defines the number of non-zero entries in the assignment vector $\hat{r}_{dn}$ for each observation n in document d. *Panel a:* Timings for the local step of the LDA topic model run for a maximum of 100 iterations at each document. With $K = 500$ topics, moderate sparsity $L = 8$ gives a throughput of over 100 documents per second, while the dense approach can process only 30 documents per second. *Panel b:* Same as *a*, but with additional restart proposals. *Panel c:* Timings for the summary step of the topic model, which computes the effective count of each word type in each topic. Even though this step takes little time, our sparse representations can speed it up. *Panel d:* Define the empirical topic distribution of document d by normalizing the count vector $[N_{d1} \dots N_{dK}]$. For each document in the minibatch, we compute the variational distance between the empirical distribution produced byTopLRespForDoc at a given L to that produced by DenseStepForDoc. This plots shows the empirical CDF of this distance across all 1000 documents.

Updating $\hat{\theta}_d$, which defines $q(\pi_d)$. We have a closed-form update for the document-topic pseudocounts $\hat{\theta}_d$ given fixed assignments $\hat{r}_{dn}$.

$$\hat{\theta}_{dk} = N_{dk}(\hat{r}_d) + \alpha/K \tag{7.10}$$

Here, $N_{dk} \triangleq \sum_{n=1}^{N} \hat{r}_{dnk}$ counts the number of tokens assigned to topic k in document d.

Updating $\hat{r}_{dn}$, which defines $q(z_{dn})$. Under the usual dense representation, the optimal update for the assignment vector of token n has a closed form like the mixture model, but with document-specific weights:

$$\begin{aligned} \hat{r}_{dn} &\leftarrow \textsc{RespFromWeights} \qquad ([W_{dn1} \dots W_{dnk}]), \qquad (7.11) \\ W_{dnk}(x_{dn}, \hat{\theta}, \hat{\lambda}) &\triangleq \mathbb{E}_q[\log \pi_{dk} + \log \phi_{k x_{dn}}], \\ \mathbb{E}_q[\log \pi_{dk}] &\triangleq \psi(\hat{\theta}_{dk}) - \psi(\textstyle\sum_{\ell=1}^{K} \hat{\theta}_{d\ell}). \end{aligned}$$

We can easily incorporate our sparsity-level constraint to enforce at most L non-zero entries in $\hat{r}_{dn}$. TopLRespFromWeights still provides the optimal solution in this case.

Sharing parameters by word type. Naively, tracking the assignments for document d requires explicitly representing a separate K-dimensional distribution for each of the N_d tokens. However, we can save memory and runtime by recognizing that for a token with word type v, the optimal value of Eq. (7.11) will be the same for all tokens in the document with the same type. We can thus share parameters with no loss in representational power, requiring U_d separate K-dimensional distributions, where $\hat{r}_{dn} \triangleq \hat{r}_{du_{dn}}$.

Iterative joint update for $q(\pi_d)$ and dense $q(z_d)$. When visiting a new document, we must infer both $q(\pi_d|\hat{\theta}_d)$ and $q(z_d|\hat{r}_d)$. Following standard practice for dense assignments, we use an block-coordinate ascent algorithm that iteratively loops between Eq. (7.10) and Eq. (7.11), as shown in Alg. 5.1. When computing the log posterior weights W_{duk}, two easy speed-ups are possible: First, we need only evaluate $C_{vk} = \mathbb{E}_q[\log \phi_{kv}]$ once for each word type v and topic k and reuse the value across iterations. Second, we can directly compute the effective log prior probability $P_{dk} \triangleq \mathbb{E}_q[\log \pi_{dk}]$ during iterations, and instantiate $\hat{\theta}$ after the algorithm converges.

To initialize the update cycle for a document, we recommend visiting each token n and updating it with initial weight $W'_{dnk} = \mathbb{E}_q[\log \phi_{k x_{dn}}]$. This lets the topic-word likelihoods drive the initial assignments. We then alternate between Eq. (7.11) and Eq. (7.10) until either a maximum number of iterations is reached (typically 100) or the maximum change of all document-topic counts N_{dk} falls below a threshold (typically 0.05).

Each iteration updates P_{dk} with cost $O(K)$, and then performs U_d evaluations of the token-specific responsibility update RespFromWeights, each with dense cost $O(K)$. On most datasets, we find local iterations are the dominant computational cost, exceeding 90% of the runtime.

Iterative joint update with sparsity. Under the constraint that each responsibility vector has at most L non-zero entries, we can use an alternative iterative algorithm we call TopLRespForDoc in Alg. 5.1. This procedure has several advantages over DenseRespForDoc. First, we have already shown in Fig. 7.1 that the cost of the subprocedure TopLRespFromWeights is much less than

RespFromWeights. Second, we further assume that once a topic's mass N_{dk} decays near zero, it will never rise again. With this assumption, at every iteration we identify the set of active topics (those with non-neglible mass) in the document: $\mathcal{A}_d \triangleq \{k : N_{dk} > \epsilon\}$. Only these topics will have weight large enough to be chosen in the top L for any token. Thus, throughout TopLRespForDoc we need only loop over the active set, and each iteration costs $O(|\mathcal{A}_d|)$ instead of $O(K)$.

Discarding topics whose mass within a document drops below ϵ is justified by previous empirical observations of the so-called "digamma problem" described in Mimno et al. (2012): for topics with negligible mass, the expected log probability term becomes vanishingly small. For example, $\psi(\frac{\alpha}{K}) \approx -200$ for $\alpha \approx 0.5$ and $K \approx 100$, and gets smaller as K increases.

In practice, after the first few iterations the active set stabilizes and each token's top L topics rarely change while the relative responsibilities continue to improve. In this regime, we can amortize the cost of TopLRespForDoc by avoiding selection altogether, instead just reweighting each token's current set of top L topics. We perform selection for the first 5 iterations and then only every 10 iterations, which yields large speedups without loss in solution quality.

Fig. 7.3 compares the runtime of TopLRespForDoc across values of sparsity-level L against a heavily-optimized version of DenseRespForDoc, which includes precomputing the exponentiated conditional likelihoods and uses optimized dense-matrix multiplication routines. For small numbers of topics $K < 100$, our gains are modest at best. However, for $K \gg 100$ our sparsified algorithm can process 1000 documents at least three times faster, with relative speed increasing with K.

Restart proposals. Our previous work in Sec. 5.3.4 (Hughes et al., 2015a) introduced *restart* proposals, a post-processing step to single document inference that can dramatically improve performance. Given the output count vector N_d from Alg. 5.1, the restart proposal constructs a candidate N'_d by setting an active entry of N_d to zero and then running two iterations forward. We then accept the new count vector if it improves the objective $\mathcal{L}$. We find these proposals are crucial to escape local optima, so we always include them. A side-by-side comparison of costs with and without restarts is in Fig. 7.3.

