Small-Variance Asymptotics for Bayesian Models

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Ke Jiang, M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2017

Dissertation Committee:

Dr. Mikhail Belkin, Advisor
Dr. Brian Kulis, Advisor
Dr. Alan Ritter
Abstract

Bayesian models have been used extensively in various machine learning tasks, often resulting in improved prediction performance through the utilization of (layers of) latent variables when modeling the generative process of the observed data. Extending the parameter space from finite to infinite-dimensional, Bayesian nonparametric models can infer the model complexity directly from the data and thus also adapt with the amount of the observed data. This is especially appealing in the age of big data. However, such benefits come at a price: the parameter training and the prediction are notoriously difficult even for parametric models. Sampling and variational inference techniques are two standard methods for inference in Bayesian models, but for many problems, neither approach scales effectively to large-scale data. Currently, there is significant ongoing research trying to scale these methods using ideas from stochastic differential equations and stochastic optimization. A recent thread of research has considered small-variance asymptotics of latent-variable models as a way to capture the benefits of rich probabilistic models while also providing a framework for designing more scalable combinatorial optimization algorithms. Such models are often motivated by the well-known connection between mixtures of Gaussians and $K$-means: as the variances of the Gaussians tend to zero, the mixture of Gaussians model approaches $K$-means, both in terms of objectives and algorithms.
In this dissertation, we will study small-variance asymptotics of Bayesian models, yielding new formulations and algorithms which may provide more efficient solutions to various unsupervised learning problems. Firstly, we consider clustering problems: exploring small-variance asymptotics for exponential family Dirichlet process (DP) and hierarchical Dirichlet process (HDP) mixture models. Utilizing connections between exponential family distributions and Bregman divergences, we derive novel clustering algorithms from the asymptotic limit of the DP and HDP mixtures that features the scalability of existing hard clustering methods as well as the flexibility of Bayesian nonparametric models. Secondly, we consider sequential models: exploring the small-variance asymptotic analysis of the infinite hidden Markov models, yielding a combinatorial objective function for discrete-data sequence observations with a non-fixed number of states. This involves a $k$-means-like term along with penalties based on state transitions and the number of states. We also present a simple, scalable, and flexible algorithm to optimize it. Lastly, we consider the topic modeling problems, which have emerged as fundamental tools in unsupervised machine learning. We approach it via combinatorial optimization, and take a small-variance limit of the latent Dirichlet allocation model to derive a new objective function. We minimize this objective by using ideas from combinatorial optimization, obtaining a new, fast, and high-quality topic modeling algorithm. In particular, we show that our results are not only significantly better than traditional small-variance asymptotic based algorithms, but also truly competitive with popular probabilistic approaches.
Dedicated to my grandparents.
Acknowledgments

First, I would like to thank my advisor Prof. Brian Kulis for being a constant source of knowledge, encouragement, support, inspiration, motivation, and guidance over the past six years. I feel deeply grateful that he encourages me to explore the research and allows me to start, stop, and fail along the way. Without him, I can not image that I am able to finish all the things I have done in the past six years. Also special thanks for Prof. Mikhail Belkin for his support during my most difficult time. I would like to thank the members of my candidacy and dissertation committee, Prof. Mikhail Belkin, Prof. Eric Fosler-Lussier, Prof. Radu Herbei, Prof. Alan Ritter, and Prof. Deliang Wang, for their comments and encouragements.

Second, I would like to thank Professors Güzin Bayraksan, Chris Hans, Laura Kubatko, Jeremy Morris, Feng Qin, Ramteen Sioshansi, Neelam Soundarajan, Christopher Stewart, Kenneth Supowit, Vincent Vu, Deliang Wang, Yusu Wang for all the things you have taught me. I would also like to thank Dr. Jian Peng, Dr. Yuefan Deng, Dr. Roman Samulyak, Dr. Xiangmin Jiao, Dr. Hongren Wei, Dr. David Bergman and Xiaohang Qu, without whom I would not be able to be in this position in the first place. I would also like to thank Dr. Misha Bilenko and Xiang Sun for giving me opportunities to work on interesting projects during the summer.

I would also like to thank my friends that share life experiences in these years, Rong Shi, Dachuan Huang, Yue Qiao, Gang Li, Mingzhe Li, Xiaotong Liu, Man Cao,
Yuxuan Wang, Jitong Chen, Qichao Que. I am also grateful to the CSE department for the academic and technical support, to the staff and faculty members in the department.

Finally and most importantly, I would like to thank my parents for their unconditionally love and support.
Vita

2007 ........................................... B.S. Computational Mathematics, Wuhan University, China
2016 ........................................... M.S. Computer Science & Engineering, The Ohio State University
2012-2015 ................................. Graduate Research Associate, The Ohio State University
2016 to present ............................. Graduate Teaching Associate, The Ohio State University

Publications


Fields of Study

Major Field: Computer Science and Engineering
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Small-Variance Asymptotics</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Classical SVA Approaches</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 GMM v.s. Kmeans: A Folklore</td>
<td>7</td>
</tr>
<tr>
<td>2.2 SVA Analysis for Exponential Family Distributions</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1 Exponential Family</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 Asymptotic Analysis</td>
<td>17</td>
</tr>
<tr>
<td>3. SVA for Infinite Mixture Models</td>
<td>20</td>
</tr>
<tr>
<td>3.1 Dirichlet Process Mixtures</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1 Dirichlet Process Mixtures</td>
<td>22</td>
</tr>
<tr>
<td>3.1.2 Asymptotic Analysis for Gaussian-DPM</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Hard Clustering for Exponential Family DP Mixtures</td>
<td>27</td>
</tr>
<tr>
<td>3.2.1 Scaling the Variance</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2 Asymptotics Behavior of the Gibbs Sampler</td>
<td>29</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Examples of small-variance asymptotics applied to standard machine learning tasks. See text for details.</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Some popular Bregman divergences and their corresponding generating functions. Here, $Q$ is a positive-definite matrix, and $\mathbb{R}_{++}$ represents the positive real numbers.</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Bregman divergences of interest for some popular regular exponential family distributions. Here, $x$ is the sufficient statistic, $\sigma$ and $N$ are assumed to be constant (source: Table 2 in [Banerjee et al., 2005]).</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Some example topics with top 15 words inferred from the NIPS and NYTimes datasets using our hard multinomial HDP algorithm.</td>
<td>40</td>
</tr>
<tr>
<td>3.2 Comparisons of the OFL algorithm with our algorithm for the cluster assignment step. Here, $\mathcal{M}$ is the set of cluster means, and $\lambda$ is the cost to introduce a new cluster.</td>
<td>42</td>
</tr>
<tr>
<td>3.3 Results of the objective function value (3.24) with squared Euclidean distance from different methods, where the number in the parentheses is the returned number of clusters. All the methods are followed by several $K$-means updates to convergence. In DPmeans++ [Bachem et al., 2015], $s = 2$. “AP-means” is initialized with the representatives returned by the affinity propagation algorithm [Frey and Dueck, 2007] and damping factor 0.5 is used. “NA” is a failed case where the AP algorithm returns all the data points as representatives. “DP-convex” is the convex exemplar-based approach proposed in [Yen et al., 2015] and the results are directly taken from [Yen et al., 2015]. In case of randomness inside the algorithm, it is run for 1000 times where the minimum is reported, and $k$-means++ [Arthur and Vassilvitskii, 2007] initialization is utilized instead of random initialization where applicable.</td>
<td>47</td>
</tr>
</tbody>
</table>
3.4 Results of the mean number of clusters for the online facility location algorithm out of 1000 runs. The numbers in the parentheses are the standard deviation. .......................................................... 48

4.1 Mean absolute percentage (MAP) error of AsymIHMM, AsymHMM, and HMM for the prediction error of the S&P 500 index. ................................. 73

5.1 The resulting combinatorial topic modeling objective function values optimized with Basic, Word, and Word+Refine algorithms with $\lambda = 10$ (smaller the better). .......................................................... 98

5.2 Comparisons of topic reconstruction $\ell_1$ errors of different algorithms for the synthesized datasets with varying number of documents. Here, “KMeans-KL” is the K-means algorithm with KL-divergence and LDA moments are used for method-of-moments style algorithms, namely Spectral, JD, and TPM. Top two results are shown in bold. .............. 101

5.3 Comparisons of topic reconstruction errors of different algorithms for the synthesized datasets, where the number inside the parentheses is the standard deviation. Here, “same length” means all the generated documents have the same number of word tokens, while “varied length” means the generated documents have varied number of word tokens. The “same length” and “varied length” documents share the same mean length. Also, “LDA” is the latent Dirichlet allocation moments [Anandkumar et al., 2012], and “DICA” is the discrete independent component analysis cumulants [Podosinnikova et al., 2015]. .............. 105

5.4 Comparisons of topic reconstruction $\ell_1$ errors of different algorithms on the semi-synthetic documents generated from the Enron and NYTimes datasets. Here, LDA moments are used for method-of-moments style algorithms, namely Spectral, JD, and TPM. ................................. 106

5.5 The discrete and held-out word log-likelihood on new documents for Enron ($K = 100$ topics) and NYTimes ($K = 100$ topics) datasets with fixed $\alpha = 0.01$ and varying $\beta$ values using both Gibbs sampler and our proposed Word+Refine algorithm. See text for details. .............. 108
5.6 Some example topics with top 10 words learned from NYTimes dataset using both the Gibbs sampler and our proposed Word+Refine algorithm. The topics are aligned using the Hungarian algorithm [Kuhn, 1955].

110
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Illustration of clustering with multiple data sets. Here, we have three data sets and six global means. Each data set has some number of local clusters, and each local cluster is associated with some global mean (source: Figure 1 in [Kulis and Jordan, 2012]).</td>
<td>34</td>
</tr>
<tr>
<td>3.2 Example images from the ImageNet data (Persian cat and elephant categories). Each image is represented via a discrete visual-bag-of-words histogram. Clustering via an asymptotic multinomial DP mixture considerably outperforms the asymptotic Gaussian DP mixture; see text for details.</td>
<td>38</td>
</tr>
<tr>
<td>4.1 The hidden Markov model: ( {x_1, x_2, \ldots, x_T} ) are the observations, and ( {z_1, z_2, \ldots, z_T} ) are the hidden states. The Markov property states that ( P(z_t</td>
<td>z_1, z_2, \ldots, z_{t-1}) = P(z_t</td>
</tr>
<tr>
<td>4.2 Synthetic observation sequence of the first 150 time steps generated with 4 hidden states. Observations with the same hidden states are represented with the same color.</td>
<td>68</td>
</tr>
<tr>
<td>4.3 Comparisons of the training accuracy and time (log-scale) of our AsymIHMM algorithm and the Beam Sampler on the synthetic Gaussian hidden Markov model data for 3000 time steps.</td>
<td>70</td>
</tr>
<tr>
<td>4.4 Comparisons of the predicted values of the S&amp;P 500 index from 12/29/1999 to 07/30/2012 using the AsymHMM, AsymIHMM and HMM methods, along with the true index values for the same period; see text for details.</td>
<td>72</td>
</tr>
</tbody>
</table>
5.1 The intuitions behind latent Dirichlet allocation. We assume there are some number of “topics”, which are distributions over words and exist for the whole document collection. Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then for each word, choose a topic assignment and choose the word from the corresponding topic (source: Figure 1 in [Blei, 2012]).

5.2 “Factorization” view of the latent Dirichlet allocation model. The “normalized” document-word count matrix can be factorized as the product of two low-dimensional matrices: the topic matrix and the document-topic matrix with the non-negativity constraint.

5.3 The facility location problem. There are a pool of fixed clients and a set of fixed potential facility locations (grey dots and dark-green factory symbols). We try to find a subset of facilities which gives the smallest combined facility cost and distance cost. Here, we have 4 open facilities (dark-green factory symbol), and 6 unopened facilities (grey dots) (source: Discrete Optimization Group DISOPT, École polytechnique fédérale de Lausanne).

5.4 Examples of “mini-topics” from an article of the AP corpus. Word tokens with the same color belong to the same topic, thus form one “mini-topic” (source: Figure 8 in [Blei et al., 2003]).

5.5 Comparisons of the normalized mutual information (NMI) scores and adjusted rand index (ARand) for the topic assignments of word tokens with the Basic, Word, and Word+Refine algorithms for Synthetic A (top) and Synthetic B (bottom) datasets with 5000 documents (higher the value, better the accuracy). The results of CGS are shown in dashed line.

5.6 The evolution of topic reconstruction $\ell_1$ errors of Gibbs sampler with different initializations: “Random” means random initialization, and “lambda=6” means initializing with the assignment learned using Word+Refine algorithm with $\lambda = 6$ for only 3 iterations.

5.7 Comparisons of the densities of the topic reconstruction $\ell_1$ error on the semi-NYTimes documents for the TPM, Anchor, Word+Refine, and CGS algorithms.
5.8 The evolution of the held-out word log-likelihood of Gibbs sampler with different initializations for the Enron dataset: “Random” means random initialization, and “W+R Init” means initializing with the assignment learned using the Word+Refine algorithm for only 3 iterations.
Bayesian models provide rich structures to model many kinds of different data, and thus have been used extensively in various machine learning tasks. Extending the parameter space from finite to infinite-dimensional, Bayesian nonparametric models [Hjort et al., 2010] can infer the model complexity directly from the data and also adapt with the amount of the observed data. This is especially appealing in the age of big data. However, the benefits of Bayesian models come at a price: the parameter training and the prediction are notoriously difficult even for parametric models. Sampling and variational inference techniques are two standard methods for inference in Bayesian models, but for many problems, neither approach scales effectively to large-scale data. Currently, there is significant ongoing research trying to scale these methods using ideas from stochastic differential equations [Welling and Teh, 2011; Patterson and Teh, 2013] and stochastic optimization [Hoffman et al., 2013].

Non-probabilistic models, on the other hand, exhibit scalability when dealing with large-scale data sets and enjoy various relaxations to embrace the rapid development of the large-scale optimization algorithms. The $K$-means algorithm, one of the simplest non-probabilistic clustering algorithms, is still considered one of the top data
mining algorithms [Wu et al., 2008]. A recent thread of research has considered small-variance asymptotics [Kulis and Jordan, 2012; Broderick et al., 2013] of latent-variable models as a way to capture the benefits of rich graphical models while also providing a framework for designing more scalable combinatorial optimization algorithms. Such models are often motivated by the well-known connection between mixtures of Gaussians and $K$-means: as the variances of the Gaussians tend to zero, the mixture of Gaussians model approaches $K$-means, both in terms of objectives and algorithms. The use of such techniques to derive scalable algorithms from rich probabilistic models is still emerging, but provides a promising approach to develop scalable learning algorithms.

In this dissertation, we will study small-variance asymptotics (SVA) of Bayesian models, yielding new formulations and algorithms which may provide more efficient solutions to various unsupervised learning problems. In particular, we will consider infinite clustering, sequential models and topic modeling problems.

**SVA for Infinite Mixture Models.** We explore small-variance asymptotics for clustering, focusing on the Dirichlet process (DP) mixture. Existing work has considered asymptotics over the Gaussian DP mixture [Kulis and Jordan, 2012], leading to $K$-means-like algorithms that do not fix the number of clusters upfront. This approach, while an important first step, raises the question of whether we can perform similar asymptotics over distributions other than the Gaussian. We answer in the affirmative by showing how such asymptotics may be applied to the exponential family distributions for DP mixtures; such analysis opens the door to a new class of scalable clustering algorithms and utilizes connections between Bregman divergences and exponential families. We further extend our approach to hierarchical nonparametric
models (specifically, the hierarchical Dirichlet process (HDP) [Teh et al., 2006a]). One of the primary advantages of generalizing beyond the Gaussian case is that it opens the door to novel scalable algorithms for discrete-data problems. We also discuss several improvements of the algorithm based on smart initialization [Sra et al., 2008], affinity propagation [Frey and Dueck, 2007], and tree-based splitting strategy [Pelleg and Moore, 2000], which perform quite nicely.