7.4 Experimental results with L-sparse topic models

7.4.1 Topic Modeling Experiments

We train topic models at several sparsity-levels using **Memo**ized and **Stoch**astic algorithms. We compare to the fast-yet-exact **SparseGibbs** (sometimes called SparseLDA) sampler of Yao et al. (2009) and the sampler-inside-stochastic **GibbsInSVI** method of Mimno et al. (2012). These external methods use Java code available in Mallet (McCallum, 2002). External methods use the default Mallet initialization, while we use **AnchorWords** (Arora et al., 2013) to initialize $q(\phi)$.

Sparsity Hyperparameters. Our primary goal is quantifying how our performance varies with the sparsity-level L. **GibbsInSVI** allows control of the number of samples S used to approximate

Algorithm 7.3 Update for document-specific responsibilities under standard topic model

Input:

α : document-topic smoothing scalar

$\{\{C_{vk}\}_{k=1}^K\}_{v=1}^V$: log prob. of word v in topic k

$C_{vk} \triangleq \mathbb{E}_q[\log \phi_{kv}]$

$\{v_{du}, c_{du}\}_{u=1}^U$: word type/count pairs for doc. d

Output: $\hat{r}_d$: dense responsibilities for doc d

function DENSERESPFORDOC(C, α, v_d, c_d)
  for $u = 1, \dots U$ **do**
    $\hat{r}_{du} =$ RESPFROMWEIGHTS($C_{v_{du}}$)
  while not converged **do**
    for $k = 1, 2 \dots K$ **do**
      $N_{dk} = \sum_u c_{du} \hat{r}_{duk}$, $P_{dk} = \psi(N_{dk} + \frac{\alpha}{K})$
    for $u = 1, 2 \dots U$ **do**
      for $k = 1, 2 \dots K$ **do**
        $W_{duk} = C_{v_{du}k} + P_{dk}$
      $\hat{r}_{du} =$ RESPFROMWEIGHTS(W_{du}) ▷ See Alg. 7.1
  return $\hat{r}_d$

Algorithm for computing responsibilities for a single document given fixed topics for a multinomial likelihood and bag-of-words data. Without sparsity constraints, each step scales with $O(K)$.

Algorithm 7.4 Update for document-specific responsibilities under L-sparse topic model.

Input:

α : document-topic smoothing scalar

$\{\{C_{vk}\}_{k=1}^K\}_{v=1}^V$: log prob. of word v in topic k

$C_{vk} \triangleq \mathbb{E}_q[\log \phi_{kv}]$

$\{v_{du}, c_{du}\}_{u=1}^U$: word type/count pairs for doc. d

Output: $\hat{r}_d$: $L-$sparse responsibilities for doc d

function TOPLRESPFORDOC(C, α, v_d, c_d, L)
  for $u = 1, \dots U$ **do**
    $\hat{r}_{du} =$ TOPLRESPFROMWEIGHTS($C_{v_{du}}, L$) ▷ See Alg. 7.2
  while not converged **do**
    for $k \in \mathcal{A}_d$ **do**
      $N_{dk} = \sum_{u=1}^U c_{du} \hat{r}_{duk}$
    $\mathcal{A}_d = \{k \in \mathcal{A}_d : N_{dk} > \epsilon\}$
    for $k \in \mathcal{A}_d$ **do**
      $P_{dk} = \psi(N_{dk} + \frac{\alpha}{K})$
    for $u = 1, 2 \dots U$ **do**
      for $k \in \mathcal{A}_d$ **do**
        $W_{duk} = C_{v_{du}k} + P_{dk}$
      $\hat{r}_{du} =$ TOPLRESPFROMWEIGHTS(W_{du}, L)
  return $\hat{r}_d$

Algorithm for computing responsibilities for a single document given fixed topics for a multinomial likelihood and bag-of-words data. Accomplishes the same goal as RESPFORDOC but forces each observation to use at most L topics. This change, plus tracking only the active topics in a document $\mathcal{A}_d$ leads to much faster iterations than the conventional RESPFORDOC.

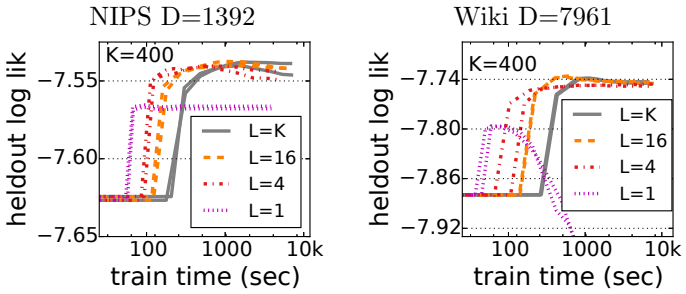

Figure 7.4: Comparison of values for sparsity-level L on topic models
Analysis of 1392 NIPS articles (*left*) and 7961 Wikipedia articles (*right*). These trace plots of heldout likelihood over time represent an internal comparison of different possible L values for memoized inference. We find that hard $L = 1$ assignments can plauteau early or fail catastrophically due to the many local optima in multinomial topic models. In contrast, moderate L values like 4 or 16 produce the same high-quality predictions as dense $L = K$ representations in a fraction of the time (note the log-scale of the x-axis).

$q(z_d)$. We consider $S = \{5, 10\}$, always discarding half of these samples as burn-in. **SparseGibbs** has no sparsity parameter.

Nuisance Hyperparameters. For all methods, we set document-topic smoothing $\alpha = 0.5$ and topic-word smoothing $\bar{\lambda} = 0.1$. We set the stochastic learning rate at iteration t to $\rho_t = (\delta + t)^{-\kappa}$. We use grid search to find the best heldout score on validation data, considering delay $\delta \in \{1, 10\}$ and decay $\kappa \in \{0.55, 0.65\}$.

We evaluate all methods on two key metrics: wallclock time and predictive power. Following Wang et al. (2011), we measure heldout performance via a document completion task. Given a heldout document x_d, we divide its words at random by type into two pieces: 80% in x_d^A and 20% in x_d^B. We use subset A to estimate the document-specific probabilities $\hat{\pi}_d$, and then evaluate the predictions of this estimate on the remaining words in B. Throughout, we point estimate each topic k to the trained posterior mean $\hat{\phi}_k = \mathbb{E}_q[\phi_k]$. Across many heldout documents, we measure the following score:

$$\text{score}(x^A, x^B, \hat{\phi}) = \frac{\sum_d \sum_{n=1}^{|x_d^B|} \log \sum_k \hat{\pi}_{dk} \hat{\phi}_{k x_{dn}^B}}{\sum_d |x_d^B|}$$

For all algorithms, we fix a point estimate of topics $\hat{\phi}$ from training and then estimate $\hat{\pi}_d$ in the same way: finding the optimal $q(\pi_d^A)$ and $q(z_d^A)$ for the words in the first piece x_d^A by using DenseRespForDoc. Finally, we take $\hat{\pi}_d = \mathbb{E}_q[\pi_d^A]$ and compute the heldout likelihood of x_d^B.

We first conduct careful comparisons on datasets which are small enough for all algorithms to make many complete laps (effective passes through the dataset). The first two rows of Fig. 7.5 show performance on 1392 NIPS articles and 7961 Wikipedia articles. The final row shows performance for the much larger New York Times Annotated Corpus: 1.8 million articles from 1987 to 2007, with 9000 documents per batch (200 batches total). Across all datasets, we reach the same conclusions:

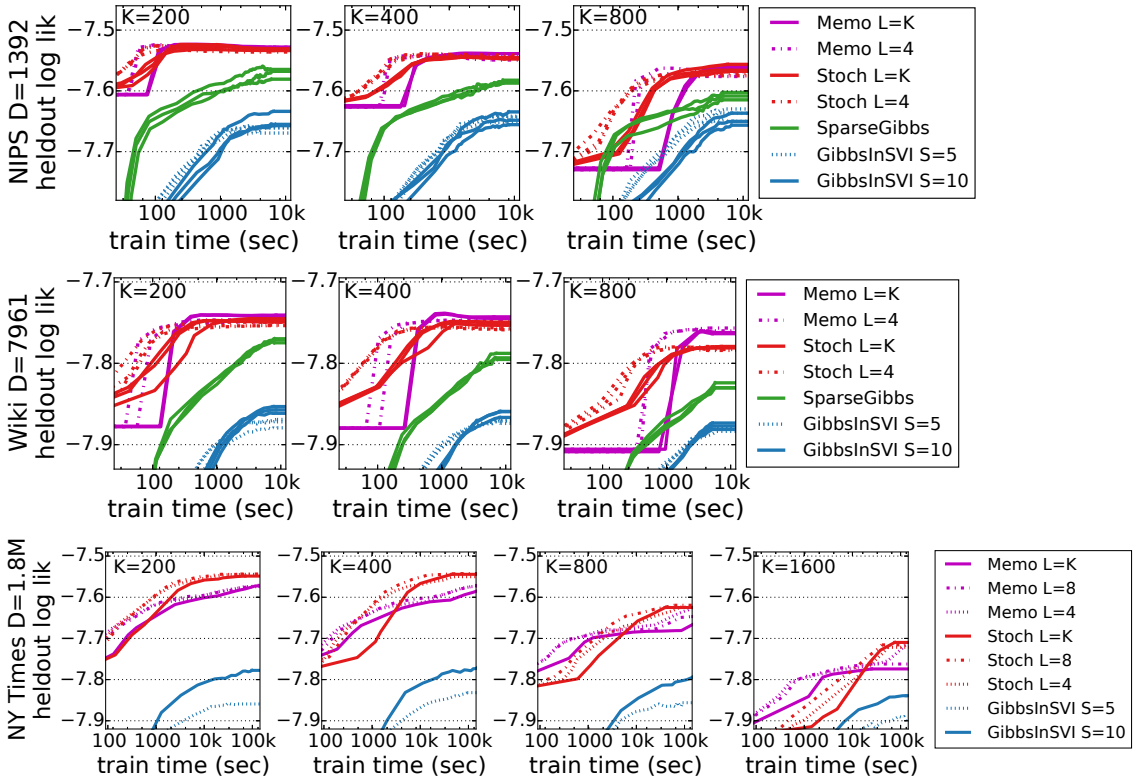

Figure 7.5: L-sparse topic model results on NIPS, Wikipedia, and NYTimes
Analysis of 1392 NIPS articles (*top row*), 7961 Wikipedia articles (*middle*), and 1.8 million New York Times articles (*bottom*). We use 5 batches for the small datasets, and 200 batches for NY Times. Each panel plots heldout scores (higher is better) over time for sparse and dense versions of our algorithms and external baselines. The number of clusters K varies from left to right.

Moderate sparsity is best. Throughout Fig. 7.5, we see that runs with sparsity-levels of $L = 4$, $L = 8$, or $L = 16$ under both memoized and stochastic algorithms converge several times faster than $L = K$ but yield indistinguishable predictions.

Hard $L = 1$ assignments can be pathological. As shown in Fig. 7.4, running memoized inference with $L = 1$ may either plateau early at noticeably worse performance (e.g. the left panel plot using the NIPS dataset) or fall into progressively worse local optima (e.g. the right panel plot using the Wiki dataset). Remember, neither memoized nor stochastic inference for LDA topic models has a monotonicity guarantee because both $q(z_d)$ and $q(\pi_d)$ are re-estimated from scratch each time we visit a document. When $L = 1$ hard assignments, the local step inference can have a strong tendency to use many different topics to explain a document given a likelihood-first initialization of the local coordinate ascent iterations. This can explain the pathological behavior of $L = 1$ assignments.

External methods converge slowly. Throughout Fig. 7.5, no run of **SparseGibbs** or **GibbsInSVI** reaches competitive predictions in the allowed time limit (3 hours for NIPS and Wiki, 2 days for NYTimes). **GibbsInSVI** benefits from more samples ($S = 10 > S = 5$) only on NYTimes. More than 10 samples did not improve performance further. Mimno et al. (2012) also found diminishing returns from tuning the number of MCMC samples S used to estimate expectations.

7.5 Discussion

In this chapter, we have introduced a simple sparsity constraint on the parameters of a variational posterior which leads to faster training times, equal or better predictive power, and more intuitive interpretation. Our algorithms can be dropped-in to any ML, MAP or full-posterior variational clustering objective and are simple to parallelize and scale to millions of examples.

We anticipate further research in adapting $L > 1$ sparsity to sequential models like HMMs, which might lead to a dynamic programming algorithms for sequential models that might scale quadratically with L rather than K. This would speed up the largest existing bottleneck in using the training algorithms from Ch. 6.

We also anticipate applying these ideas to other factors in the approximate posterior. For example, for multinomial topic models we might directly constrain the shape parameters $q(\phi_k)$ so that at most W vocabulary words have non-zero probability. We could also maybe encode some relaxation of an *anchor-word* assumption (Arora et al., 2013). These kinds of constraints could improve interpretability as well as speed.