**SVA for Infinite Hidden Markov Models.** We explore small-variance asymptotics for sequential models, focusing on the infinite hidden Markov models [Beal et al., 2002; Teh et al., 2006a]. Hidden Markov model (HMM) is one of the most widely used probabilistic models for discrete sequence data, with diverse applications including DNA sequence analysis, natural language processing and speech recognition [Bishop, 2006]. Infinite-state hidden Markov models are nonparametric Bayesian extensions of the finite-state HMMs where the HDP priors are used to allow for unspecified number of states. To develop scalable algorithms for sequential data, we apply small-variance asymptotics to the infinite HMM. For this nonparametric model we obtain an objective that effectively combines the asymptotics from the parametric HMM with the asymptotics for the HDP. We obtain a $K$-means-like objective with three penalties: one for state transitions, one for the number of reachable states out of each state, and one for the number of total states. The key aspect of our resulting formulation is that, unlike the standard sampler for the infinite HMM, dynamic programming can be used. In particular, we describe a simple algorithm that monotonically decreases the underlying objective function. We present promising results comparing our non-probabilistic algorithm to its probabilistic counterparts, on a number of real and synthetic data sets.
**Combinatorial Topic Models.** We explore small-variance asymptotics for the standard latent Dirichlet allocation (LDA) [Blei et al., 2003] topic models. We formulate a combinatorial topic model via asymptotics directly on the LDA model. But merely using SVA to obtain a combinatorial topic model is insufficient: we also need effective algorithms to optimize the model. A direct application of the popular greedy combinatorial procedures on the LDA-based SVA model fails to compete with the probabilistic approaches. The greedy local assignment step can succeed only under certain circumstances and often fails to minimize the objective function greatly under random initialization. This setback necessitates a new idea. We develop algorithms for optimizing this combinatorial model by using ideas from facility location and incremental refinement. Moreover, we show how our procedure can be implemented in linear complexity with the number of word tokens and the number of topics. These transform the SVA approach into a competitive topic modeling algorithm. We demonstrate that this new approach not only improves significantly over the traditional SVA algorithms, but also competes favorably with existing state-of-the-art topic modeling algorithms. Moreover, we show that the sampler’s mixing time improves substantially when initialized using our combinatorial method for just a few iterations. We also compare favorably against several theoretically-motivated algorithms [Anandkumar et al., 2012; Arora et al., 2013; Podosinnikova et al., 2015].

The remainder of this dissertation is organized as follows. Chapter 2 reviews the concept of small-variance asymptotics and its application on exponential family distributions. Chapter 3 describes our asymptotic analysis on the infinite mixture models, focusing on the exponential family DP and HDP mixtures. Chapter 4 presents our asymptotic analysis on sequential models, focusing on the infinite hidden Markov
model. Chapter 5 shows our work on combinatorial topic models and efficient algorithms using ideas from combinatorial optimization. Chapter 6 concludes this dissertation.
Chapter 2: Small-Variance Asymptotics

Inference in large-scale probabilistic models remains a challenge, particularly for modern “big data” problems. While graphical models are undisputedly important as a way to build rich probability distributions, existing sampling-based and variational inference techniques still leave some problems out of reach.

Small-variance asymptotics has recently emerged as a way to capture the benefits of rich graphical models while also providing a framework for designing more scalable combinatorial optimization algorithms. Such models are often motivated by the well-known connection between mixtures of Gaussians and $K$-means: as the covariances of the Gaussians tend to zero, the mixture of Gaussians model approaches $K$-means, both in terms of objectives and algorithms.

In this chapter, we introduce the concept of small-variance asymptotics (SVA) for probabilistic models, which will relax the probabilistic model into a non-probabilistic formulation that may benefit from scalable combinatorial optimization algorithms. This analysis often based on letting the variance of particular distributions in the model go to zero, thus “small-variance” asymptotics.

This chapter is organized as follows. In Section 2.1, we review the asymptotic relationship between several probabilistic models and their corresponding asymptotic
models, focusing on Gaussian distributions. In Section 2.2, we review the exponential family distributions and their asymptotic analysis.

## 2.1 Classical SVA Approaches

In this section, we highlight the relationship between several probabilistic models and their corresponding asymptotic models for some popular machine learning tasks.

### 2.1.1 GMM v.s. Kmeans: A Folklore

A *Gaussian mixture model* (GMM) [Bishop, 2006] is a probabilistic model that assumes the data points \( \{x_n\} \in \mathbb{R}^d \) are generated from a mixture of a finite number of Gaussian distributions, and is also one of the most used unsupervised learning techniques. Specially, the generative process of the GMM is as follows:

\[
\begin{align*}
  z_n &\sim \text{Categorical}(q), \\
  x_n &\sim \mathcal{N}(x \mid \mu_{z_n}, \Sigma_{z_n}),
\end{align*}
\]

where \( q \in \Delta^K, \Delta^d \) is the \( d \)-dimensional simplex, \( K \) is the number of mixture components, \( z_n \) is the mixture component indicator for \( x_n \), \( \mu_k \in \mathbb{R}^d \) is the mean of the Gaussian distribution, and \( \Sigma_k \) is the \( d \times d \) covariance matrix. The joint probability for \( N \) independent observations with their mixture associations is:

\[
p(X, Z \mid q, \mu, \Sigma) = \prod_{n=1}^{N} \prod_{k=1}^{K} q_{zk}^{z_{nk}} \mathcal{N}(x_n \mid \mu_k, \Sigma_k)^{z_{nk}},
\]

where \( X \) denotes all the observations, and \( Z \) denotes all the mixture component indicators. Here, we use the “one-hot” encoding for the mixture component indicator \( z \); that is, \( z_{nk} = 1 \) if \( x_n \sim \mathcal{N}(\mu_k, \Sigma_k) \), and \( z_{nj} = 0 \) for \( j \neq k \).
On the other hand, $K$-means [MacQueen, 1967] is a clustering algorithm which tries to partition the given data set into $K$ clusters that minimizes the total squared Euclidean distance of data points from their corresponding cluster means:

$$
\sum_{k=1}^{K} \sum_{\mathbf{x} \in \ell_k} \| \mathbf{x} - \mu_k \|^2, \quad (2.4)
$$

where $\ell_k$ is the set of data points that are partitioned into cluster $k$, and $\mu_k = \sum_{\mathbf{x} \in \ell_k} \mathbf{x} / |\ell_k|$. The objective (2.4) can also be equivalently written as

$$
\sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \| \mathbf{x}_n - \mu_k \|^2, \quad (2.5)
$$

where $z_{nk} = 1$ if $\mathbf{x}_n$ is assigned to cluster $k$, 0 otherwise. For given partitions $\{\ell_k\}$, the minimizer of (2.4) is given by the centroids in each partition which justifies the “mean” in the name.

Now, we consider the relationship between Gaussian mixture model (2.3) and the $K$-means objective (2.5). We will show in the following that the mixture of Gaussians model approaches $K$-means both in terms of objectives and algorithms as the covariances of the Gaussians tend to zero.

First, we show the asymptotics for GMM from model likelihood perspective, which yields the $K$-means objective (2.5). Consider the joint likelihood (2.3). Assuming the covariance matrices of all mixture components are $\sigma^2 I$ with $\sigma > 0$, the negative logarithm of the joint log-likelihood becomes

$$
-2\sigma^2 \log p(X, Z \mid q, \mu, \Sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} [\| \mathbf{x}_n - \mu_k \|^2 + d\sigma^2 \log(2\pi\sigma^2) - 2\sigma^2 \log q_k].
$$

8
We can see that $\sigma^2 \log(\sigma^2) \to 0$ as $\sigma^2 \to 0$ since $0 \log 0 = 0$, and $\sigma^2 \log q_k \to 0$ as $\sigma^2 \to 0$. Therefore, we have as $\sigma^2 \to 0$,

$$-2\sigma^2 \log p(X, Z \mid \mathbf{q}, \mathbf{\mu}, \Sigma) \xrightarrow{\sigma^2 \to 0} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \| \mathbf{x}_n - \mathbf{\mu}_k \|^2.$$  

Thus we see in the limit, maximizing the complete-data log-likelihood (2.3) is equivalent to minimizing the $K$-means objective (2.5).

Next, we show the asymptotics for the maximum likelihood estimation of the GMM model from algorithmic perspectives. Unlike the single-component Gaussian model, there are no closed-form solutions for the model parameters. We need to turn to iterative algorithms. Expectation-maximization (EM) algorithm [Dempster et al., 1977] is a classical method to find maximum likelihood estimates of parameters in models involving unobserved variables. It alternates between performing an expectation (E) step, which finds the expectation of the likelihood evaluated at the current parameter estimates, and a maximization (M) step, which updates the model parameters by maximizing the expected likelihood. Most of the time, we will work with log-likelihood for convenience. Specifically for GMM, the EM algorithm will alternate between the following two steps:

- **E step**: here, we evaluate the responsibility of mixture component $k$ for data point $\mathbf{x}_n$.

$$\gamma(z_{nk}) = \frac{q_k \mathcal{N}(\mathbf{x}_n \mid \mathbf{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} q_j \mathcal{N}(\mathbf{x}_n \mid \mathbf{\mu}_j, \Sigma_j)} = \frac{q_k \exp\{-\|\mathbf{x}_n - \mathbf{\mu}_k\|^2/2\sigma^2\}}{\sum_{j=1}^{K} q_j \exp\{-\|\mathbf{x}_n - \mathbf{\mu}_j\|^2/2\sigma^2\}} \xrightarrow{\sigma^2 \to 0} \begin{cases} 1, & \text{if } k = \text{argmin}_j \|\mathbf{x}_n - \mathbf{\mu}_j\|^2 \\ 0, & \text{otherwise} \end{cases}$$
Here, all the responsibilities will become binary as $\sigma^2 \to 0$. More specifically, all the $K$ values will be increasingly dominated by the smallest value of $\{\|\mathbf{x}_n - \mu_1\|^2, \ldots, \|\mathbf{x}_n - \mu_K\|^2\}$. As $\sigma^2 \to 0$, only the smallest of these values will receive a non-zero probability. That is, the data point $\mathbf{x}_n$ will be assigned to the nearest cluster in terms of the squared Euclidean distance.

- **M step**: here, we re-estimate the parameters:

\[
\mu_k^{(\text{new})} = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \to \frac{1}{\sum_{n=1}^{N} z_{nk}} \sum_{n=1}^{N} z_{nk} \mathbf{x}_n.
\]

We can see that the estimate of the mean parameter approaches the cluster centroid as $\sigma^2 \to 0$.

Thus, in the limit, we obtain a hard assignment of data points to clusters, just as in the classical iterative $K$-means algorithm [Lloyd, 1982; Forgy, 1965] to minimize (2.5).

Through small-variance asymptotics, we have established the relationship between the Gaussian mixture models and the $K$-means problems by “hardening” probabilistic models. Similar connections have been shown to exist in other machine learning domains, including probabilistic principal component analysis (PCA) [Roweis, 1998; Tipping and Bishop, 1999] with standard principal component analysis [Pearson, 1901], probabilistic canonical correlation analysis with standard canonical correlation analysis (CCA) [Hotelling, 1936], restricted Bayes optimal classifier [Tong and Koller, 2000] with support vector machine (SVM) [Cortes and Vapnik, 1995], and hidden variable conditional random field (CRF) [Samdani et al., 2014] with latent structural
<table>
<thead>
<tr>
<th>Probabilistic Model</th>
<th>Asymptotic Model</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Mixture</td>
<td>$k$-means</td>
<td>Clustering</td>
</tr>
<tr>
<td>Probabilistic PCA</td>
<td>PCA</td>
<td>Dimensionality Reduction</td>
</tr>
<tr>
<td>Probabilistic CCA</td>
<td>CCA</td>
<td>Multi-view Learning</td>
</tr>
<tr>
<td>Restricted Bayes Optimal Classifier</td>
<td>Support Vector Machine</td>
<td>Classification</td>
</tr>
<tr>
<td>Hidden Variable CRF</td>
<td>Latent Structural SVM</td>
<td>Structured Prediction</td>
</tr>
</tbody>
</table>

Table 2.1: Examples of small-variance asymptotics applied to standard machine learning tasks. See text for details.

SVM [Yu and Joachims, 2009] (see Table 2.1 for corresponding tasks). Unlike their probabilistic counterparts, asymptotic models tend to learn faster due to the hard assignment. They are most of the time very intuitive, and usually provide very good results. Moreover, they can also be used to initialize the probabilistic approaches.

### 2.2 SVA Analysis for Exponential Family Distributions

Gaussian distributions have very nice properties, but they are inadequate to model discrete data. In this section, we review the extension of the asymptotic analysis to the exponential family distributions, which include many widely-used distributions: Gaussian, Multinomial, Exponential, Poisson, Dirichlet, and Gamma distribution.

#### 2.2.1 Exponential Family

Consider the \textit{exponential family} with natural parameter $\theta = \{\theta_j\}_{j=1}^d \in \Theta \subseteq \mathbb{R}^d$; then its probability density function can be written as [Schervish, 1995]:

$$
    p(x \mid \theta) = \exp \left( \langle x, \theta \rangle - \psi(\theta) - h(x) \right),
$$

where $\psi(\theta) = \log \int \exp(\langle x, \theta \rangle - h(x))dx$ is the log-partition function. Here we assume for simplicity that $x$ is a minimal sufficient statistic for the natural parameter $\theta$. In
addition, if the parameter space $\Theta$ is open, we call $p$ is in the regular exponential family. We can compute many statistics from $\psi(\theta)$; in particular,

$$\text{expectation} = \nabla \psi(\theta), \quad \text{covariance} = \nabla^2 \psi(\theta).$$ (2.7)

**Conjugate Prior.** In a Bayesian setting, the natural parameter $\theta$ is also considered as a random variable. We will require a prior distribution over the natural parameter $\theta$ and the inference task is to estimate the posterior distribution of $\theta$ given the observed data $X$ and the corresponding prior $p(\theta)$:

$$p(\theta \mid X) \propto p(X \mid \theta)p(\theta).$$ (2.8)

A convenient property of the exponential family is that a conjugate prior distribution of $\theta$ exists; that is, the posterior distribution has the same form as the prior. In particular, given any specific distribution in the regular exponential family with minimal statistics $x$, the conjugate prior can be parametrized as:

$$p(\theta \mid \tau, \eta) = \exp \left( \langle \theta, \tau \rangle - \eta \psi(\theta) - m(\tau, \eta) \right).$$ (2.9)

Here, the $\psi(\cdot)$ function is the same as that of the likelihood function (2.6), and $m(\tau, \eta)$ is a base measure. Given a data point $x_i$, the posterior distribution of $\theta$ has the same form as the prior, with $\tau \rightarrow \tau + x_i$ and $\eta \rightarrow \eta + 1$.

**Bregman-divergence Representation.** Another important property of the regular exponential family is the equivalence-representation relationship with regular Bregman divergences [Bregman, 1967].

**Definition 2.1.** Let $\phi : S \rightarrow \mathbb{R}$ be a differentiable, strictly convex function defined on a convex set $S \subseteq \mathbb{R}^d$. The Bregman divergence for any pair of points $x, y \in S$ is
Table 2.2: Some popular Bregman divergences and their corresponding generating functions. Here, $Q$ is a positive-definite matrix, and $\mathbb{R}^+\!\!\!\!_+$ represents the positive real numbers.

<table>
<thead>
<tr>
<th>Divergence</th>
<th>$D_\phi(x, y)$</th>
<th>$\phi(x)$</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>squared Euclidean distance</td>
<td>$|x - y|^2$</td>
<td>$|x|^2$</td>
<td>$\mathbb{R}^d$</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>$\frac{1}{2}(x - y)^T Q(x - y)$</td>
<td>$\frac{1}{2}x^T Q x$</td>
<td>$\mathbb{R}^d$</td>
</tr>
<tr>
<td>Kullback-Leibler divergence</td>
<td>$\sum_j x_j \log\left(\frac{x_j}{y_j}\right)$</td>
<td>$\sum_j x_j \log x_j$</td>
<td>$d$-simplex</td>
</tr>
<tr>
<td>Itakura-Saito distance</td>
<td>$\sum_j \left(x_j y_j - \log(x_j y_j) - 1\right)$</td>
<td>$- \sum_j \log x_j$</td>
<td>$\mathbb{R}^+!!!!_+$</td>
</tr>
</tbody>
</table>

defined as:

$$D_\phi(x, y) = \phi(x) - \phi(y) - \langle x - y, \nabla \phi(y) \rangle,$$

(2.10)

where $\nabla \phi(y)$ is the differential of $\phi$ at $y$.

The Bregman divergences can be viewed as a generalized distortion measure. Some popular Bregman divergences and their corresponding generating functions are shown in Table 2.2.