Chapter 8

Recommendations

Throughout this thesis, we have presented a unified framework for training Bayesian nonparametric probabilistic models models via variational optimization algorithms. In this chapter, we discuss four possible extensions to our work which would make it applicable to a much broader family of modeling scenarios and improve overall effectiveness. We review each recommendation for future work below, with further detailed discussion in later sections.

Sec. 8.1: Parallelization for extreme scalability. Scaling our optimization algorithms up to billions of examples presents several technical challenges. Several of our baseline algorithms for fixed-truncation inference are "embarrassingly" parallelizable. Creating parallelized implementations of birth, merge, and delete proposal moves remains an engineering challenge, though the general task seems feasible. We hope that parallel implementations together with possible feed-forward approximations to the local step provide a path to reliable clustering on huge datasets that can still adequately solve the model selection problem.

Sec. 8.2: Approximation-quality guarantees for variational optimization. Both the Bregman divergence k-means++ initialization algorithm in Alg. 2.2 (Ackermann and Blömer, 2010) and the original k-means++ algorithm of Arthur and Vassilvitskii (2007) offer approximation-quality guarantees relative to the globally-optimal clustering which are quite good. Spectral methods have also recently drawn attention as provable methods for point estimation in related clustering problems (Hsu et al., 2012). To our knowledge, no work exists in examining whether these guarantees would apply for the type of variational optimization objectives we explore in this thesis. Providing such a guarantee for either some initialization procedure, or more ideally for some proposal procedures, could make our optimization approach much more attractive.

Sec. 8.3: Semi-supervised clustering. Many clustering applications come with some additional side information, such as the five-star ratings that accompany Yelp reviews or the low-level and high-level action labels from motion capture datasets. We envision coherent methods for training

large-scale BNP models from the union of small curated datasets with supervised labels and millions of unlabeled data observations.

Sec. 8.4: Extension to probabilistic programming. The adaptive-proposal optimization methods we have presented for mixture, topic, and hidden Markov model should be extensible to a broad family of Bayesian nonparametric clustering models. We envision a model specification language for this family as well as automatic inference in the spirit of probabilistic programming. We hope our practical BNPy software could evolve into this system, supporting a narrower set of models than alternatives like Stan (Carpenter et al., 2015) but offering scalable and reliable inference for this specialized family.

8.1 Parallelization and other tricks for extreme scalability

Two possible ways to speed up our variational optimization for Bayesian nonparametric clustering models for scaling to billions of examples are: finding faster algorithms (in terms of order-of-growth runtime as well as practical speed) and deploying these algorithms in parallelized hardware. For example, the L-sparse approximations from Ch. 7 led to much faster algorithms by reducing some substeps from $O(K)$ runtime cost to $O(L)$. We first discuss parallelized implementation of our existing algorithms, identifying bottlenecks and opportunities. We later mention some approximations we can make to the local step of optimization which may yield substantial speed improvements.

Taking advantage of parallelization is a must for an algorithm to gain traction on industry-scale applications. We have preliminary success scaling our existing fixed-truncation algorithms for topic models and hidden Markov models to the single-machine, multiple-cores setting. Fig. 6.6 shows over 25x speed improvement when using 64 cores on data at the scale of the human genome. We are excited about performing adaptive birth, merge, and delete proposals inference in a truly distributed setting, with dozens of parallel workers each creating new clusters from disjoint batches and integrating them into a coherent centralized model. Some early work on merge proposals in this multiprocessor setting has been done by Campbell et al. (2015) for an alternative variational approach to HDP topic models. Implementing such parallelization into BNPy would make our clustering methods accessible to many industry-scale applications.

Procedurally, the bottleneck of inference across all our models is the runtime cost of the "local" step, where we assign data to clusters. Rosenbaum and Weiss (2015) studied the image patch modeling application from Fig. 1.2 and showed that applying a pre-trained Gaussian mixture model to test images can be dramatically accelerated by a feed-forward approximation of the local step. Other recent approaches (Mnih and Gregor, 2014) combine the fast, feed-forward properties of neural networks during training itself, optimizing the feed-forward network as part of the the Bayesian variational optimization problem. By training a feed-forward network to approximate the local posterior of $q(z_n)$, as done by Gan et al. (2015), these methods use information from previous examples to cluster new data faster. With some of these tricks, our memoized algorithm could train models

with thousands of clusters and billions of examples.

8.2 Approximation guarantees for variational optimization

8.2.1 Distance-biased random initializations with guarantees

Our Bregman k-means++ initialization in Alg. 2.2 is motivated by the k-means++ algorithm Arthur and Vassilvitskii (2007), with the extension to Bregman divergences motivated by Theorem 1 of Ackermann and Blömer (2010). Formally, Ackermann and Blömer (2010) prove that if a dataset satisfies basic separability conditions and the Bregman divergence function can be bounded by a Mahalonobis distance, then a k-means++ initialization yields a constant-factor approximation to the solution quality with probability $2^{-O(K)}$. Some of this work appears in an earlier conference paper (Ackermann and Blömer, 2009). Developing extensions that justify this procedure for all Bregman divergences, not just those for which the Mahalonobis distance bound applies, is of great interest.

Acharyya et al. (2013) show how to generalize any Bregman divergence to a "general symmetric" divergence function that is symmetric and obeys the triangle inequality. They prove kmeans++ style guarantees for this class of "general symmetric" divergences. This generalization voids the interpretation of our procedure in Alg. 2.2 as solving a MAP estimation problem for exponential families. However, such general properties may prove useful for achieving formal guarantees.

More scalable guarantees

The k-means++ initialization requires K total passes through the full dataset, or whatever subset is used for initialization. In each pass, exactly one additional cluster mean is chosen. Each step is conditioned on results of the previous pass, so the procedure isn't easily parallelized. When the number of clusters K is large, in the hundreds or thousands, this can be quite expensive.

Bahmani et al. (2012) presents an alternative initialization algorithm with similar performance guarantees but much cheaper initialization. Their procedure requires only a small constant (≈ 5) number of passes through the dataset in practice, instead of K total passes. The basic idea is that we commit to doing R passes, and at each round we select each data observation x_n as a possible cluster center with probability:

$$p_n \propto LD(x_n, \mu_n) \tag{8.1}$$

where L is a user defined positive scalar. After all the R rounds, the total number of chosen clusters will be a random variable, though its expected value can be controlled by the parameter L as detailed in Bahmani et al. (2012). If exactly K clusters are needed, we can apply post-processing heuristics to select these.

This sampling technique is easy to generalize to our Bregman divergences case in practice. Showing that the guarantees apply requires sophisticated theoretical arguments to get around the triangle inequality.

8.2.2 Provable spectral algorithms

Another prominent line of recent research are so-called *spectral* methods. These approaches exist for finite mixture models, topic models (Arora et al., 2013), hidden Markov models (Hsu et al., 2012), and probabilistic context-free grammars (Cohen et al., 2014). The primary advantage of these methods is that they deliver very fast point estimates of global parameters. Often such methods come with formal guarantees of solution quality, but these require two assumptions. First, they assume infinitely many training examples. Second, they assume that the data is truly generated by the underlying likelihood model. Guarantees like those in Arora et al. (2013) do note exploit regularization from the prior or give any quality guarantees when the model is an approximation of the true generating process. Nevertheless, finding ways to more closely connect these methods with our Bayesian nonparametric approach could yield promising theoretical results, and we have found such procedures useful as initializations for our methods in practice.

8.3 Extensions for semi-supervised clustering

In many applications, supervised information like the number of stars accompanying an online review or the number of links to some Wikipedia article may be of interest. Often, supervised information is expensive to obtain, so we seek approaches that can easily handle joint learning of latent cluster representations when only a subset of the data has observed response variables. This can include cases where every data token has a response we'd like to predict, or cases where groups of tokens have a single response (such as labels for whole documents or entire sequences).

Throughout this section, we'll make discussion concrete by fixating on a particular example: prediction of document-level labels via a HDP topic model. Here, each document d has one observed response variable y_d. The response y_d may be binary, multi-class, or real-valued in practice, though each choice requires slightly different inference machinery. We'll take it to be real-valued throughout this running example, which implies our intended task is *regression* rather than classification. Our goal is to find clusters or topics that predict both the observed data x_d and this document-level response variable y_d.

We consider two conceptual approaches for achieving our goal of finding latent clusters that predict response variables. The first conceptual approach comes from prediction-focused *generative* models. This tradition treats response variables like any other random variable in the graphical model. For our target task of document-level regression, the most prominent example is sLDA – supervised Latent Dirichlet Allocation (Mcauliffe and Blei, 2008) – which models response variable y_d as a child of topic indicators z_d. Extensions have modified this parametric model to a proper BNP model via the HDP (Zhang et al., 2013) using fixed-truncation variational inference.

The second conceptual tradition is *maximum entropy discrimination* (MED). This approach takes a more purely discriminative view that integrates topic models with the machinery behind support vector methods, which are widely used for classification and regression. MED methods define specific margin constraints which a proposed cluster-to-response prediction model must satisfy. For the topic

modeling problem, these constraints force the hidden clusters to predict the response y_d within some tolerance, and encourage big changes to the topic assignments of any documents that violate these constraints. Zhu et al. (2012) proposed a supervised topic model in this tradition called MED-LDA, inspired by Jebara's earlier work with more general prediction problems (Jebara, 2001). The original MED-LDA used variational inference, while recent efforts have developed a data-augmented Gibbs sampler for parametric topic models (Zhu et al., 2013) and Bayesian nonparametric HMMs (Zhang et al., 2014). Below, we show how both the sLDA and MED-LDA variational approaches are special cases of a loss-augmented version of our original variational objective $\mathcal{L}$. This view allows us to compare and contrast the two approaches more closely.

Alternative *upstream* generative approaches are not considered here. These methods do not align with our focus on predictive performance, but still pursue label-informed clustering for text data (Mimno and McCallum, 2008; Lacoste-Julien et al., 2009), relational models (Kim et al., 2012), and visual scene categorization (Li et al., 2009). Upstream models tell a generative story in which labels y_d are the parents of topic assignments z_d, rather than the opposite story told by *downstream* models like sLDA. Simply put, these models are not set up for predicting labels. In fact, under these models both imputing or marginalizing away missing labels can be computationally difficult. Nevertheless, these models offer interesting ways to inform latent representations when metadata is available, and could be a direction for future work.

Possible variational optimization objectives for supervision

Adding supervision to the topic model from Ch. 5 requires specifying the prediction rule for producing response variable y_d from the assignments z_d. Following Mcauliffe and Blei (2008), we choose a simple linear prediction rule given approximate posterior $q(z_{dt})$ at each token t in document d parameterized by responsibility vector $\hat{r}_{dt}$.

$$y_d = w^T \mathbb{E}[\bar{z}_d], \quad \text{where } \mathbb{E}[\bar{z}_d] \triangleq \frac{1}{T_d} \sum_{n=1}^{T_d} [\hat{r}_{dt1} \ \hat{r}_{dt2} \ \ldots \hat{r}_{dtK}] \tag{8.2}$$

Here, $\bar{z}_d$ is a K-length vector of empirical topic frequencies in document d. Each vector z_d has positive entries that sum to one. The parameter w describes a linear transformation from vector $\bar{z}_d$ to scalar y_d, where w is a K-length vector of weight coefficients.

Extending our general unsupervised objective $\mathcal{L}$ for topic models from earlier chapters to incorporate this prediction model for response variables y_d, we arrive at the following general form:

$$\mathcal{L}^s(w, \hat{r}, \hat{\theta}, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}) = \mathcal{L}_{alloc}(\hat{r}, \hat{\theta}, \hat{u}, \hat{\omega}) + \mathcal{L}_{data}(x, \hat{r}, \hat{\tau}, \hat{\nu}) + R(w) - \sum_{d=1}^{D} \text{Loss}(y_d, w^T \mathbb{E}[\bar{z}_d]) \tag{8.3}$$

We still seek to maximize this general *supervised* objective $\mathcal{L}^s$ by finding the ideal free parameters $\hat{r}, \hat{\theta}, \hat{\tau}, \hat{\nu}, \hat{u}, \hat{\omega}$ as before. However, by incorporating supervised labels this optimization now requires finding cluster indicators that explain the data x *and* minimize the loss in predicting the responses y_d at each document via the cluster indicators. Now, specifying our objective requires both a concrete loss function Loss and a choice of regularization R on the weight vector. Both sLDA and MED-LDA are special cases of this objective, each with a specialized choice of loss function.

sLDA objective. Prediction-focused generative models specify a loss function for the response variable y_d. When y_d is real-valued, a natural model is a simple Gaussian $y_d \sim \mathcal{N}(w^T \mathbb{E}[z_d], \lambda^{-1})$, which implies the following form of the loss function

$$\text{Loss}^{sLDA}(y_d, w^T \mathbb{E}[z_d]) \triangleq -\log p(y_d | w, z_d) = \frac{\lambda}{2}(y_d - w^T \mathbb{E}[z_d])^2 \tag{8.4}$$

Here, $\lambda > 0$ is a scalar precision: larger values imply more precise estimates of y_d.

The major problem with this objective is that it forces learned clusters to explain both the observed data x_d and the response y_d, with no clear encoding of how to manage the tradeoff. When the number of observed tokens T_d in a document is large, this can be problematic. The learned topics must provide good explanations for T_d tokens *and* predict y_d with precision λ. Without carefully setting λ, the clusters can easily favor configurations that explaing the tokens well but not the response, since the objective weights tokens over response predictions by a factor of (roughly) T_d/λ. Thus, tuning the parameter λ requires careful attention for good performance.