An important result connecting Bregman divergences and exponential families was discussed in [Banerjee et al., 2005]:

**Theorem 2.2.** (Theorem 4 in [Banerjee et al., 2005]) Let $p_{(\psi, \theta)}$ be the probability density function of a regular exponential family distribution. Let $\phi$ be the conjugate function of $\psi$ such that $(\text{int}(\text{dom}(\phi)), \phi)$ is the Legendre dual of $(\Theta, \psi)$. Let $\theta \in \Theta$ be the natural parameter and $\mu \in \text{int}(\text{dom}(\phi))$ be the corresponding expectation parameter, Let $d_\phi$ be the Bregman divergence derived from $\phi$. Then $p_{(\psi, \theta)}$ can be uniquely expressed as

$$p_{\psi, \theta}(x) = \exp(-D_\phi(x, \mu))b_\phi(x), \quad \forall x \in \text{dom}(\phi),$$

(2.11)

where $b_\phi : \text{dom}(\phi) \to \mathbb{R}_+$ is a uniquely determined function.
<table>
<thead>
<tr>
<th>Distribution</th>
<th>$p(x; \theta)$</th>
<th>$\mu$</th>
<th>$\phi(\mu)$</th>
<th>$D_\phi(x, \mu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-d Gaussian</td>
<td>$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-a)^2}{2\sigma^2}\right)$</td>
<td>$a$</td>
<td>$\frac{1}{2\sigma^2} \mu^2$</td>
<td>$\frac{1}{2\sigma^2} (x - \mu)^2$</td>
</tr>
<tr>
<td>1-d Poisson</td>
<td>$\frac{\lambda^x}{x!} \exp(-\lambda)$</td>
<td>$\lambda$</td>
<td>$\mu \log \mu - \mu$</td>
<td>$x \log(\frac{x}{\mu}) - (x - \mu)$</td>
</tr>
<tr>
<td>1-d Bernoulli</td>
<td>$q^x (1-q)^{1-x}$</td>
<td>$q$</td>
<td>$\mu \log \mu + (1-\mu) \log(1-\mu)$</td>
<td>$x \log(\frac{x}{\mu}) + (1-x) \log(\frac{1-x}{1-\mu})$</td>
</tr>
<tr>
<td>1-d Binomial</td>
<td>$\frac{N!}{x!(N-x)!} q^x (1-q)^{N-x}$</td>
<td>$Nq$</td>
<td>$\mu \log(\frac{Nq}{N}) + (N-\mu) \log(\frac{N-N\mu}{N-N})$</td>
<td>$x \log(\frac{x}{\mu}) + (N-x) \log(\frac{N-x}{N-N})$</td>
</tr>
<tr>
<td>1-d Exponential</td>
<td>$\lambda \exp(-\lambda x)$</td>
<td>$1/\lambda$</td>
<td>$-\log \mu - 1$</td>
<td>$\frac{x}{\mu} - \log(\frac{x}{\mu}) - 1$</td>
</tr>
<tr>
<td>m-d Sph. Gaussian</td>
<td>$\frac{1}{\sqrt{(2\pi\sigma^2)^m}} \exp\left(-\frac{|x-a|^2}{2\sigma^2}\right)$</td>
<td>$a$</td>
<td>$\frac{1}{2\sigma^2} |\mu|^2$</td>
<td>$\frac{1}{2\sigma^2} |x - \mu|^2$</td>
</tr>
<tr>
<td>m-d Multinomial</td>
<td>$\prod_{i=1}^N x_i \prod_{j=1}^m q_j^{x_j}$</td>
<td>$[N q_j]_{j=1}^{m-1}$</td>
<td>$\sum_{j=1}^m \mu_j \log(\frac{\mu_j}{N})$</td>
<td>$\sum_{j=1}^m x_j \log(\frac{x_j}{\mu_j})$</td>
</tr>
</tbody>
</table>

Table 2.3: Bregman divergences of interest for some popular regular exponential family distributions. Here, $x$ is the sufficient statistic, $\sigma$ and $N$ are assumed to be constant (source: Table 2 in [Banerjee et al., 2005]).
Table 2.3 shows the Bregman divergences of interest for some popular regular exponential family distributions. A key consequence of this result is that we can equivalently parameterize both the likelihood \( p(x | \theta) \) and the prior \( p(\theta | \tau, \eta) \) in terms of the expectation parameter \( \mu \):

\[
p(x | \theta) = p(x | \mu) = \exp(-D\phi(x, \mu)) f_\phi(x), \tag{2.12}
\]

\[
p(\theta | \tau, \eta) = p(\mu | \tau, \eta) = \exp \left( -\eta D\phi(\frac{\tau}{\eta}, \mu) \right) g_\phi(\tau, \eta), \tag{2.13}
\]

where \( \phi(\cdot) \) is the Legendre-conjugate function of \( \psi(\cdot) \) (denoted as \( \phi = \psi^* \)), \( f_\phi(x) = \exp(\phi(x) - h(x)) \), and \( \mu \) is the expectation parameter which satisfies \( \mu = \nabla \psi(\theta) \) (and also \( \mu = \theta^* \)). The Bregman divergence representation provides a natural way to parametrize the exponential family distributions with its expectation parameter (which is very similar to the standard Gaussian form) and, as we will see, it is quite convenient to work with this form. In the next few chapters, we will often work with the Multinomial distributions, and we now present its detail in the following example.

**Example 2.3.** (Example 10 in [Banerjee et al., 2005]) For a Multinomial distribution, we have

\[
p(x_1, \ldots, x_m | q) = \frac{N!}{\prod_{j=1}^m x_j} \prod_{j=1}^m q_j^{x_j},
\]

where \( x_j \in \mathbb{Z}_+, \sum_{j=1}^m x_j = N \), and \( q \in \Delta^m \). \( p(x | q) \) can be expressed as the density of an exponential family distribution in \( x = \{x_j\}_{j=1}^{m-1} \) with natural parameter \( \theta = \{\log(q_j/q_m)\}_{j=1}^{m-1} \) and log-partition function \( \psi(\theta) = -N \log q_m = N \log(1 + \sum_{j=1}^{m-1} e^{\theta_j}) \).

Thus, the expectation parameter \( \mu \) is given by

\[
\mu = \nabla \psi(\theta) = \left[ \frac{N e^{\theta_j}}{1 + \sum_{k=1}^{m-1} e^{\theta_k}} \right]_{j=1}^{m-1} = [N q_j]_{j=1}^{m-1},
\]

15
and the conjugate dual $\phi$ is

$$\phi(\mathbf{\mu}) = \sum_{j=1}^{m-1} Nq_j \log\left(\frac{q_j}{q_m}\right) + N \log q_m$$

$$= \sum_{j=1}^{m-1} Nq_j \log\left(\frac{q_j}{q_m}\right) + \sum_{j=1}^{m} Nq_m \log q_m$$

$$= \sum_{j=1}^{m} Nq_j \log q_j = \sum_{j=1}^{m} \left(\frac{\mu_j}{N}\right) \log(\frac{\mu_j}{N}).$$

And the corresponding Bregman divergence is just the KL-divergence:

$$D_{\phi}(\mathbf{x}, \mathbf{\mu}) = N \sum_{j=1}^{m} \frac{x_j}{N} \log\left(\frac{x_j/N}{\mu_j/N}\right),$$

with the $b_{\phi}(\mathbf{x}) = \prod_{j=1}^{m} \frac{x_j^N}{N^{x_j} \prod_{j=1}^{m} x_j^!}$ as in (2.11).

A special example is the Categorical distribution, which is just a Multinomial with $N = 1$. From the above, we know that it can be written as

$$p(\mathbf{x} | \mathbf{\mu}) = \exp\left(- \sum_{j=1}^{m} x_j \log\left(\frac{x_j}{\mu_j}\right)\right) = \exp\left(- \log\left(\frac{1}{q_k}\right)\right),$$

where $\mu_j = q_j$, $x_k = 1$, and $x_j = 0 \forall j \neq k$.

In order to perform the small-variance asymptotics, we define regular Bregman divergences using exponentially convex functions and show that there is a bijection between regular exponential families and regular Bregman divergences.

**Definition 2.4.** ([Banerjee et al., 2005]) Let $f : \Theta \to \mathbb{R}_{++}$ be a continuous exponentially convex function such that $\Theta$ is open and $\psi(\mathbf{\theta}) = \log(f(\mathbf{\theta}))$ is strictly convex. Let $\phi$ be the conjugate function of $\psi$. Then we say that the Bregman divergences $D_{\phi}$ derived from $\phi$ is a regular Bregman divergence.

Here, a function $f : \Theta \to \mathbb{R}_{++}, \Theta \subseteq \mathbb{R}^d$ is called exponentially convex if the kernel $K_f(\mathbf{\alpha}, \mathbf{\beta}) = f(\mathbf{\alpha} + \mathbf{\beta})$, with $\mathbf{\alpha} + \mathbf{\beta} \in \Theta$, is positive semi-definite.
Theorem 2.5. (Theorem 6 in [Banerjee et al., 2005]) There is a bijection between regular exponential families and regular Bregman divergences.

2.2.2 Asymptotic Analysis

In this section, we will discuss how to perform the small-variance asymptotics on exponential family distributions.

The Scaled Exponential Family. Given an exponential family distribution $p(x | \theta)$ with the natural parameter $\theta$ and the log-partition function $\psi(\theta)$, consider a scaled exponential family distribution:

\[
\tilde{\theta} = \beta \theta, \tag{2.14}
\]
\[
\tilde{\psi}(\tilde{\theta}) = \beta \psi(\tilde{\theta} / \beta), \tag{2.15}
\]

where $\tilde{\theta}$ is the new natural parameter, $\tilde{\psi}(\tilde{\theta})$ the new log-partition function, and $\beta > 0$.

The following result characterizes the relationship between the mean and covariance of the original and scaled exponential family distributions.

Lemma 2.6. Denote $\mu(\theta)$ as the mean, and $\text{cov}(\theta)$ as the covariance, of $p(x | \theta)$ with log-partition $\psi(\theta)$. Given a scaled exponential family with $\tilde{\theta} = \beta \theta$ and $\tilde{\psi}(\tilde{\theta}) = \beta \psi(\tilde{\theta} / \beta)$, the mean $\tilde{\mu}(\tilde{\theta})$ of the scaled distribution is $\mu(\theta)$ and the covariance, $\tilde{\text{cov}}(\tilde{\theta})$, is $\text{cov}(\theta) / \beta$.

Proof. This lemma follows directly from

\[
\tilde{\mu}(\tilde{\theta}) = \nabla_{\theta} \tilde{\psi}(\tilde{\theta}) = \beta \nabla_{\theta} \psi(\tilde{\theta} / \beta)
\]
\[
= \nabla_{\theta} \psi(\tilde{\theta} / \beta) = \nabla_{\theta} \psi(\theta)
\]
\[
= \mu(\theta),
\]
\[ \text{cov}(\tilde{\theta}) = \nabla^2_{\tilde{\theta}}(\tilde{\psi}(\tilde{\theta})) = \beta \nabla^2_{\theta}(\nabla_{\tilde{\theta}}\psi(\tilde{\theta}/\beta)) = \frac{1}{\beta} \times \nabla^2_{\tilde{\theta}}\tilde{\psi}(\tilde{\theta}/\beta) = \frac{1}{\beta} \times \nabla^2_{\theta}\psi(\theta) = \text{cov}(\theta)/\beta. \]

It is perhaps intuitively simpler to observe what happens to the distribution using the Bregman divergence representation. Recall that the generating function \( \phi \) for the Bregman divergence is given by the Legendre-conjugate of \( \psi \). Using standard properties of convex conjugates, we see that the conjugate of \( \tilde{\psi} \) is simply \( \tilde{\phi} = \beta \phi \).

The Bregman divergence representation for the scaled distribution is given by

\[
p(x | \tilde{\theta}) = p(x | \tilde{\mu}) = \exp(-D_{\tilde{\phi}}(\tilde{x}, \tilde{\mu}))f_{\tilde{\phi}}(\tilde{x}) = \exp(-\beta D_{\phi}(x, \mu))f_{\beta \phi}(x),
\]

where the last equality follows from Lemma 2.6 and the fact that, for a Bregman divergence, \( D_{\beta \phi}(\cdot, \cdot) = \beta D_{\phi}(\cdot, \cdot) \). Thus, as \( \beta \) increases under the above scaling, the mean is fixed while the distribution becomes increasingly concentrated around the mean (that is, \( 1/\beta \) serves as the same effect as the \( \sigma^2 \) in the Gaussian case). Therefore, this nice property of exponential family distributions provides us a nice tool to perform small-variance asymptotics. The direct scaling on the Bregman divergence has been studied in [Banerjee et al., 2005] for the EM algorithm for exponential family distributions in the non-Bayesian setting. The authors showed a connection between EM and a general \( K \)-means-like algorithm (similar to Section 2.1), where the squared
Euclidean distance is replaced by the Bregman divergence corresponding to the exponential family distribution of interest. But they did not show how to perform the scaling from the canonical parameters. In the following chapters, we will extend the analysis to the Bayesian models to derive similar connections.

Note, the model (2.16) is also called a reproductive exponential dispersion model [Jorgensen, 1997] if $\theta \in \mathbb{R}$. With this one-dimensional model, we can also approximate (2.16) in the limit as

$$
p(x \mid \tilde{\mu}) = \frac{\sqrt{\beta}}{\sqrt{2\pi v(x)}} \exp(-\beta D_{\phi}(x, \mu)) \quad \text{as} \quad \beta \to \infty,
\tag{2.17}
$$

where $v(x)$ is the variance function. This approximation is known as the saddle-point approximation [Daniels, 1954].
Chapter 3: SVA for Infinite Mixture Models

In this chapter, we explore small-variance asymptotics for clustering, focusing on the DP mixtures [Antoniak, 1974]. Existing work has considered asymptotics over the Gaussian DP-mixture [Kulis and Jordan, 2012], leading to $K$-means-like algorithms that do not fix the number of clusters upfront. This approach, while an important first step, raises the question of whether we can perform similar asymptotics over distributions other than the Gaussian. We answer in the affirmative by showing how such asymptotics may be applied to the exponential family distributions for DP mixtures; such analysis opens the door to a new class of scalable clustering algorithms and utilizes connections between Bregman divergences and exponential families described in Section 2.2.1. We further extend our approach to hierarchical nonparametric models (specifically, the hierarchical Dirichlet process (HDP) [Teh et al., 2006a]).

One of the primary advantages of generalizing beyond the Gaussian case is that it opens the door to novel scalable algorithms for discrete-data problems. For instance, visual bag-of-words [Li and Perona, 2005] has become a standard representation for images in a variety of computer vision tasks, but many existing probabilistic models in vision cannot scale to the size of data sets now commonly available. Similarly, text document analysis models (e.g., LDA [Blei et al., 2003]) are almost exclusively on discrete-data problems. Our analysis covers such problems; for instance, a particular
special case of our analysis is a hard version of the HDP topic models. We demonstrate the utility of our methods by exploring applications in text and vision.

In the non-Bayesian setting, asymptotics for the expectation-maximization algorithm for exponential family distributions were studied in [Banerjee et al., 2005]. The authors showed a connection between EM and a general $K$-means-like algorithm, where the squared Euclidean distance is replaced by the Bregman divergence corresponding to exponential family distribution of interest. Our results may be viewed as generalizing this approach to the Bayesian nonparametric setting. As discussed above, our results may also be viewed as generalizing the approach of [Kulis and Jordan, 2012], where the asymptotics were performed for the DP mixture with a Gaussian likelihood and conjugate priors, leading to a $K$-means-like algorithm where the number of clusters is not fixed upfront. Note that our setting is considerably more involved than either of these previous works, particularly since we will require an appropriate technique for computing an asymptotic marginal likelihood. Other connections between hard clustering and probabilistic models were explored in [Kurihara and Welling, 2009], which proposes a “Bayesian $K$-means” algorithm by performing a maximization-expectation algorithm.

This chapter is organized as follows. In Section 3.1, we review the Dirichlet process mixture models and the asymptotic analysis on Gaussian-DPM. In Section 3.2, we present the small-variance asymptotic analysis of the Gibbs sampler for the exponential family DP mixtures. Experiment results are given in Section 3.3, where we highlight applications of our analysis to image clustering and topic modeling. We will also discuss relationships with existing work and improvements of the algorithm in Section 3.4.
3.1 Dirichlet Process Mixtures

In this section, we briefly review the Dirichlet process mixtures, and asymptotic analysis on Gaussian-DPM.

3.1.1 Dirichlet Process Mixtures

Definition 3.1. The Dirichlet distribution is a distribution over the probability simplex. If \( q = (q_1, \ldots, q_K) \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K) \), then

\[
p(q_1, \ldots, q_K) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K q_k^{\alpha_k-1},
\]

where \( \alpha_k > 0, q_k \in (0, 1) \), and \( q_K = 1 - \sum_{k=1}^{K-1} q_k \).