MED-LDA objective. For regression, MED-LDA employs an $\epsilon-$insensitive loss function:

$$\text{Loss}^{MED-LDA}(y_d, w^T \mathbb{E}[z_d]) \triangleq C \max(0, |y_d - w^T \mathbb{E}[z_d]| - \epsilon) \tag{8.5}$$

Here, ϵ is a free parameter that specifies tolerance to error. Setting $\epsilon = 0.1$ on the Yelp example would encode that errors of less than 0.1 on the 5 star scale should not be penalized. No such free parameter exists in the sLDA objective. $C > 0$ is a scalar cost parameter. Like λ above, it must be tuned carefully during training.

We can alternatively express the unconstrained objective $\mathcal{L}^s$ for MED-LDA as a constrained optimization problem, with a constraint for every document requiring that that $|y_d - w^T \mathbb{E}[\bar{z}_d]| \leq \epsilon + \xi_d$. This constraint enforces the predictions to lie within tolerance ϵ. Slack variables ξ_d allow some predictions to exceed this desired tolerance since a perfect model may not be possible, but penalize this transgression via increased loss.

Comparing the MED-LDA objective with sLDA, we see that the MED-LDA approach more naturally lends itself to the goal of strong predictions. Both MED-LDA free parameters ϵ and C are directly interpreted as a desired tolerance and a penalty for violating this tolerance. In particular, this conceptual view has no trouble with tuning C differently as more data arrives. However, with sLDA the idea of tuning λ to match predictive performance runs contrary to the generative approach. Furthermore, the generalization performance associated with the MED machinery brings further potential benefits.

Inference for the MED-LDA objective above is quite tractable. The updates to all our earlier variational free parameters do not change at all, except for the cluster indicator parameters $\hat{r}_d$. Finding the optimal weight vector w can seen as a solution to a quadradic program when function $R(w)$ encodes an L2 penalty. Numerical solvers for these kind of QPs are well-studied. For improved regularization, we may seek free parameters for an propor variational distribution $q(w)$, rather than a simple point estimate of w. This is the approach taken by Zhu et al. (2012) and will be essential for properly comparing models of different truncation levels.

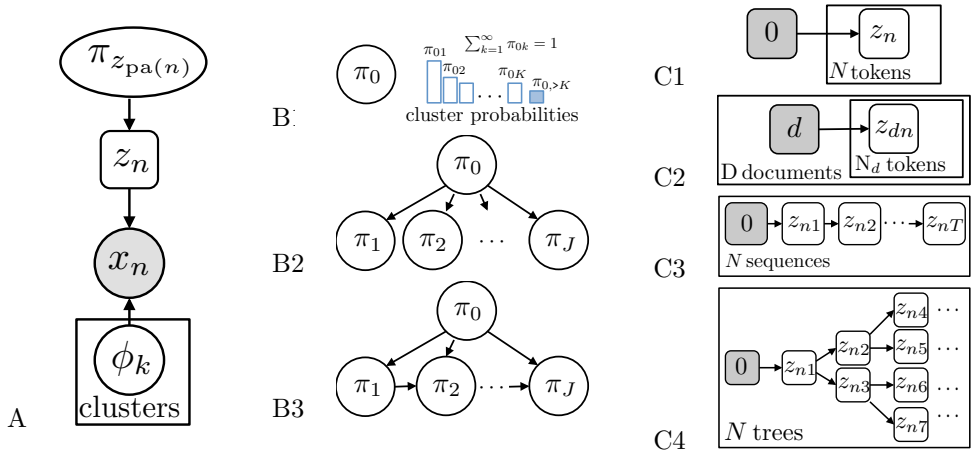

Figure 8.1: A probabilistic programming language for specifying clustering models
Our compositional view of clustering models. *Col. A:* Generative model for one data token x_n. *Col. B:* Possible dependency graphs for cluster probability vectors π: DP (top), HDP (middle), and dependent DPs (bottom). *Col. C:* Possible graphs for cluster indicators z.

The experiments of Zhu et al. (2012) clearly indicate the need for a method that can adapt the truncation level. Several experiments studying predictive performance as a function of the number of topics show unsurprisingly that using too few topics can severely handicap predictions, while too many topics can cause slightly worse predictions while requiring more computation. Methods which can reliably explore the combinatorial space of clusterings via non-local changes will be quite important for finding interpretable models that have good predictive power.

8.4 Extensions with probabilistic programming

A prominent trend in modern machine learning research is the development of powerful, general-purpose representation languages called probabilistic programming languages (PPLs) These languages allow the user to specify complex probabilistic models as a generative processs, and then *automatically* perform inference for the specified model given data. PPLs such as Stan (Carpenter et al., 2015), WebPPL (Goodman and Stuhlmüller, 2014), and Venture (Mansinghka et al., 2014) vary in the domain of possible models but generally use a version of MCMC sampling or particle methods to perform inference. CrossCat (Mansinghka et al., 2015) is a recent example of using probabilistic programming for Bayesian nonparametric models of data tables.

We have a vision for a PPL specific to a broad family of clustering models which includes mixtures, topic models, and hidden Markov models as well as many more possibilities.

8.4.1 A preliminary model specification language

Our preliminary model specification language is illustrated in Fig. 8.1. As in the models discussed in this thesis, the unifying property of all models our language supports is that each data token x_n

is generated from a single cluster indicated by discrete variable $z_n \in \{1, 2, \dots K, \dots\}$. The chosen cluster $z_n = k$ specifies the *observation model* for data x_n. If $z_n = k$, we draw x_n from an exponential family (EF) likelihood density with natural parameter ϕ_k: $p(x_n|\phi_k) \propto \exp[s(x_n)^T \phi_k]$, where $s(x_n)$ is a sufficient statistic for the observation model, as detailed in Ch. 2.

Specifying an allocation model. The allocation of cluster assignments z requires a set of frequency vectors π and indicators z. The number of assignment variables z_n is exactly equal to the number of data tokens. The number of π_j variables varies from model to model, with each vector π_j a non-negative vector that sums to unity with an entry for each cluster. By choosing a fixed graph structure for π and z, we encode structural assumptions into the model, as shown in Fig. 8.1.

Each assignment z_n is drawn from a frequency distribution over clusters defined by exactly one π_j node. The value of the parent of node z_n in the z graph determines which π_j variable is used to generate z_n:

$$p(z_n = k|z_{\text{pa}(n)} = j) = \pi_{jk} \tag{8.6}$$

This requires each z_n variable to have exactly one parent, so the topology of the z graph must be restricted to trees (or multiple trees). However, the π graph may be any directed a-cyclic graph.

Our framework defines a single *allocation* model by combining fixed graph structures for π, z from columns B and C. Each model can be either parameteric or nonparametric, based on the prior distribution of the top-level π^G. The pair (B1,C1) yields mixture models (Blei and Jordan, 2006), while (B2, C2) gives topic models (Blei et al., 2003; Teh et al., 2006), and (B2, C3) gives hidden Markov models (Beal et al., 2001). The pair (B2, C4) yields hidden Markov trees used for multi-scale image modeling (Crouse et al., 1998; Kivinen et al., 2007) and text parsing (Finkel et al., 2007; Liang et al., 2007). B3 and C2 could yield a topic model where frequencies vary over time, as in (Blei and Lafferty, 2006). This framework also extends to relational block models (Airoldi et al., 2009; Kemp et al., 2006), hierarchical hidden Markov models (Heller et al., 2009), and spatial models for image segmentation (Sudderth and Jordan, 2009).

8.4.2 Towards general-purpose inference

As a long-term goal, we wish to develop more general-purpose inference methods. Our current code base requires custom implementation of each allocation model (combination of π and z graphs), which makes reusing pieces more difficult than we would like. Therefore, we will investigate developing general-purpose inference algorithms that are modular with respect to the compositional structure of Fig. 8.1, developing a message-passing framework that can handle general π graphs and z trees.

One considerable challenge here is developing scalable and adaptive inference methods that are more general purpose. For example, we might wish to handle a desired likelihood $p(x_n|\phi_k)$ that does not belong to the exponential family. Toward this end, general purpose "black-box" variational methods (Kucukelbir et al., 2015; Ranganath et al., 2014) could inspire new solutions when

combined with the adaptive proposals from this work. There are also possible connections to recent work on variational auto-encoders (Kingma and Welling, 2014). These methods are possible even with models for which the usual mean-field variational objective function is intractable to compute. We are optimistic that with significant research investment, we could develop reliable and scalable general-purpose inference for a wide range of possible Bayesian nonparametric clustering models.

Bibliography

S. Acharyya, A. Banerjee, and D. Boley. Bregman divergences and triangle inequality. In *SIAM International Conference on Data Mining*, 2013.

M. R. Ackermann and J. Blömer. Coresets and approximate clustering for Bregman divergences. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*, 2009.

M. R. Ackermann and J. Blömer. Bregman clustering for separable instances. In *Proceedings of the 12th Scandinavian conference on Algorithm Theory (SWAT)*, 2010.

A. Agarwal and H. Daumé III. A geometric view of conjugate priors. *Machine Learning*, 81(1):99–113, 2010.

E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. In *Neural Information Processing Systems*, 2009.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. In *International Conference on Machine Learning*, 2013.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

M. J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

M. J. Beal and Z. Ghahramani. Variational inference for bayesian mixtures of factor analysers. In *Neural Information Processing Systems*, 1999.

M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Neural Information Processing Systems*, 2001.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1): 121–143, 2006.

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

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448 – 461, 1973.

C. A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *Image Processing, IEEE Transactions on*, 3(2):162–177, 1994.

T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan. Streaming variational Bayes. In *Neural Information Processing Systems*, 2013.

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

T. Campbell, J. Straub, J. W. Fisher III, and J. P. How. Streaming, massively parallel variational inference for bayesian nonparametrics. In *Neural Information Processing Systems*, 2015.

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: a probabilistic programming language. *Journal of Statistical Software*, 2015.

G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

J. Chang and J. W. Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. In *Neural Information Processing Systems*, 2013.

J. Chang and J. W. Fisher III. Parallel sampling of HDPs using sub-cluster splits. In *Neural Information Processing Systems*, 2014.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *The Journal of Machine Learning Research*, 15(1):2399–2449, 2014.

M. S. Crouse, R. D. Nowak, and R. G. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46(4):886–902, 1998.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, pages 1–38, 1977.

J. Ernst and M. Kellis. Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nature Biotechnology*, 28(8):817–825, 2010.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

B. S. Everitt. *Finite mixture distributions*. Wiley Online Library, 1981.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973.

J. R. Finkel, T. Grenager, and C. D. Manning. The infinite tree. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2007.

N. Foti, J. Xu, D. Laird, and E. Fox. Stochastic variational inference for hidden Markov models. In *Neural Information Processing Systems*, 2014.

E. B. Fox. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. PhD thesis, Massachusetts Institute of Technology, 2009.

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. A sticky HDP-HMM with application to speaker diarization. *Annals of Applied Statistics*, 5(2A):1020–1056, 2011.

E. B. Fox, M. C. Hughes, E. B. Sudderth, and M. I. Jordan. Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *Annals of Applied Statistics*, 8(3): 1281–1313, 2014.

Z. Gan, C. Li, R. Henao, D. Carlson, and L. Carin. Deep temporal sigmoid belief networks for sequence modeling. In *Neural Information Processing Systems*, 2015.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 2013.

N. D. Goodman and A. Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2016-3-18.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. In *Neural Information Processing Systems*, 2007.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 2004.

K. A. Heller, Y. W. Teh, and D. Görür. Infinite hierarchical hidden Markov models. In *Artificial Intelligence and Statistics*, 2009.

M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1), 2013.

M. D. Hoffman, D. M. Blei, and F. R. Bach. Online learning for latent Dirichlet allocation. In *Neural Information Processing Systems*, 2010.

M. M. Hoffman, O. J. Buske, J. Wang, Z. Weng, J. A. Bilmes, and W. S. Noble. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature methods*, 9(5):473–476, 2012.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

M. C. Hughes and E. B. Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *Neural Information Processing Systems*, 2013.

M. C. Hughes, D. I. Kim, and E. B. Sudderth. Reliable and scalable variational inference for the hierarchical Dirichlet process. In *Artificial Intelligence and Statistics*, 2015a.

M. C. Hughes, W. T. Stephenson, and E. B. Sudderth. Scalable adaptation of state complexity for nonparametric hidden Markov models. In *Neural Information Processing Systems*, 2015b.

H. Ishwaran and M. Zarepour. Exact and approximate sum representations for the Dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283, 2002.

A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

S. Jain and R. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004.

T. Jebara. *Discriminative, generative and imitative learning*. PhD thesis, Massachusetts Institute of Technology, 2001.

W. H. Jefferys and J. O. Berger. Ockham's razor and Bayesian analysis. *American Scientist*, 80(1):64–72, 1992.

K. Jiang, B. Kulis, and M. I. Jordan. Small-variance asymptotics for exponential family Dirichlet process mixture models. In *Neural Information Processing Systems*, 2012.