An important property of the Dirichlet distribution is that it is the conjugate prior distribution of the Multinomial distribution. Therefore, it is most of the time used as a prior distribution for the mixture indicator variables in the Bayesian mixture models [Bishop, 2006]. Now, consider the Bayesian Gaussian mixture model with \( K \) components:

\[
\pi \sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K),
\]

\[
\theta_k = (\mu_k, \Sigma_k) \sim \text{Normal-Inverse-Wishart}(\nu) = H,
\]

\[
z_i \sim \text{Categorical}(\pi),
\]

\[
x_i \sim \mathcal{N}(x|\theta_k),
\]

we can write the prior distribution of the mixture model as:

\[
G = \sum_{k=1}^K \pi_k \delta_{\theta_k}.
\]

Next, we consider the nonparametric extension of the Dirichlet distribution, which can be used as the prior for mixture models with unbounded number of components.
Definition 3.2. [Antoniak, 1974] Let $H$ be a probability measure on the measurable space $(\Omega, B)$ and $\alpha > 0$. The Dirichlet process $DP(\alpha, H)$ is the distribution on probability measures $G$ such that for any finite partition $(A_1, \ldots, A_m)$ of $\Omega$,

$$(G(A_1), \ldots, G(A_m)) \sim \text{Dirichlet}(\alpha H(A_1), \ldots, \alpha H(A_m)).$$

The Dirichlet process (DP) is a Bayesian nonparametric [Hjort et al., 2010] prior on discrete measures. Due to its discreteness, these measures can be used to introduce clustering over data points where we group all data points assigned to the same $\theta \in H$ together. The Dirichlet process can be seen as an infinite dimensional generalization of the Dirichlet distribution with $K$ components and parameter $\alpha/K$ when the $K \to \infty$. Similarly, it can be written as (stick-breaking representation [Sethuraman, 1994]):

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k},$$

(3.3)

where $\pi_k = \beta_k \prod_{i=1}^{k-1} (1 - \beta_i)$, $\beta_i \sim \text{Beta}(1, \alpha)$, and $\theta_k \sim H$ with the property that a.s. $\sum_{i=1}^{\infty} \beta_i = 1$. Thus, using a DP, the complexity of the model can grow as we get more data since we place a prior on an unbounded number of parameters.

Now consider sampling from the Dirichlet process:

$$G \sim DP(\alpha, H), \quad X_n | G \sim G \text{ for } n = 1, \ldots, N.$$

If we marginalize out $G$, the conditional distribution follows:

$$X_n | X_1, \ldots, X_{n-1} = \begin{cases} 
  x_i, & \text{with probability } \frac{1}{n-1+\alpha}, \\
  \text{new draw from } H, & \text{with probability } \frac{\alpha}{n-1+\alpha},
\end{cases}$$

(3.4)
where \( \{x_1, \ldots, x_{N-1}\} \) are the instantiated values from the first \( N - 1 \) draws. Thus, the joint probability is

\[
p(X_1, \ldots, X_N) = p(X_1)p(X_2|X_1)\cdots p(X_N|X_1, \ldots, X_{N-1})
\]

\[
= \frac{\alpha^K \prod_{k=1}^K (n_k - 1)!}{\alpha(1 + \alpha) \cdots (N - 1 + \alpha)} \prod_{k=1}^K H(x_k),
\]

where \( n_k \) is the number of data points in cluster \( k \). This distribution is exchangeable, as the probability of the allocation of points to specific classes does not change when the points are permuted. The process to generate the partition is usually called the Chinese restaurant process (CRP) [Pitman, 2006].

We now look at the Dirichlet process mixtures:

\[
G \sim DP(\alpha, H),
\]

\[
G = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k},
\]

\[
\theta_i \sim G,
\]

\[
x_i \sim p(x|\theta_i).
\]

Markov chain Monte Carlo techniques are the standard techniques for inference on models that use the Dirichlet process. The simple Gibbs sampler [Neal, 2000] utilizing the CRP assignment metaphor is the most popular one. The detailed Gibbs updates are:

\[
P(z_i = c | z_{-i}, x_i, \mu) = \frac{n_{-i,c}}{Z(n - 1 + \alpha)} p(x_i | \mu_c)
\]

\[
P(z_i = c_{\text{new}} | z_{-i}, x_i, \mu) = \frac{\alpha}{Z(n - 1 + \alpha)} \int p(x_i | \mu) dG,
\]

where \( Z \) is the normalizing constant, \( n_{-i,c} \) is the number of data points (excluding \( x_i \)) that are currently assigned to cluster \( c \), and \( G \) is a prior over \( \mu \). If we choose to
start a new cluster during the Gibbs update, we sample its mean from the posterior distribution obtained from the prior distribution $G$ and the single observation $x_i$. After performing a whole set of Gibbs moves on the cluster indicators, we update the cluster means $\mu_c$ by sampling from the posterior of $\mu_c$ given the data points assigned to cluster $c$. Since the above Gibbs assignment steps are highly coupled, it may take a long time to converge to the equilibrium distribution. Augmented-variable Gibbs sampling methods such as slice sampling [Walker, 2007] have also been considered for DPM models.

Variational inference [Wainwright and Jordan, 2008] is another popular inference method for DP base models, where mean-field variational distribution is used as an approximate posterior distribution based on the stick-breaking construction (3.3). Since there are potentially infinite number of sticks, a truncation [Blei and Jordan, 2004] must be performed in order to do the optimization. It has been extended to variational inference methods without truncation [Wang and Blei, 2012], but sampling methods are involved to avoid the truncation.

### 3.1.2 Asymptotic Analysis for Gaussian-DPM

In this section, we briefly review the asymptotic analysis of the Gibbs sampler (3.6) and (3.7) of the Gaussian DP mixture model [Kulis and Jordan, 2012], where the likelihood is Gaussian and conjugate prior is used.

As in the non-Bayesian parametric clustering case described in Section 2.1.1, we also assume the covariance matrices of all mixture components are $\sigma^2 I$. The prior distribution for $\{\mu_k\}$ is $\mathcal{N}(0, \rho^2 I)$. Thus, the predictive likelihood in (3.7) is

$$
\alpha \int p(x|\mu) dG = \alpha [2\pi(\sigma^2 + \rho^2)]^{-d/2} \exp\{-||x||^2/(2\sigma^2 + \rho^2)\}.
$$

(3.8)
Now, letting $\alpha = (1 + \rho^2/\sigma^2)^{d/2} \exp[-\lambda/(2\sigma^2)]$, $\lambda > 0$ and $\sigma^2 \to 0$,

- First, consider the assignment sampling step: sampling $z_i$

$$z_i | Z_{-i}, \mu, X \propto \sum_{k=1}^{K} n_{-i,k} \exp[-\frac{\|x_i - \mu_k\|^2}{2\sigma^2}] \mathbb{I}[z_{ik} = 1] + \exp\{-\frac{1}{2\sigma^2}[\lambda + \frac{\sigma^2}{\sigma^2 + \rho^2 \|x_i\|^2}]\} \mathbb{I}[z_{i,K+1} = 1]$$

$$z_{ik} \xrightarrow{\sigma^2 \to 0} \begin{cases} 1, & \text{if } (k = \text{argmin}_j \|x_i - \mu_j\|^2 \\
 & \text{and } \lambda \geq \text{min}_j \|x_i - \mu_j\|^2) \\
0, & \text{otherwise} \end{cases}$$

where $K$ is currently utilized number of mixture components. We can see that as $\sigma^2 \to 0$, the data point $x_i$ will be assigned to the nearest cluster with a divergence at most $\lambda$.

- Next, consider the model parameter sampling step: sampling $\mu_k$

$$\mu_k | Z, X \sim \mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k), \quad \sigma^2 \to 0 \sum_{i=1}^{N} z_{ik} x_i \xrightarrow{\sigma^2 \to 0} \frac{\sum_{i=1}^{N} z_{ik} x_i}{n_k}$$

where $n_k$ is the number of data points assigned to mixture component $k$, and

$$\tilde{\mu}_k = \frac{\sum_{i=1}^{N} z_{ik} x_i}{n_k[1 + \sigma^2/(n_k\rho^2)]}, \quad \tilde{\Sigma}_k = \frac{\sigma^2 \rho^2}{\sigma^2 + n_k \rho^2} I.$$ 

We can see that the mass of the posterior distribution becomes concentrated around the sample mean as $\sigma^2 \to 0$.

We can see that the resulting algorithm leads to a $K$-means-like algorithm that do not fix the number of clusters upfront. The penalty parameter $\lambda$ serves as the control of creating new clusters, which has some similarity with the “self-similarity” parameter in the affinity propagation (AP) algorithm [Frey and Dueck, 2007] for the exemplar-based clustering problem. But unlike AP, it is directly derived from the probabilistic
model. The derived algorithm can be seen as minimizing the following DP-means [Kulis and Jordan, 2012] objective:

\[ \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \| x_n - \mu_k \|^2 + \lambda K. \]  

(3.9)

The objective function can also be directly derived from the asymptotics of the maximum a posterior (MAP) estimation of the Gaussian-DPM model [Broderick et al., 2013] using (3.5).

3.2 Hard Clustering for Exponential Family DP Mixtures

In this section, we explore the small-variance asymptotics for the exponential family DP mixtures. Our goal is to analyze what happens as we perform small-variance asymptotics when running the Gibbs sampler described in Section 3.1.1. When considering the large exponential family, we must face the following two challenges which are not the case for the Gaussian case [Kulis and Jordan, 2012]. On one hand, there is no obvious way to do the small-variance analysis. On the other hand, there is no closed-form expression for the predictive likelihood function. We will first show that the small-variance analysis can be performed with the scaled exponential family described in Section 2.2.2, and then apply the Laplace approximation for the predictive likelihood function to derive the hard clustering problem.

3.2.1 Scaling the Variance

We begin by considering how to accommodate the scaling of the exponential family likelihood with both the prior distribution and the predictive likelihood as they are essential for the analysis of the asymptotic behavior of the Gibbs sampler.
Given an exponential family distribution $p(x | \theta)$ with the natural parameter $\theta \in \mathbb{R}^d$ and the log-partition function $\psi(\theta)$, the likelihood is given by (2.12) and the prior is given by (2.13). Consider the scaled exponential family described in Section 2.2.2, whose natural parameter is $\tilde{\theta} = \beta \theta$ and log-partition function is $\tilde{\psi}(\tilde{\theta}) = \beta \psi(\theta/\beta)$, where $\beta > 0$. The Bregman divergence representation for the scaled likelihood is given by (2.16) and we repeat it here for convenience:

$$p(x | \tilde{\theta}) = p(x | \tilde{\mu}) = \exp(-\beta D_{\phi}(x, \mu))f_{\beta\phi}(x).$$

From Lemma 2.6, we know that the mean is fixed while the distribution becomes increasingly concentrated around the mean as $\beta$ goes to infinity (that is, $1/\beta$ serves as the same effect as the $\sigma^2$ in the Gaussian case), which is exactly what we want for small-variance analysis.

Next, we consider the conjugate prior distribution under the scaled exponential family as well. When scaling by $\beta$, we also need to scale the hyper-parameters $\tau$ and $\eta$, namely

$$\tau \rightarrow \tau/\beta, \quad \eta \rightarrow \eta/\beta. \tag{3.10}$$

This gives the following prior written using the Bregman divergence, where we are now explicitly conditioning on $\beta$:

$$p(\tilde{\theta} | \tau, \eta, \beta) = \exp\left(-\frac{\eta}{\beta} D_{\phi}(\frac{\tau}{\beta}, \mu)\right)g_{\phi}(\frac{\tau}{\beta}, \frac{\eta}{\beta}).$$

Finally, we compute the predictive likelihood for $x$ by integrating out $\tilde{\theta}$, as it will be necessary for the Gibbs sampler. Standard algebraic manipulations yield the
following:

\[ p(x | \tau, \eta, \beta) = \int p(x | \tilde{\theta}) \times p(\tilde{\theta} | \tau, \eta, \beta) d\tilde{\theta} \]

\[ = f_\phi(x) \times g_\phi(\tau, \eta, \beta) \times \beta^d \times \]

\[ \int \exp \left( - (\beta + \eta) D_\phi \left( \frac{\beta x + \tau}{\beta + \eta}, \mu(\theta) \right) \right) d\theta. \] (3.12)

Here, \( A_{(\phi, \tau, \eta, \beta)}(x) = \exp \left( - (\beta \phi(x) + \eta \phi(\frac{x}{\eta}) - (\beta + \eta) \phi(\frac{\beta x + \tau}{\beta + \eta})), \right) \), which arises when combining the Bregman divergences from the likelihood and the prior. Unlike the Gaussian case (3.8), there is no closed-form expression.

Now we make the following key insight, which will allow us to perform the necessary asymptotics. We can write the integral from the last line above (denoted as \( I \)) via Laplace’s method [Bishop, 2006]. Since \( D_\phi(\frac{\beta x + \tau}{\beta + \eta}, \mu) \) has a local minimum (which is global in this case) at \( \hat{\theta} = \hat{\mu}^* = (\frac{\beta x + \tau}{\beta + \eta})^* \), we have:

\[ I = \exp \left( - (\beta + \eta) D_\phi \left( \frac{\beta x + \tau}{\beta + \eta}, \hat{\mu} \right) \right) \times \]

\[ \left( \frac{2\pi}{\beta + \eta} \right)^{d/2} \left\| \frac{\partial^2 D_\phi(\frac{\beta x + \tau}{\beta + \eta}, \hat{\mu})}{\partial \theta \partial \theta^T} \right\|^{-1/2} + O\left( \frac{1}{\beta} \right) \]

\[ = \left( \frac{2\pi}{\beta + \eta} \right)^{d/2} \left\| \frac{\partial^2 D_\phi(\frac{\beta x + \tau}{\beta + \eta}, \hat{\mu})}{\partial \theta \partial \theta^T} \right\|^{-1/2} + O\left( \frac{1}{\beta} \right) \] (3.13)

where \( \frac{\partial^2 D_\phi(\frac{\beta x + \tau}{\beta + \eta}, \hat{\mu})}{\partial \theta \partial \theta^T} = \text{cov}(\hat{\theta}) \) is the covariance matrix of the likelihood function instantiated at \( \hat{\theta} \) and approaches \( \text{cov}(x^*) \) when \( \beta \) goes to \( \infty \). Note that the exponential term equals one since the divergence inside is 0.

### 3.2.2 Asymptotics Behavior of the Gibbs Sampler

We now have the tools to consider the asymptotic behavior of the Gibbs sampler for the exponential family DP mixture as we let \( \beta \to \infty \). The resulting will be a general
$K$-means-like hard clustering algorithm which utilizes the appropriate Bregman divergence in place of the squared Euclidean distance without a pre-specified number of clusters.

**Sampling the cluster assignments.** First, we consider the asymptotic behavior of the cluster assignment sampling step. Recall the conditional probabilities for performing Gibbs moves on the cluster indicators $z_i$, where we now are considering the scaled distributions:

$$P(z_i = c \mid z_{-i}, x_i, \beta, \mu) = \frac{n_{-i,c}}{Z} \exp(-\beta D_\phi(x_i, \mu_c)) f_\phi(x_i)$$  \hspace{1cm} (3.14)

$$P(z_i = c_{new} \mid z_{-i}, x_i, \beta, \mu) = \frac{\alpha}{Z} p(x_i \mid \tau, \eta, \beta),$$  \hspace{1cm} (3.15)

where $Z$ is the normalizing factor, and the marginal probability $p(x_i \mid \tau, \eta, \beta)$ is given by the derivations in (3.12) and (3.13). Now, we consider the asymptotic behavior of these probabilities as $\beta \to \infty$. Note that

$$\lim_{\beta \to \infty} \frac{\beta x_i + \tau}{\beta + \eta} = x_i, \hspace{1cm} (3.16)$$

$$\lim_{\beta \to \infty} A_{(\phi, \tau, \eta, \beta)}(x_i) = \exp(-\eta(\phi(\tau/\eta) - \phi(x_i))), \hspace{1cm} (3.17)$$

and that the Laplace approximation error term in (3.13) goes to zero as $\beta \to \infty$.

Further, we define $\alpha$ as a function of $\beta$, $\eta$, and $\tau$ (but independent of the data):

$$\alpha = \left(g_{\phi}(\tau, \eta) \beta^d \left(\frac{2\pi}{\beta + \eta}\right)^{d/2} \cdot \beta^d\right)^{-1} \cdot \exp(-\beta \lambda),$$  \hspace{1cm} (3.18)

for some $\lambda$. After canceling out the $f_\phi(x_i)$ terms from all probabilities, we can then write the Gibbs probabilities as

$$P(z_i = c \mid z_{-i}, x_i, \beta, \mu) = \frac{n_{-i,c} \cdot \exp(-\beta D_\phi(x_i, \mu_c))}{C_{x_i} \cdot \exp(-\beta \lambda) + \sum_{j=1}^{K} n_{-i,j} \cdot \exp(-\beta D_\phi(x_i, \mu_j))}$$  \hspace{1cm} (3.19)

$$P(z_i = c_{new} \mid z_{-i}, x_i, \beta, \mu) = \frac{C_{x_i} \cdot \exp(-\beta \lambda)}{C_{x_i} \cdot \exp(-\beta \lambda) + \sum_{j=1}^{K} n_{-i,j} \cdot \exp(-\beta D_\phi(x_i, \mu_j))},$$  \hspace{1cm} (3.20)

where $C_{x_i}$ approaches a positive, finite constant for a given $x_i$ as $\beta \to \infty$. 

30
Now, all of the above probabilities will become binary as $\beta \to \infty$. More specifically, all the $K + 1$ values will be increasingly dominated by the smallest value of

$$\{D_\phi(x_i, \mu_1), \ldots, D_\phi(x_i, \mu_K), \lambda\},$$

where $K$ is the current utilized number of clusters. As $\beta \to \infty$, only the smallest of these values will receive a non-zero probability:

$$z_{ic} \xrightarrow{\beta \to \infty} \begin{cases} 1, & \text{if } c = \arg\min_j D_\phi(x_i, \mu_j) \text{ & } \lambda \geq \min_j D_\phi(x_i, \mu_j), \\ 0, & \text{otherwise} \end{cases} \quad (3.21)$$

That is, the data point $x_i$ will be assigned to the nearest cluster with a divergence at most $\lambda$. If the closest mean has a divergence greater than $\lambda$, we start a new cluster containing only $x_i$.

**Sampling cluster parameters.** Second, we consider the asymptotic behavior of the cluster parameter sampling steps. Sampling $\mu_c$ from the posterior distribution is achieved by simply computing the empirical mean of a cluster in the limit as shown in the following. During Gibbs sampling, once we have performed one complete set of Gibbs moves on the cluster assignments, we need to sample the $\mu_c$ conditioned on all the assignments and observations. If we let $n_c$ be the number of data points assigned to cluster $c$, then the posterior distribution (parameterized by the expectation parameter) for cluster $c$ is

$$p(\mu_c | X, z, \tau, \eta, \beta) \propto p(X_c | \mu_c, \beta) \times p(\mu_c | \tau, \eta, \beta) \propto \exp \left( - (\beta n_c + \eta)D_\phi \left( \frac{\sum_{i=1}^{n_c} \beta x_i^c + \tau}{\beta n_c + \eta}, \mu \right) \right), \quad (3.22)$$

where $X$ is all the data, $X_c = \{x_1^c, \ldots, x_{n_c}^c\}$ is the set of points currently assigned to cluster $c$, and $z$ is the set of all current assignments. We can see that the mass of the
posterior distribution becomes concentrated around the sample mean as $\beta \rightarrow \infty$:

$$\mu_c \mid Z, X \xrightarrow{\beta \rightarrow \infty} \frac{\sum_{i=1}^{N} z_{ic} x_i}{n_c}.$$  \hspace{1cm} (3.23)

In other words, after we determine the assignments of data points to clusters, we update the means as the sample mean of the data points in each cluster. This is equivalent to the standard $K$-means cluster mean update step.

### 3.2.3 Objective Function and Algorithm

From the above asymptotic analysis of the Gibbs sampler, we observe a new algorithm which can be utilized for hard clustering. It is as simple as the popular $K$-means algorithm, but also provides the ability to adapt the number of clusters depending on the data as well as incorporate different distortion measures. The detailed description is in Algorithm 3.1. We iterate between the assignment and mean update steps until local convergence. Note that the initialization used here – placing all data points into a single cluster – is not necessary, but is one natural way to initialize the algorithm. And the global mean is the global minimizer for $K = 1$. Also note that the algorithm depends heavily on the choice of $\lambda$; heuristics for selecting $\lambda$ were briefly discussed for the Gaussian case in [Kulis and Jordan, 2012], and can be generalized in the obvious way to Bregman divergences.

We can easily show that the underlying objective function for our algorithm is quite similar to that in [Kulis and Jordan, 2012], replacing the squared Euclidean distance with an appropriate Bregman divergence. Recall that the squared Euclidean distance is the Bregman divergence corresponding to the Gaussian distribution. Thus, the objective function in [Kulis and Jordan, 2012] can be seen as a special case of our
Algorithm 3.1 Hard Clustering for EP-DPM

Input: Data: $\mathbf{x}_1, \ldots, \mathbf{x}_N$, Penalty: $\lambda > 0$.
Output: Clustering $\ell_1, \ldots, \ell_K$ and the number of clusters $K$.

1: Initialize $\mu_1 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$.
2: repeat
   //Update assignments:
3:   for every data point $\mathbf{x}_i$ do
4:      Compute the divergence $D_\phi(\mathbf{x}_i, \mu_c)$ for $c = 1, \ldots, K$.
5:      Set $z_i = c_0$ where $c_0 = \arg\min_c D_\phi(\mathbf{x}_i, \mu_c)$ and $\min_c D_\phi(\mathbf{x}_i, \mu_c) \leq \lambda$;
6:      otherwise, set $K = K + 1$, $z_i = K$, and $\mu_K = \mathbf{x}_i$.
7:   end for
   //Update means:
8:   for every cluster mean $\mu_j$ do
9:      Generate cluster $\ell_j$: $\ell_j = \{ \mathbf{x}_i | z_i = j \}$.
10:     Set $\mu_j = \frac{1}{|\ell_j|} \sum_{\mathbf{x} \in \ell_j} \mathbf{x}$.
11: end for
12: until no change in the assignments.

The objective function optimized by our derived algorithm is the following:

$$\min_{\{\ell_c\}_{c=1}^{K}} \sum_{c=1}^{K} \sum_{\mathbf{x} \in \ell_c} D_\phi(\mathbf{x}, \mu_c) + \lambda K$$

(3.24)

where $\{\ell_c\}$ are the resulting clusters, $K$ is the total number of clusters, $\phi$ is the conjugate function of the log-partition function of the chosen exponential family distribution, and $\mu_c$ is the sample mean of cluster $c$. The penalty term $\lambda$ controls the trade-offs between the likelihood and the model complexity, where a large $\lambda$ favors small model complexity (i.e., fewer clusters) while a small $\lambda$ favors large number of clusters. The penalty $\lambda$ here serves a similar effect as $\alpha$ in the probabilistic approach.

Given the above objective function, our algorithm can be shown to monotonically decrease the objective function value until convergence to some local minima.

We can also directly derive this objective function from the asymptotics of the maximum a posterior (MAP) estimation similar to the Gaussian case.
3.2.4 Extension to Hierarchies

A key benefit of the Bayesian approach is its natural ability to form hierarchical models. In the context of clustering, a hierarchical mixture allows one to cluster multiple groups of data – each group is clustered into a set of local clusters, but these local clusters are shared among the groups (i.e., set of local clusters across groups form global clusters, with a shared global mean). For Bayesian nonparametric mixture models, one way of achieving such hierarchies arises via the hierarchical Dirichlet process (HDP) [Teh et al., 2006a]:

\[ G_i | G_0 \sim DP(\alpha, G_0), \quad G_0 \sim DP(\gamma, H), \quad (3.25) \]

where \( G_0 \) is discrete with probability 1, and \( G\)'s are sharing the same atoms but with different weights. Thus the HDP provides a nonparametric approach to allow sharing of clusters among a set of DP mixtures.

In this section, we will discuss the extension of our analysis to the HDP mixtures, which yields a natural extension of our methods to groups of data. We assume that
there are $J$ data sets (groups) which we index by $j = 1, \ldots, J$. Data point $x_{ij}$ refers to data point $i$ from set $j$. The HDP model can be viewed as clustering each data set into local clusters, but where each local cluster is associated to a global mean (see Figure 3.1 for an illustration). Global means may be shared across data sets. When performing the asymptotics, we require variables for the global means ($\mu_1, \ldots, \mu_g$), the associations of data points to local clusters, $z_{ij}$, and the associations of local clusters to global means, $v_{jt}$, where $t$ indexes the local clusters for a data set. A standard Gibbs sampler considers updates on all of these variables, and in the nonparametric setting does not fix the number of local or global clusters beforehand.

The tools from the previous section may be nearly directly applied to the hierarchical case. As opposed to the flat model, the hard HDP requires two parameters: a value $\lambda_{\text{top}}$ which is utilized when starting a global (top-level) cluster, and a value $\lambda_{\text{bottom}}$ which is utilized when starting a local cluster. The resulting hard clustering algorithm first performs local assignment moves on the $z_{ij}$, then updates the local cluster assignments, and finally updates all global means. The detailed description is presented in Algorithm 3.2. A notable difference from the flat clustering algorithm is the “local move” part (step 20-28 in Algorithm 3.2), where we try to improve the objective value by creating a global cluster from a local cluster. This reduces to Algorithm 2 from [Kulis and Jordan, 2012] with the squared Euclidean distance replaced by an appropriate Bregman divergence. However, despite the similarity to the existing Gaussian case, we do view the extension to hierarchies as a promising application of our analysis. In particular, our approach opens the door to hard hierarchical algorithms over discrete data, such as text, and we briefly discuss an application of our derived algorithm to topic modeling.
Algorithm 3.2 Hard Clustering for EP-HDP

**Input:** Data: \( \{x_{ij}\} \), Local penalty: \( \lambda_{\text{bottom}} > 0 \), Global penalty: \( \lambda_{\text{top}} > 0 \).

**Output:** Global clustering \( \ell_1, \ldots, \ell_g \) and number of clusters \( K_j \) for each data set \( j \).

1. Initialize \( g = 1, K_j = 1 \) for all \( j \), and \( \mu_1 \) to be the global mean across all data sets.
2. Initialize local cluster indicators \( z_{ij} = 1 \) for all \( i \) and \( j \), and global cluster associations \( v_{j1} = 1 \) for all \( j \).
3. repeat
   //Update assignments:
   4: for every data point \( x_{ij} \) do
   5:   Compute the divergence \( d_{ijp} = D_{\phi}(x_{ij}, \mu_p) \) for \( p = 1, \ldots, g \).
   6:   if \( v_{jc} \neq p \) for all \( c = 1, \ldots, K_j \) then
   7:       set \( d_{ijp} = d_{ijp} + \lambda_{\text{bottom}} \).
   8:   end if
   9:   Let \( \hat{p} = \arg\min_p d_{ijp} \).
   10:  if \( d_{ij\hat{p}} > \lambda_{\text{bottom}} + \lambda_{\text{top}} \) then
   11:      set \( K_j = K_j + 1, z_{ij} = K_j, g = g + 1, v_{jK_j} = g \), and \( \mu_g = x_{ij} \).
   12:  else
   13:      if \( v_{jc} = \hat{p} \) for some \( c \) then
   14:         set \( z_{ij} = c, \) and \( v_{jc} = \hat{p} \).
   15:      else
   16:         set \( K_j = K_j + 1, z_{ij} = K_j, \) and \( v_{jK_j} = \hat{p} \).
   17:   end if
   18: end if
   19: end for
   //Update means:
   20: for every local cluster do
   21:   Let \( S_{jc} = \{x_{ij}|z_{ij} = c\} \), and \( \bar{\mu}_{jc} = \sum_{x \in S_{jc}} x/|S_{jc}| \).
   22:   Compute \( \bar{d}_{jcp} = \sum_{x \in S_{jc}} D_{\phi}(x, \mu_p) \) for \( p = 1, \ldots, g \).
   23:   if \( \min_p \bar{d}_{jcp} > \lambda_{\text{top}} + \sum_{x \in S_{jc}} D_{\phi}(x, \bar{\mu}_{jc}) \) then
   24:      set \( g = g + 1, v_{jc} = g \), and \( \mu_g = \bar{\mu}_{jc} \).
   25:   else
   26:      set \( v_{jc} = \arg\min_p \bar{d}_{jcp} \).
   27:   end if
   28: end for
4: until no change in the assignments.
The resulting objective function that is monotonically minimized by our algorithm is given as follows:

$$\min_{\{\ell_c\}_{c=1}^K} \sum_{c=1}^K \sum_{x_{ij} \in \ell_c} D_\phi(x_{ij}, \mu_c) + \lambda_{\text{bottom}} T + \lambda_{\text{top}} K, \quad (3.26)$$

where $K$ is the total number of global clusters and $T$ is the total number of local clusters. The bottom-level penalty term $\lambda_{\text{bottom}}$ controls both the number of local and top-level clusters, where larger $\lambda_{\text{bottom}}$ tends to give fewer local clusters and more top-level clusters. Meanwhile, the top-level penalty term $\lambda_{\text{top}}$, as in the one-level case, controls the trade-offs between the likelihood and model complexity.

### 3.3 Experiments

We now present a brief set of experiments highlighting applications of our analysis to discrete-data problems, namely image clustering and topic modeling. For all experiments, we randomly permute the data points at each iteration, as this tends to improve performance (unlike standard $K$-means, the order in which the data points are processed impacts the resulting clusters).

**Image Clustering.** We first explore an application of our techniques to image clustering, focusing on the ImageNet data [Deng et al., 2009]. We utilize a subset of this data for quantitative experiments, sampling 100 images from 10 different categories of this data set (Persian cat, African elephant, fire engine, motor scooter, wheelchair, park bench, cello, French horn, television, and goblet), for a total of 1000 images. Each image is processed via standard visual-bag-of-words: SIFT [Lowe, 2004] is densely applied on top of image patches in image, and the resulting SIFT vectors are quantized into 1000 visual words. We use the resulting histograms as our discrete
Figure 3.2: Example images from the ImageNet data (Persian cat and elephant categories). Each image is represented via a discrete visual-bag-of-words histogram. Clustering via an asymptotic multinomial DP mixture considerably outperforms the asymptotic Gaussian DP mixture; see text for details.

representation for an image, as is standard. Some example images from this data set are shown in Figure 3.2.

We explore whether the discrete version of our hard clustering algorithm based on a multinomial DP mixture outperforms the Gaussian mixture version (i.e., DP-means [Kulis and Jordan, 2012]); this will validate our generalization beyond the Gaussian setting. For both the Gaussian and multinomial cases, we utilize a farthest-first approach for both selecting \( \lambda \) as well as initializing the clusters (see [Kulis and Jordan, 2012] for a discussion of farthest-first for selecting \( \lambda \)).

We compute the normalized mutual information (NMI) between the true clusters and the results of the two algorithms on this difficult data set. The Gaussian version performs poorly, achieving an NMI of 0.06 on this data, whereas the hard multinomial version achieves a score of 0.27. While the multinomial version is far from perfect, it performs significantly better than DP-means. Scalability to large data sets is clearly feasible, given that the method scales linearly in the number of data points. Note that comparisons between the Gibbs sampler and the corresponding hard clustering algorithm for the Gaussian case were considered in [Kulis and Jordan, 2012], where
experiments on several data sets showed comparable clustering accuracy results between the sampler and the hard clustering method. Furthermore, for a fully Bayesian model that places a prior on the concentration parameter, the sampler was shown to be considerably slower than the corresponding hard clustering method. Given the similarity of the sampler for the Gaussian and multinomial case, we expect similar behavior with the multinomial Gibbs sampler.

**Hard Topic Models.** We also highlight an application to topic modeling, by providing some qualitative results over two common document collections. Utilizing our general algorithm for a hard version of the multinomial HDP is straightforward. In order to apply the hard hierarchical algorithm to topic modeling, we simply utilize the discrete KL-divergence (as shown in Example 2.3) in the hard exponential family HDP, since topic modeling [Teh *et al.*, 2006a] for text uses a categorical distribution for the data likelihood.