M. J. Johnson and A. S. Willsky. Stochastic variational inference for Bayesian time series models. In *International Conference on Machine Learning*, 2014.

M. I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

B.-H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating parameters of hidden Markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(9):1639–1641, 1990.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI Conference on Artificial Intelligence*, 2006.

D. I. Kim, M. Hughes, and E. Sudderth. The nonparametric metadata dependent relational model. In *International Conference on Machine Learning*, 2012.

D. Kingma and M. Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

J. J. Kivinen, E. B. Sudderth, and M. I. Jordan. Learning multiscale representations of natural scenes using Dirichlet processes. In *International Conference on Computer Vision*, 2007.

A. Kucukelbir, R. Ranganath, A. Gelman, and D. M. Blei. Automatic variational inference in Stan. In *Neural Information Processing Systems*, 2015.

B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *International Conference on Machine Learning*, 2012.

K. Kurihara and M. Welling. Bayesian k-means as a "maximization-expectation" algorithm. *Neural computation*, 21(4):1145–1172, 2009.

K. Kurihara, M. Welling, and N. Vlassis. Accelerated variational Dirichlet process mixtures. In *Neural Information Processing Systems*, 2006.

K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational dirichlet process mixture models. *International Joint Conference on Artificial Intelligence*, 2007.

S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Neural Information Processing Systems*, 2009.

S. L. Lauritzen. *Graphical models*. Clarendon Press, 1996.

L.-J. Li, R. Socher, and F.-F. Li. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Empirical Methods in Natural Language Processing*, 2007.

S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

D. J. C. MacKay. Ensemble learning for hidden Markov models. Technical report, Department of Physics, University of Cambridge, 1997.

D. J. C. MacKay. Choice of basis for Laplace approximation. *Machine Learning*, 33(1), 1998.

V. Mansinghka, D. Selsam, and Y. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

V. Mansinghka, P. Shafto, E. Jonas, C. Petschulat, M. Gasner, and J. B. Tenenbaum. Crosscat: A fully bayesian nonparametric method for analyzing heterogeneous, high dimensional data. *arXiv preprint arXiv:1512.01272*, 2015.

J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *Neural Information Processing Systems*, 2008.

A. K. McCallum. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

D. Mimno and A. McCallum. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, 2008.

D. Mimno, M. Hoffman, and D. Blei. Sparse stochastic inference for latent Dirichlet allocation. In *International Conference on Machine Learning*, 2012.

D. Mimno, D. M. Blei, and B. E. Engelhardt. Posterior predictive checks to quantify lack-of-fit in admixture models of latent population structure. *Proceedings of the National Academy of Sciences*, 112(26), 2015.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 2014.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

D. R. Musser. Introspective sorting and selection algorithms. *Softw., Pract. Exper.*, 27(8):983–993, 1997.

R. M. Neal. Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods*, pages 197–211. Springer, 1992.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

S.-K. Ng and G. J. McLachlan. Speeding up the EM algorithm for mixture model-based segmentation of magnetic resonance images. *Pattern Recognition*, 37(8):1573–1589, 2004.

NIST. Rich transcriptions database. *http://www.nist.gov/speech/tests/rt/*, 2007.

P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. Springer, 2010.

J. Paisley, C. Wang, and D. Blei. The discrete infinite logistic normal distribution for mixed-membership modeling. In *Artificial Intelligence and Statistics*, 2011.

G. Parisi. *Statistical Field Theory*. Addison-Wesley, 1988.

J. Pitman and J. Picard. *Combinatorial Stochastic Processes*. Combinatorial Stochastic Processes: École D'Été de Probabilités de Saint-Flour XXXII - 2002. Springer, 2006.

J. K. Pritchard, M. Stephens, N. A. Rosenberg, and P. Donnelly. Association mapping in structured populations. *The American Journal of Human Genetics*, 67(1):170–181, 2000.

J. M. Quintana and M. West. An analysis of international exchange rates using multivariate dlm's. *The Statistician*, pages 275–281, 1987.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

R. Ranganath, S. Gerrish, and D. M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

C. E. Rasmussen. The infinite gaussian mixture model. In *Neural Information Processing Systems*, 1999.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. In *Neural Information Processing Systems*, 2001.

D. Rosenbaum and Y. Weiss. The return of the gating network: Combining generative models and discriminative training in natural image priors. In *Neural Information Processing Systems*, 2015.

S. L. Scott. Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351, 2002.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

E. B. Sudderth. *Graphical Models for Visual Object Recognition and Tracking*. PhD thesis, Massachusetts Institute of Technology, 2006.

E. B. Sudderth and M. I. Jordan. Shared segmentation of natural scenes using dependent pitman-yor processes. In *Neural Information Processing Systems*, 2009.

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.

Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. In *Neural Information Processing Systems*, 2008.

L. Theis and M. D. Hoffman. A trust-region method for stochastic variational inference with applications to streaming data. In *International Conference on Machine Learning*, 2015.

N. Ueda and Z. Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(1):1223–1241, 2002.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

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.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

S. G. Walker. Sampling the dirichlet mixture model with slices. *Communications in Statistics—Simulation and Computation®*, 36(1):45–54, 2007.

C. Wang and D. Blei. Truncation-free online variational inference for Bayesian nonparametric models. In *Neural Information Processing Systems*, 2012a.

C. Wang and D. M. Blei. A split-merge MCMC algorithm for the hierarchical Dirichlet process. *arXiv preprint arXiv:1201.1657*, 2012b.

C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical Dirichlet process. In *Artificial Intelligence and Statistics*, 2011.

Y. Wang, P. Sabzmeydani, and G. Mori. Semi-latent dirichlet allocation: A hierarchical model for human action recognition. In *Human Motion–Understanding, Modeling, Capture and Animation*, pages 240–254. Springer, 2007.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

A. Zhang, J. Zhu, and B. Zhang. Max-margin infinite hidden markov models. In *International Conference on Machine Learning*, 2014.

C. Zhang, C. H. Ek, X. Gratal, F. T. Pokorny, and H. Kjellstrom. Supervised hierarchical Dirichlet processes with variational inference. In *ICCV Workshop on Inference for probabilistic graphical models*, 2013.

J. Zhu, A. Ahmed, and E. P. Xing. MedLDA: maximum margin supervised topic models. *The Journal of Machine Learning Research*, 13(1):2237–2278, 2012.

J. Zhu, N. Chen, H. Perkins, and B. Zhang. Gibbs max-margin topic models with fast sampling algorithms. In *International Conference on Machine Learning*, 2013.

D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*, 2011.

D. Zoran and Y. Weiss. Natural images, Gaussian mixtures and dead leaves. In *Neural Information Processing Systems*, 2012.