We apply our hard multinomial HDP algorithm on the NIPS 1-12\(^1\) and the NYTimes [Frank and Asuncion, 2010] datasets. For the NIPS dataset, we use the whole dataset, which contains 1,740 total documents, 13,649 words in the vocabulary, and 2,301,375 total words. For the NYTimes dataset, we randomly sampled 2,971 documents with 10,171 vocabulary words, and 853,451 words in total; we also eliminated low-frequency words (those with less than ten occurrences). The prevailing metric to measure the goodness of topic models is perplexity; however, this is based on the predictive probability, which has no counterpart in the hard clustering case. Furthermore, ground truth for topic models is difficult to obtain. This makes quantitative

\(^1\)http://www.cs.nyu.edu/~roweis/data.html
<table>
<thead>
<tr>
<th>NIPS</th>
<th>NYTimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 neurons, memory, patterns, activity,</td>
<td>team, game, season, play, games, point,</td>
</tr>
<tr>
<td>response, neuron, stimulus, firing,</td>
<td>player, coach, win, won, guy, played,</td>
</tr>
<tr>
<td>cortex, recurrent, pattern, spike,</td>
<td>playing, record, final</td>
</tr>
<tr>
<td>stimuli, delay, responses</td>
<td></td>
</tr>
<tr>
<td>2 neural, networks, state, weight,</td>
<td>percent, campaign, money, fund, quarter,</td>
</tr>
<tr>
<td>states, results, synaptic, threshold,</td>
<td>federal, public, pay, cost, according,</td>
</tr>
<tr>
<td>large, time, systems, activation, small,</td>
<td>income, half, term, program, increase</td>
</tr>
<tr>
<td>work, weights</td>
<td></td>
</tr>
<tr>
<td>3 training, hidden, recognition, layer,</td>
<td>president, power, government, country,</td>
</tr>
<tr>
<td>performance, probability, parameter,</td>
<td>peace, trial, public, reform, patriot,</td>
</tr>
<tr>
<td>error, speech, class, weights, trained,</td>
<td>economic, past, clear, interview, religious,</td>
</tr>
<tr>
<td>algorithm, approach, order</td>
<td>early</td>
</tr>
<tr>
<td>4 cells, visual, cell, orientation,</td>
<td>family, father, room, line, shares, recount,</td>
</tr>
<tr>
<td>cortical, connection, receptive, field,</td>
<td>told, mother, friend, speech, expression,</td>
</tr>
<tr>
<td>center, tuning, low, ocular, present,</td>
<td>won, offer, card, real</td>
</tr>
<tr>
<td>dominance, fields</td>
<td></td>
</tr>
<tr>
<td>5 energy, solution, methods, function,</td>
<td>company, companies, stock, market, business,</td>
</tr>
<tr>
<td>solutions, local, equations, minimum,</td>
<td>billion, firm, computer, analyst, industry,</td>
</tr>
<tr>
<td>hopfield, temperature, adaptation, term,</td>
<td>internet, chief, technology, customer,</td>
</tr>
<tr>
<td>optimization, computational, procedure</td>
<td>number</td>
</tr>
<tr>
<td>6 noise, classifier, classifiers, note,</td>
<td>right, human, decision, need, leadership,</td>
</tr>
<tr>
<td>margin, noisy, regularization,</td>
<td>foundation, number, question, country,</td>
</tr>
<tr>
<td>generalization, hypothesis, multiclass,</td>
<td>strike, set, called, support, law, train</td>
</tr>
<tr>
<td>cases, boosting, fig, pattern</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Some example topics with top 15 words inferred from the NIPS and NYTimes datasets using our hard multinomial HDP algorithm.
comparisons difficult for topic modeling, and so we therefore focus on qualitative results. Some sample topics (with the corresponding top 15 terms) discovered by our approach from both the NIPS and NYTimes datasets are given in Table 3.1; we can see that the topics appear to be quite reasonable. Also, we highlight the scalability of our approach: the number of iterations needed for convergence on these data sets ranges from 13 to 25. In contrast, for sampling methods, it is notoriously difficult to detect convergence, and generally a large number of iterations is required. Thus, we expect this approach to scale favorably to large data sets.

3.4 Discussions

In this section, we will discuss relationships with existing work and some improvements of the algorithm proposed in Section 3.2.3.

Consider the cluster assignment update step in Algorithm 3.1, where the means are fixed and we try to assign the data points to the means to decrease the objective function (3.24). If we consider this in an online scenario, there is a closely related problem, the online facility location (OFL) problem, where we would like to select a set $F$ of facilities to open in order to provide service to a set $U$ of demands. The goal is to minimize the total cost $\lambda |F| + \sum_{u \in U} \min_{f \in F} \text{dist}(u, f)$, where $\lambda$ is the cost to open a new facility. In the online setting, we must either open a facility at the point of demand or send this demand to some already-open facility as each demand arrives. In [Meyerson, 2001], the authors proposed a constant-factor approximation algorithm to the online problem, which can be seen as a probabilistic version of the assignment step in Algorithm 3.1 (see Table 3.2 for an comparison). We can see that the two algorithms perform the same if $d(x, \mathcal{M}) > \lambda$, but the OFL algorithm will
The OFL Algorithm

\[ d(x, M) = \min_{\mu \in M} d(x, \mu) \]

Add \( x \) to \( M \) with probability \( \min(1, \frac{d(x, M)}{\lambda}) \)

Our Algorithm

\[ d(x, M) = \min_{\mu \in M} d(x, \mu) \]

Add \( x \) to \( M \) if \( d(x, M) > \lambda \)

Table 3.2: Comparisons of the OFL algorithm with our algorithm for the cluster assignment step. Here, \( M \) is the set of cluster means, and \( \lambda \) is the cost to introduce a new cluster.

also introduce new clusters with non-zero probability if \( d(x, M) < \lambda \). Thus, the OFL algorithm usually produces many more number of clusters than needed. It has been shown that the OFL algorithm also provides a constant-factor approximation to the objective function (3.24) if the data points arrive in random order and the squared Euclidean distance is used [Pan et al., 2013]. However, we are not constrained to the online setting and introducing a new cluster can be dependent on multiple data points, thus this algorithm is far from optimal. More importantly, there is no cluster merge strategy once more-than-needed clusters have been introduced.

If we restrict the cluster representatives to be the data points (so-called exemplar-based clustering), there is an algorithm, Affinity Propagation (AP) [Frey and Dueck, 2007], which tries to solve a similar optimization problem. The objective function after having a clustering is:

\[
\max_{\mathcal{M}} \sum_{i \notin \mathcal{M}} s(x_i, x_{z_i}) + \sum_{i \in \mathcal{M}} s(x_i, x_i) = \min_{\mathcal{M}} \sum_{i \notin \mathcal{M}} d(x_i, x_{z_i}) + \lambda K,
\]

where \( \mathcal{M} \) is the set of cluster representatives, \( z_i \) is the cluster indicator for \( x_i \), \( K \) is the size of \( \mathcal{M} \), \( s(x, y) \) is the similarity function between \( x \) and \( y \), \( s(x, x) = -\lambda \) is the shared self-similarity, and \( d(x, y) = -s(x, y) \). We can see that the objective function is almost the same as (3.24) when all the data points share the same self-similarity.

Instead of the traditional EM algorithm for \( K \)-medoids problem, [Frey and Dueck,
2007] proposed a message-passing algorithm to optimize (3.27). It performs relatively well for small-scaled data sets, but the performance deteriorates as the number of data points increases. Besides, there is no guarantee that the algorithm will converge.

**Algorithm 3.3** Smart Initialization for (3.24)

**Input:** Set of data points $X$, Penalty: $\lambda$, Relaxation parameter $s > 1$.

**Output:** Set of selected cluster representative $M$ of cardinality $K'$.

1. Uniformly sample $\mu \in X$, and set $M = \{\mu\}$.
2. while $\sum_{x \in X} D_\phi(x, M) > 4s\lambda |M| \sigma_1 (\log_2 |M| + 2)$ do
3. Sample $\mu \in X$ with probability $p(\mu) = \frac{D_\phi(\mu, M)}{\sum_{x \in X} D_\phi(x, M)}$.
4. end while

We can derive a smart initialization algorithm for (3.24) with performance guarantee, which combines the recently proposed DP-means++ algorithm [Bachem et al., 2015] with the Bregman extension [Sra et al., 2008] of the $k$-means++ algorithm [Arthur and Vassilvitskii, 2007]. One necessary restriction in the Bregman divergence setting is that, we need to assume that the Bregman divergence being minimized has bounded curvature [Sra et al., 2008; Ackermann and Blömer, 2009]. Here, we follow the notations of [Sra et al., 2008] and assume there exist $\sigma_1, \sigma_2$ with $0 < \sigma_1 \leq \sigma_2 < \infty$, such that

$$\sigma_1 \|x - y\|^2 \leq D_\phi(x, y) \leq \sigma_2 \|x - y\|^2,$$

where the bounds only need to be satisfied in the convex hull of the input data points. It seems that without such assumptions on the curvature of $D_\phi$, one might not be able to obtain approximation ratios without dependence on the number of input data points [Chaudhuri and McGregor, 2008]. The algorithm is described in Algorithm 3.3, where $D_\phi(x, M) = \min_{\mu \in M} D_\phi(x, \mu)$. We have the following performance guarantee.
Theorem 3.3. Let $\lambda > 0$. Suppose the set $\mathcal{M}$ was sampled using Algorithm 3.3. Then with probability at least $1 - 1/s$, where $s > 1$, the set $\mathcal{M}$ is $O(\log K')$-competitive solution to the optimization problem (3.24) and $K' = |\mathcal{M}|$, i.e.

$$\text{cost}(X, \mathcal{M}) \leq 4s \frac{\sigma_2}{\sigma_1} (1 + \frac{\sigma_2}{\sigma_1}) \log_2 K' + 8s \frac{\sigma_2}{\sigma_1} (1 + \frac{\sigma_2}{\sigma_1}) + 2 \text{OPT}(X, \lambda),$$

(3.28)

where $\text{OPT}(X, \lambda)$ is the minimum value of (3.24).

Corollary 3.4. Let $\lambda > 0$. Denote by $K^*$ the optimal number of clusters for (3.24) and suppose $K'$ points are sampled using Algorithm 3.3 with parameter $\lambda$. Then

$$K^* \leq K' \left( 4s \frac{\sigma_2}{\sigma_1} (1 + \frac{\sigma_2}{\sigma_1}) \log_2 K' + 8s \frac{\sigma_2}{\sigma_1} (1 + \frac{\sigma_2}{\sigma_1}) + 1 \right).$$

(3.29)

The above results are derived using Markov’s inequality, which is known to be loose. In practice, it often stops early and select less number of clusters than needed.

In Algorithm 3.1, we start with the optimal solution for $K = 1$, and try to introduce new clusters by looking at each data point individually. This top-down splitting strategy has a crucial drawback that we only consider one data point for new clusters. Alternatively, we can consider a set of data points for introducing new clusters following the tree-structured $x$-means [Pelleg and Moore, 2000] framework. This should improve over Algorithm 3.1 since we split the existing clusters based on multiple data points rather than one.

The details of this tree-based splitting strategy are presented in Algorithm 3.4 and 3.5, where $D_\phi(x, \mathcal{L}_k) = D_\phi(x, \mu_k)$ and $\mu_k$ is the centroid of $\mathcal{L}_k$. We continuously split each existing cluster into two clusters to detect whether this would result in lower objective function value, and accept the split if it does decrease the value. We will run the Bregman $K$-means [Banerjee et al., 2005] in the end to further minimize the objective function if no more splits are accepted.
Algorithm 3.4 Tree-based Splitting Algorithm for (3.24) based on X-means

Input: Set of data points $X$, Penalty: $\lambda > 0$.
Output: The clustering $L$.
1: Initialize $K = 1$, $\mu_1 = \frac{1}{|X|} \sum_{x \in X} x$, canSplit = True.
2: while canSplit do
3: $L_{new} = \text{split}(X, L, \lambda)$.
4: if $|L_{new}| = |L|$ then
5: Set canSplit = False.
6: else
7: Run Bregman $K$-means algorithm on $X$ initialized with $L_{new}$, where $K = |L_{new}|$.
8: end if
9: end while

Algorithm 3.5 Split Function in Algorithm 3.4

function $\text{split}(X, L, \lambda)$
1: Initialize $L_{new} = \emptyset$.
2: for $k = 1 : |L|$ do
3: Set $X_k = \{x \mid x \in X, k = \arg\min_j D_\phi(x, L_j)\}$.
4: Set $Obj_k = \sum_{x \in X_k} D_\phi(x, L_k)$.
5: Run Bregman 2-means on $X_k$ to get two new clusters $L_{k,1}, L_{k,2}$, and get the new objective function value $Obj_{k,new}$.
6: if $Obj_k > Obj_{k,new} + \lambda$ then
7: Add $L_{k,1}, L_{k,2}$ to $L_{new}$.
8: else
9: Add $L_k$ to $L_{new}$.
10: end if
11: end for
12: return $L_{new}$.
In the following, we present some quantitative results comparing different methods discussed here for optimizing the objective function (3.24). For simplicity, we only consider the squared Euclidean distance. The results are conducted on the following four data sets from the UCI data repository [Frank and Asuncion, 2010]: Iris, Glass, Wine and Segment. We consider the following algorithms for comparison: DP-means++ [Bachem et al., 2015] which samples the cluster representatives from the data set using importance sampling similar to [Arthur and Vassilvitskii, 2007], Affinity Propagation (AP) which uses message passing algorithm to optimize (3.24), DP-convex [Yen et al., 2015] which takes an convex relaxation of the exemplar-based approach, DP-means [Kulis and Jordan, 2012] which is just our algorithm with squared Euclidean distance as the distortion measure, and x-DPmeans which is just the tree-based splitting algorithm proposed earlier.

Table 3.3 shows the resulting objective function values and number of clusters. We also include the brute-force algorithm for comparison: we solve the problem for each $K$ until the (3.24) stops to decrease. In practice, we can run multiple $K$-means for each $K$ and treat the minimum value as the “true” one. We can see that our algorithm fails to split the data set even with relatively large $\lambda$ value as in the Wine and Segment datasets. This demonstrates the inability of introducing new clusters by solely considering one data point. With the probabilistic smart initialization, it can introduce multiple clusters in the relative large $\lambda$ scenarios as in the Wine and Segment datasets, but it also delivers worse results than our simple algorithm in the Iris dataset. Thus, there is no guarantee that we can get enough number of clusters from this initialization. On the other hand, with the online facility location algorithm where we perform the algorithm on a single scan of a random permutation of the data
Table 3.3: Results of the objective function value \((3.24)\) with squared Euclidean distance from different methods, where the number in the parentheses is the returned number of clusters. All the methods are followed by several \(K\)-means updates to convergence. In DPmeans++ [Bachem et al., 2015], \(s = 2\). “AP-means” is initialized with the representatives returned by the affinity propagation algorithm [Frey and Dueck, 2007] and damping factor 0.5 is used. “NA” is a failed case where the AP algorithm returns all the data points as representatives. “DP-convex” is the convex exemplar-based approach proposed in [Yen et al., 2015] and the results are directly taken from [Yen et al., 2015]. In case of randomness inside the algorithm, it is run for 1000 times where the minimum is reported, and \(k\)-means++ [Arthur and Vassilvitskii, 2007] initialization is utilized instead of random initialization where applicable.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Iris ((\lambda = 2))</th>
<th>Glass ((\lambda = 9))</th>
<th>Wine ((\lambda = 20))</th>
<th>Segment ((\lambda = 600))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute-force</td>
<td>27.66 (K=6)</td>
<td>126.97 (K=6)</td>
<td>255.82 (K=3)</td>
<td>4547.64 (K=4)</td>
</tr>
<tr>
<td>DPmeans++</td>
<td>39.99 (K=3)</td>
<td>168.34 (K=3)</td>
<td>338.15 (K=2)</td>
<td>6499.55 (K=2)</td>
</tr>
<tr>
<td>AP-means</td>
<td>27.93 (K=7)</td>
<td>128.76 (K=7)</td>
<td>255.97 (K=3)</td>
<td>NA</td>
</tr>
<tr>
<td>DP-means</td>
<td>30.20 (K=4)</td>
<td>154.66 (K=2)</td>
<td>402.40 (K=1)</td>
<td>7898.98 (K=1)</td>
</tr>
<tr>
<td>DP-convex</td>
<td>27.97 (K=7)</td>
<td>128.13 (K=6)</td>
<td>263.79 (K=4)</td>
<td>4572.30 (K=4)</td>
</tr>
<tr>
<td>x-DPmeans</td>
<td><strong>27.66</strong> (K=6)</td>
<td><strong>126.97</strong> (K=6)</td>
<td><strong>255.82</strong> (K=3)</td>
<td><strong>4547.64</strong> (K=4)</td>
</tr>
</tbody>
</table>

set, we usually get many more number of clusters than needed (see Table 3.4 for an illustration). For AP-based initialization, we can see that it performs relatively well for small-scaled dataset, returning near-optimal solution within seconds. However, the performance deteriorates as the number of data points increases. Even for the Segment dataset which only has 2310 data points, it runs very slowly and fails catastrophically. The tree-based splitting algorithm 3.4 performs very well, often producing the optimal results and exceeding the sophisticated convex relaxation algorithm [Yen et al., 2015]. In terms of speed, it also runs very quickly, taking roughly \(\log_2(K^*)\) rounds of splitting. Even when the optimal \(K^*\) is very large, it only needs small number of rounds to achieve the desired number. We can further facilitate the computation by taking only a few \(K\)-means iterations in each round, 5-10 steps are usually sufficient.
<table>
<thead>
<tr>
<th>#clusters</th>
<th>Iris ($\lambda = 2$)</th>
<th>Glass ($\lambda = 9$)</th>
<th>Wine ($\lambda = 20$)</th>
<th>Segment ($\lambda = 200$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14.0 (1.9)</td>
<td>15.2 (2.2)</td>
<td>14.6 (2.9)</td>
<td>8.7 (1.7)</td>
</tr>
</tbody>
</table>

Table 3.4: Results of the mean number of clusters for the online facility location algorithm out of 1000 runs. The numbers in the parentheses are the standard deviation.

Also, once a cluster can not be split further, we do not need to consider it in the next round.

Ideally, there should be a merge step along with the split step, similar to the effective split-merge techniques [Dahl, 2003; Jain and Neal, 2004; Hughes et al., 2012] in the sampling methods. Due to the nice property of Bregman divergences, the differences between objective function values before and after merging two clusters can be computed in constant time. In order to apply the merge step, we need a good split strategy in the first hand. For example, there will be no effect of the merge step when applied along with our algorithm for the Wine dataset with $\lambda = 20$.

Unfortunately, the improvements discussed here can not be directly extended to the hierarchical cases due to the complications introduced by multiple layers of clusters. One possible solution would be a two-stage method: we first split the global clusters, then split each individual data set independently and synchronize with possible merge steps. We will show in Chapter 5 that a similar strategy works very nicely for a subclass of the hierarchical objective (3.26).

### 3.5 Conclusion

In this chapter, we have considered a general small-variance asymptotic analysis for the exponential family DP and HDP mixture models. Crucially, this analysis allows us
to move beyond the Gaussian distribution in such models, and opens the door to new clustering applications, such as those involving discrete data. Our analysis utilizes connections between Bregman divergences and exponential families, and results in a simple and scalable hard clustering algorithm which may be viewed as generalizing existing non-Bayesian Bregman hard clustering algorithms [Banerjee et al., 2005] as well as the DP-means algorithm [Kulis and Jordan, 2012].
Chapter 4: SVA for Infinite Hidden Markov Models

In this chapter, we explore small-variance asymptotics for sequential models, focusing on the infinite hidden Markov models [Beal et al., 2002; Teh et al., 2006a]. Hidden Markov models (HMM) are one of the most widely used probabilistic models for discrete sequence data, with diverse applications including DNA sequence analysis, natural language processing and speech recognition [Bishop, 2006]. The HMMs consist of a discrete hidden state sequence that evolves according to Markov assumptions, along with independent observations at each time step depending on the hidden state. The learning problem is to estimate the model given only the observation data. Infinite-state HMMs [Beal et al., 2002; Teh et al., 2006a] are nonparametric Bayesian extensions of the finite-state HMMs where the HDP priors are used to allow for an unspecified number of states. Exact inference in this model is intractable, so one typically resorts to sampling methods. The standard Gibbs sampling methods [Teh et al., 2006a] are notoriously slow to converge and cannot exploit the forward-backward structure of the HMMs. [Gael et al., 2008] presents a Beam sampling method which bypasses this obstacle via slice sampling [Neal, 2003; Walker, 2007], where only a finite number of hidden states are considered in each iteration. However, this approach is still computationally intensive since it works in the non-collapsed space. Thus infinite-state HMMs, while desirable from a modeling perspective, have been
limited by their inability to scale to large data sets – this is precisely the situation in which small-variance asymptotics has the potential to be beneficial.

To develop scalable algorithms for sequential data, we apply small-variance asymptotics to the infinite-state HMM. For this nonparametric model we obtain an objective that effectively combines the asymptotics from the parametric HMM with the asymptotics for the HDP. We obtain a $K$-means-like objective with three penalties: one for state transitions, one for the number of reachable states out of each state, and one for the number of total states. The key aspect of our resulting formulation is that, unlike the standard sampler for the infinite-state HMM, dynamic programming can be used. In particular, we describe a simple algorithm that monotonically decreases the underlying objective function. Finally, we present results comparing our nonprobabilistic algorithms to their probabilistic counterparts, on a number of real and synthetic data sets.

This chapter is organized as follows. In Section 4.1, we review the hidden Markov models and the infinite hidden Markov models using the HDP framework. In Section 4.2, we present the small-variance asymptotic analysis on the infinite Gaussian-HMMs. In Section 4.3, we propose a new algorithm to optimize the derived objective function based on approximate dynamic programming. Experiment results are given in Section 4.4, where we focus on comparison with probabilistic methods. Results are shown for both synthesized and S&P 500 Index data.

4.1 Infinite Hidden Markov Models

In this section, we briefly review the finite HMM, and the infinite HMM using the HDP framework.
**Hidden Markov Model.** HMM assumes a hidden state sequence \( Z = \{z_1, \ldots, z_T\} \) drawn from a finite discrete state space \( \{1, \ldots, K\} \), coupled with the observation sequence \( X = \{x_1, \ldots, x_T\} \) (see Figure 4.1 for an illustration). The resulting generative model defines a probability distribution over the hidden state sequence \( Z \) and the observation sequence \( X \).

Let \( \Pi \in \mathbb{R}^{K \times K} \) be the stationary transition probability matrix of the hidden state sequence with \( \Pi_i = \pi_i \in \triangle^K \) being a distribution over the latent states. Then the transition and observation probabilities are:

\[
p(z_t = j | z_{t-1} = i) = \Pi_{ij}, \quad (4.1)
\]

\[
p(x_t | z_t = j) = p(x_t | \theta_j), \quad (4.2)
\]

where \( \theta_j \) is the parameter of the observation probability. The initial state distribution is \( \pi_1 \in \triangle^K \). The Markov structure dictates that \( z_t \sim \text{Categorical}(\pi_{z_{t-1}}) \). The observation probability is assumed invariant, and the Markov structure induces conditional independence of the observations given the latent states.

In the standard parametric setting, inference of the model parameters (\( \Pi, \theta \)) is typically done using maximum likelihood estimation, using an EM-based algorithm [Rabiner, 1989] that applies a forward-backward pass through the entire sequence in the E-step to calculate the posterior marginal densities, which are then used in the M-step to optimize the parameters via maximization of the expected log-likelihood.

In a Bayesian setting, we can put priors on the parameters to perform the posterior analysis of the HMM:

\[
(\pi_{j1}, \ldots, \pi_{jK}) \sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K), \quad (4.3)
\]

\[
\theta \sim H. \quad (4.4)
\]
Figure 4.1: The hidden Markov model: \( \{x_1, x_2, \ldots, x_T\} \) are the observations, and \( \{z_1, z_2, \ldots, z_T\} \) are the hidden states. The Markov property states that \( P(z_t | z_1, z_2, \ldots, z_{t-1}) = P(z_t | z_{t-1}) \).

To extend the analysis to the infinite-state setting, one may easily consider letting \( K \to \infty \) to get the Dirichlet process HMM, similar to that of the Dirichlet process mixtures described in Section 3.1.1. But simply letting \( K \to \infty \) independently for each state will not work because they will not share the same state set. In order to perform the nonparametric analysis, we need to couple them together.

**Infinite Hidden Markov Model.** The infinite HMM (iHMM), also known as the HDP-HMM [Beal *et al.*, 2002; Teh *et al.*, 2006a] is a nonparametric Bayesian extension to the HMM, where an HDP prior is used to allow for an unspecified number of states. The HDP is a set of Dirichlet processes (DPs) with a shared base distribution, that are themselves drawn from a Dirichlet process [Teh *et al.*, 2006a]. Formally, we can write

\[
G_0 \sim \text{DP}(\gamma, H), \quad G_k \sim \text{DP}(\alpha, G_0), \tag{4.5}
\]

where \( H \) is the global base distribution, \( G_0 \) is discrete with probability 1, and \( G_k \)s are sharing the same atoms but with different weights. The parameters \( \alpha \) and \( \gamma \) are the concentration parameters for the \( G_k \) and \( G_0 \) measures, respectively. Thus the
HDP provides a nonparametric approach to allow sharing of states among a set of DPs. The *Chinese restaurant franchise* (CRF) [Teh et al., 2006a] process describes the partition generation of the HDP using the Chinese restaurant metaphor, similar to the CRP of the DP described in Section 3.1.1:

- There are different Chinese restaurants each with infinite number of tables.
- All the restaurants are sharing the same menu with infinite number of dishes.
- Each table in every restaurant can only serve one dish, and the first customer who sits at that table decides the dish.

Locally in each restaurant, it is just a CRP. The first customer sits at the first table, and samples a dish from the shared menu. The $n$th customer in restaurant $j$ sits at a table drawn from the following distribution:

\[
p(\text{customer } n \text{ sits at table } i \mid \text{existing seatings}) = \frac{n_{ji}}{\alpha + n - 1}, \quad (4.6) \\
p(\text{customer } n \text{ sits at a new table } \mid \text{existing seatings}) = \frac{\alpha}{\alpha + n - 1}, \quad (4.7)
\]

where $n_{ji}$ is the number of customers currently sitting at table $i$ in restaurant $j$.

Globally when sampling the menu, it is also a CRP. The first ever sampling request enjoys the first dish. When any customer sits at a new table, we go back to the shared menu:

\[
p(\text{customer samples dish } k \mid \text{dish } k \text{ has been sampled}) = \frac{m_k}{\gamma + m_{..}}, \quad (4.8) \\
p(\text{customer samples a new dish } \mid \text{existing sampled dishes}) = \frac{\gamma}{\gamma + m_{..}}, \quad (4.9)
\]

where $m_{jk}$ is the number of tables in restaurant $j$ serving dish $k$, $m_k$ is the total number of tables serving dish $k$, and $m_{..}$ is the total number of tables occupied.
To apply HDPs to sequential data, the iHMM can be formulated as follows:

\[
\begin{align*}
\beta & \sim \text{GEM}(\gamma), \\
\pi_k | \beta & \sim \text{DP}(\alpha, \beta), \\
\theta_k & \sim H, \\
\tau_t | \tau_{t-1} & \sim \text{Categorical}(\pi_{\tau_{t-1}}), \\
x_t & \sim p(x | \theta_{\tau_t}),
\end{align*}
\]  

(4.10)  
(4.11)  
(4.12)  
(4.13)  
(4.14)

where GEM is the Griffiths-Engen-McCloskey distribution [Pitman, 2006]. For a full Bayesian treatment, Gamma priors are placed on the concentration parameters (though we will not employ such priors in our asymptotic analysis). Exact inference in this model is intractable, so one typically resorts to sampling methods. The standard Gibbs sampling methods [Teh et al., 2006a] utilize the CRF process, where the sampling is highly-coupled. They are notoriously slow to converge. [Gael et al., 2008] presents a Beam sampling method which bypasses this obstacle via slice sampling [Neal, 2003; Walker, 2007], where only a finite number of hidden states are considered in each iteration. However, this approach is still computationally intensive since it works in the non-collapsed space.

4.2 Asymptotic Analysis of the Infinite HMM

In the following, we present the asymptotic treatment for the infinite HMM with Gaussian observation densities \( p(x_t | \tau_{tk} = 1) = \mathcal{N}(x_t | \mu_k, \sigma^2 I_d) \). Here \( \theta_{\tau_k} = \mu_k \), since we only consider the observation means. The extensions to exponential family observation probabilities can be done similarly as in Chapter 3. At a high level, the
connection we seek to establish can be proven in different ways. Here, we will primarily focus on examining small-variance asymptotics directly on the joint probability distribution, as done in [Broderick et al., 2013] for clustering and feature learning problems. Our ideas can be more clearly expressed by this technique, and it is independent of any inference algorithm. We will present a new algorithm to optimize our derived objective function in Section 4.3. By deriving an algorithm directly, we ensure that our method takes advantage of dynamic programming, unlike the standard sampler.

Our goal is to write down the full joint likelihood in the above model. Given the observation sequences $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$, for any possible partition of the $T$ observations into $K$ states, the generative model can be written as:

$$p(\mathcal{X}, \mathcal{Z}) = p(\mathcal{X} | \mathcal{Z}) p(\mathcal{Z}).$$

Specifically, $p(\mathcal{Z})$ is the HDP prior probability on the hidden state labelings. As discussed in Section 4.1, the hierarchical Dirichlet process yields assignments that follow the Chinese restaurant franchise [Teh et al., 2006a] process, and thus the infinite HMM additionally incorporates a term in the joint likelihood involving the prior probability of a set of state assignments arising from the CRF.

Suppose an assignment of observations to states has $K$ resulting states (i.e., number of restaurants in the franchise, see Section 4.1 for the detailed description of the metaphor), $s_i$ is the number of states that can be reached from state $i$ in one step (i.e., number of tables in each restaurant $i$), and $n_i$ is the number of observations in each state $i$ (i.e., number of customers in each restaurant). Then the probability of an assignment in the HDP can be written as (after integrating out mixture weights)
[Antoniak, 1974; Sudderth, 2012]:

\[ p(Z|\alpha, \gamma) \propto \gamma^{K-1} \frac{\Gamma(\gamma + 1)}{\Gamma(\gamma + \sum_{k=1}^{K} s_k)} \prod_{k=1}^{K} \alpha^{s_k - 1} \frac{\Gamma(\alpha + 1)}{\Gamma(\alpha + n_i)}, \]  

(4.15)

if we only consider terms that would not be constants after we do the asymptotic analysis [Broderick et al., 2013]. A nice thing about (4.15) is that it bypasses the infinite-state challenges, and provides a natural way to directly condition on the resulting number of states.

For the likelihood, we follow the simple assumption: the observation densities are Gaussians with a shared covariance matrix \( \sigma^2 I_d \). Further, the means are drawn independently from the prior \( \mathcal{N}(0, \rho^2 I_d) \), where \( \rho^2 > 0 \) (similar to the analysis in Section 3.1.2). Therefore, the joint probability is

\[
p(X, Z) \propto p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \times \prod_{t=1}^{T} \mathcal{N}(x_t|\mu_{z_t}, \sigma^2 I_d) \times \\
p(Z|\alpha, \gamma) \times p(\mu_{1:K}),
\]

(4.16)

where \( p(\mu_{1:K}) = \prod_{k=1}^{K} \mathcal{N}(\mu_k|0, \rho^2 I_d) \).

Now, we can perform the small-variance analysis on the generative model (4.16). In order to retain the impact of the hyperparameters \( \alpha \) and \( \gamma \) in the asymptotics, we can choose some constants \( \lambda_1 > 0, \lambda_2 > 0 \) such that

\[
\alpha = \exp(-\lambda_1 \beta), \quad \gamma = \exp(-\lambda_2 \beta),
\]

(4.17)

where \( \beta = 1/(2\sigma^2) \). Note that, in this way, we have \( \alpha \to 0 \) and \( \gamma \to 0 \) when \( \beta \to \infty \).
We now can consider the objective function for maximizing the generative probability as we let \( \beta \to \infty \). This gives
\[
p(\mathcal{X}, \mathcal{Z}) \propto \exp \left[ -\beta \left( \sum_{t=1}^{T} \| \mathbf{x}_t - \mu_{z_t} \|^2 - \lambda \sum_{t=2}^{T} \log \Pi_{z_{t-1},z_t} \right) - \lambda_1 \beta \sum_{k=1}^{K} (s_k - 1) - \lambda_2 \beta (K - 1) + \log(p(z_1)) \right]
\]
where \( \Pi \) is the transition probability matrix with \( \Pi_{i,j} = P(z_{t+1} = j | z_t = i) \). The second term comes from the asymptotics of the state transition probabilities using the Bregman divergence representation of the Categorical distribution (see Example 2.3 for details) and the scaling \( \lambda \beta \) with \( \lambda > 0 \) as described in Section 2.2.1. We are expressing the transition penalties here in terms of \( \Pi \) as opposed to the KL divergence. Here, we did not perform the asymptotics on the initial state distribution. We always assign the first observation to the first state.

Therefore, maximizing the joint probability is asymptotically equivalent to the following optimization problem:
\[
\min_{K,Z,\mu,\Pi} \sum_{t=1}^{T} \| \mathbf{x}_t - \mu_{z_t} \|^2 - \lambda \sum_{t=2}^{T} \log \Pi_{z_{t-1},z_t} + \lambda_1 \sum_{k=1}^{K} (s_k - 1) + \lambda_2 (K - 1). \quad (4.19)
\]
In words, this objective seeks to minimize a penalized \( K \)-means-like objective, with additional constraints. The first is the constraint based on the transitions from state to state. With a large \( \lambda \), it will favor most probable transition; with a small \( \lambda \), it will have little impact on the state association. The second penalty constrains the number of transitions out of each state, and the third penalty constrains the total number of states, which have similar effects as in Section 3.2.4. The “minus one” here is intuitive, since we need at least one state for the model and at least one outgoing transition for each state.
4.3 Algorithms

Algorithm 4.1 High-level Description of the Asymptotic iHMM Algorithm

Input: Data: $x_1, \ldots, x_T$, Penalties: $\lambda > 0, \lambda_1 > 0, \lambda_2 > 0$.
Output: number of states $K$, state association $Z$, state mean $\mu$, and state transitions $\Pi$.

1: Initialize with one hidden state. The parameters are therefore $K = 1, \mu_1 = \frac{1}{T} \sum_{i=1}^{T} x_i$, $\Pi = 1$.
2: repeat
3: Perform a forward-backward step (via approximate dynamic programming) to update the state association $Z$.
4: Update $K, \mu, \Pi$.
5: For each state $i, (i = 1, \ldots, K)$, check if the set of observations to any state $j$ that are reached by transitioning out of $i$ can form a new dedicated hidden state and lower the objective function in the process. If there are several such moves, choose the one with the maximum improvement in objective function.
6: Update $K, \mu, \Pi$.
7: until convergence

Now, we follow an alternating minimization framework to optimize (4.19). We first determine the sequence of states to optimize the objective when all but $Z$ is fixed using a forward-backward routine. Here, we cannot apply exact dynamic programming, due to the possible creation of new states as well as the change in transition probabilities when either creating a new state or transiting to a new state (paying a $\lambda_1$ penalty).

We then update the means of each state as the empirical means based on the state assignments, and the transition matrix as the empirical transition matrix. We further adopt a move analogous to that described for the hard HDP in Section 3.2.4. This step determines if the objective will decrease if we create a new hidden state in a certain fashion; in particular, for each existing state $j$, we compute the change in objective function value that occurs when observations that transition from state $j$
to some state $k$ are given their own new hidden state. A high-level description is in Algorithm 4.1.

### 4.3.1 Forward-Backward Step: Approximate Dynamic Programming

In the state association update step, we consider the forward-backward framework. We will compute a $K \times T$ matrix $\alpha$, where $\alpha(c, t)$ represents the minimum cost over paths of length $t$ from the beginning of the sequence that reach state $c$ at time step $t$. We use the term “cost” to refer to the sum of the distances of data points to state means, as well as the additive penalties incurred. Since we are interested in the possibility of potentially creating new states during this forward-backward process, and the creation of new states will necessarily change the transition probabilities, it does not appear that $\alpha$ can be computed exactly. We instead describe a procedure that computes an upper bound for each value of $\alpha$.

To give further intuition for why it is difficult to compute the exact value of $\alpha$: suppose we have computed the minimum cost of paths up to step $t - 1$ and we would like to compute the values of $\alpha$ for step $t$. The value of a path that ends in state $c$ is obtained by examining, for all states $i$, the cost of a path that ends at $i$ at step $t - 1$ and then transitions to state $c$ at step $t$. Thus, we must consider the transition from $i$ to $c$. If there are existing transitions from state $i$ to state $c$, then we proceed as in a standard forward-backward algorithm. However, we are also interested in two other cases:

- one where there are no existing transitions from $i$ to $c$ but we consider this transition along with a penalty $\lambda_1$, 


the other where an entirely new state is formed and we pay a penalty \( \lambda_2 \).

In the first case, the standard forward-backward routine faces an immediate problem, since when we try to compute the cost of the path given by \( \alpha(c, t) \), the cost will be infinite as there is a \(-\log(0)\) term from the transition probability. We must therefore alter the forward-backward routine, or there will never be new states created nor transitions to an existing state which previously had no transitions.

The main idea is to derive and use bounds on how much the transition matrix could change under the above scenarios. As long as we could show that the values we obtain for \( \alpha \) are upper bounds, then we can show that the objective function will decrease after the forward-backward routine, as the existing sequence of states is also a valid path (with no new incurred penalties). That is, the cost we compute here is an upper bound of both the non-penalized objective function value and newly-introduced penalties. If there are no new states created or no new transitions happened, the upper bound for the newly-introduced penalties is zero as mentioned before.

Now we describe the algorithm in more detail. In the following description, \( K_{old} \) is the number of states from the previous iteration, and \( K \) is the number of states before the computation of \( \alpha \) at time step \( t \). Then, for each state \( 1 \leq c \leq K \),

(I). If we are transitioning from a state \( i \leq K_{old} \),

- Standard situation: If \( c \leq K_{old} \) and \( \Pi_{i,c} \neq 0 \), we compute (as in the parametric case):

\[
d(i, c) = \| \mathbf{x}_t - \boldsymbol{\mu}_c \|^2 - \lambda \times \log(\Pi_{i,c}).
\]

(4.20)
• No existing transitions: Otherwise, we use the following upper bound when transitioning to state $c$:

$$d(i, c) = \|\mathbf{x}_t - \mu_c\|^2 - \lambda \times \log \left(\frac{1}{n_i}\right) + \lambda \times E_i + \lambda_1,$$  \hspace{1cm} (4.21)

where,

- $n_i$ is the total number of transitions out of state $i$.
- $E_i$ is the upper bound of the possible change in one transition probability from state $i$ incurred by adding one state to the reachable pool of state $i$. We compute the change of the maximum of $\frac{n_{ij}}{n_i}$, and we have (assume $j$ is the largest one)

$$E_i = (n_{ij} - 1) \times \left(\log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i}\right),$$  \hspace{1cm} (4.22)

- $\lambda_1$ is the penalty incurred from transiting to a new state.

(II). Entirely new state: Otherwise, we use the bound:

$$d(i, c) = \|\mathbf{x}_t - \mu_c\|^2 - \lambda \times \log \left(\frac{1}{N - 1}\right) + \lambda_1.$$  \hspace{1cm} (4.23)

where,

- $\frac{1}{N - 1}$ is an upper bound of the transition probability from state $i$ to state $c$,
- $0$ is the upper bound of possible change in the transition probabilities from state $k$ where the original transition occurred for this transition.

Here, in this path which involves transition from an entirely new state $i$ to state $c$, we not only add a new row to the transition matrix $\Pi$, but also change another existing row of $\Pi$. That is, the transit-out probability of state $k$ will be changed since it loses one count of transition to state $c$. 

\begin{itemize}
  \item $\lambda_1$ is the penalty incurred when adding this new state transition out of state $i$.
\end{itemize}

Empirically, we will use $d(i,c) = \|x_t - \mu_c\|^2 + \lambda_1$ since it still decreases the objective function value monotonically most of the time in practice and yields better results.

(III). We compute the minimum among all states:

$$
\alpha(c,t) = \min_{1 \leq i \leq K} \alpha(i,t-1) + d(i,c) \quad (4.24)
$$

To check if this time step can be created as a new hidden state, we find

$$
d_{\text{min}} = \min_{i,c} d(i,c). \quad (4.25)
$$

If $d_{\text{min}} > \lambda_1 + \lambda_2$, we create a new hidden state (this is the penalty incurred for transiting to a new state and a new hidden state). We let $K = K + 1$,

$$
\alpha(K,t) = \min_{1 \leq k \leq K-1} \alpha(k,t-1) + \lambda_1 + \lambda_2. \quad (4.26)
$$

This follows from the description of (II).

Now, we prove the correctness of the upper bounds. For the first bound, when we consider transitioning from $i$ to $c$ such that $\Pi_{ic} = 0$, we are adding a new transition out of state $i$, which then impacts the other transition probabilities out of state $i$. Thus, the $E_i$ bound determines how much the change in the other transition probabilities impacts the current path cost.

**Lemma 4.1.** $E_i$ is an upper bound of the possible change in the objective function value, in terms of other transition probabilities from state $i$, incurred by adding one transition to a new state in Step (I). Here, we assume the total number of transitions from state $i$ is fixed.
Proof. Denote \( n_i \) as the total number of transitions from state \( i \), \( n_{ij} \) as the total number of transitions from state \( i \) to state \( j \) with \( n_{ij} > 0 \). Thus, the possible change for other transitions is

\[
(n_{ij} - 1) \times \left( \log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i} \right) \geq 0.
\]

Let \( f(x) = (x - 1)(\log x - \log(x - 1)), x \geq 2 \), we have

\[
f'(x) = \log x - \log(x - 1) - \frac{1}{x}
\]

\[
= \log x - [\log x + \frac{-1}{x} - \frac{1}{2x^2} + o(\frac{1}{2x^2})] - \frac{1}{x}
\]

\[
= \frac{1}{2x^2} + o(\frac{1}{2x^2}) > 0
\]

Thus, \( f(x) \) is increasing as \( x \) increases for \( x \geq 2 \). When \( x = 1 \), we have by definition \( f(1) = 0 \). Therefore,

\[
E_i = (n_{ik} - 1) \times \left( \log \frac{n_{ik}}{n_i} - \log \frac{n_{ik} - 1}{n_i} \right)
\]

\[
= \max_j (n_{ij} - 1) \times \left( \log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i} \right),
\]

where \( k = \text{argmax}_j n_{ij} \). \qed

The second bound deals with transitioning from an entirely new state. We are adding one row to the transition matrix \( \Pi \), which then also changes another row of transition probabilities out of state \( k \) where this transition previously is coming from state \( k \).

Lemma 4.2. 0 is an upper bound of possible change in the objective function value in terms of the transition probabilities from state \( k \) where the original transition occurred for this transition in Step (II).

Proof. Denote \( n_k \) the total number of transitions from state \( k \), \( n_{kj} \) the total number of transitions from state \( k \) to state \( j \) with \( n_{kj} > 0 \). Without loss of generality, we
assume that the lost transitions are all from \( n_{ki} \) and the number of lost transitions is \( x \). Thus, the change is

\[
\sum_j n_{kj} \log \frac{n_{kj}}{n_k} - \sum_{j \neq i} n_{kj} \log \frac{n_{kj}}{n_k - x} - (n_{ki} - x) \log \frac{n_{ki} - x}{n_k - x}
\]

\[
= \sum_{j \neq i} n_{kj} \log \frac{n_k - x}{n_k} + n_{ki} \left( \log \frac{n_{ki}}{n_k} - \log \frac{n_{ki} - x}{n_k - x} \right) + x \log \frac{n_{ki} - x}{n_k - x}
\]

\[
\leq 0,
\]

since \( \frac{n_{ki}}{n_k} \leq 1, \frac{n_k - x}{n_k} < 1, \frac{n_{ki} - x}{n_k - x} \leq 1, \) and \( \frac{n_{ki}}{n_k} \geq \frac{n_{ki} - x}{n_k - x} \). 

From Lemma 4.1 and 4.2, we know that in each time step we compute an upper bound of the minimum non-penalized objective function value. And since we add \( \lambda_1 \) and \( \lambda_2 \) in \( \alpha \) whenever we transition to a state where there was no existing transitions or we create an entirely new state respectively, we also manage to upper-bound the newly-introduced penalties. Combining these two together, we have the following:

**Proposition 4.3.** The computation of \( \alpha \) gives an upper bound of the minimum cost of every possible path.

### 4.3.2 Local State Creation

Now, we look at Step 5 in Algorithm 4.1. After finishing the forward-backward step and updating the means and transition matrix, we check locally if we should create a new state by determining, for all \( i \) and \( k \), if the objective function is lowered when all data points in state \( k \) that reached state \( k \) via state \( i \) are put into a new state. In particular, we determine and execute the single best such move.

In detail, for each state \( 1 \leq i \leq K \), we consider all the time steps \( A_i \) that are one step from a time step with state \( i \). These time steps \( A_i \) can be grouped by their states: \( A_i(1), \ldots, A_i(K) \), where \( A_i(j) \) indicates time steps belonging to state \( j \). Then, for all
states \( j \) with \( |A_i(j)| > 0 \), we can compute the objective function contribution from these time steps. We have

\[
old = \sum_{t \in A_i(j)} \left\{ \|x_t - \mu_{z_t}\|^2 - \lambda \times \log \Pi_{i,z_t} - \lambda \times \log \Pi_{z_t,z_{t+1}} \right\}.
\]

If we let \( A_i(j) \) be a new hidden state, the contribution would be

\[
new = \sum_{t \in A_i(j)} \left\{ \|x_t - \bar{x}_t\|^2 - \lambda \times \log \Pi_{i,z_t} - \lambda \times \log \Pi_{K+1,z_{t+1}} \right\} + \lambda_1 \times K_i(j) + \lambda_2.
\]

where

- \( \bar{x}_t = \frac{1}{|A_i(j)|} \sum_{t \in A_i(j)} x_t \);

- Let \( B_i(j) \) be the time steps which are one-step from those of \( A_i(j) \). \( \Pi_{K+1,z_{t+1}} \) is just the empirical transition probability from time step \( t \in A_i(j) \) to the next time step \( t + 1 \in B_i(j) \).

- \( K_i(j) \) is the number of states in \( B_i(j) \).

- Since there is no change of \( \lambda_1 \) penalty for state \( i \), we do not need to consider this in the “old” contribution.

If \( new < old \), then \( A_i(j) \) is considered as a candidate for a new hidden state.

After a whole pass of the states, we find the largest reduction

\[
[i, j] \in \text{argmax}_{i,j} \; old_i(j) - new_i(j),
\]

and create \( A_i(j) \) as a new hidden state.

In summary, in the forward-backward step, the cost we compute is an upper bound of both the non-penalized objective function value and newly-introduced penalties.

If there are no new states created or no new transitions happened, the upper bound
for the newly-introduced penalties is zero. That is, we preserve the non-penalized objective function value of any path from previous iteration, and upper-bound the non-penalized objective function value and the newly-introduced penalties of any new path we find. In the local move step, we further reduce the objective function value by considering locally for each state we get in the forward-backward step. Therefore, we have

**Proposition 4.4.** The algorithm decreases the objective function value in each iteration.

*Proof.* We know that the best path $P_{old}$ obtained from last iteration would be a possible path in this iteration, and from the forward-backward step, it would preserve its cost from the previous iteration. Thus, we have

$$\alpha(P_{new}) = \min_{all \ possible \ P} \alpha(P) \leq \alpha(P_{old}).$$

From Proposition 1, we know $\alpha$ gives the upper bound of the additional incurred cost, thus

$$\text{cost}(P_{new}) \leq \text{cost}(P_{old}).$$

We also further decreases the cost from the local move, which preserves the inequality. That is, we have

$$\text{cost}(P_{new}) \leq \text{cost}(P_{old}).$$

\[\square\]
4.4 Experiments

We now present a brief set of experiments designed to highlight the benefits of our approach. Namely, we will show that our methods have benefits over the nonparametric HMM algorithms in terms of speed and accuracy.

4.4.1 Synthetic Data

First, we compare our nonparametric algorithm with the Beam sampler [Gael et al., 2008] for the iHMM. A sequence of length 3000 was generated over a varying number of hidden states with the all-zeros transition matrix except that $\Pi_{i,i+1} = 0.8$ and $\Pi_{i,i+2} = 0.2$ (when $i + 1 > K$, the total number of states, we choose $j = i + 1 \mod K$ and let $\Pi_{i,j} = 0.8$, and similarly for $i+2$). Observations are sampled from symmetric
Gaussian distributions with means of \{3, 6, \ldots, 3K\} and a variance of 0.9. Figure 4.2 shows one example sequence with 4 hidden states for the first 150 time steps.

The data described above is trained using our nonparametric algorithm (\texttt{AsymIHMM}) and the Beam sampler. For our nonparametric algorithm, we perform a grid search over all three parameters and select the parameters using a heuristic. For the Beam sampler, we use the following hyper-parameter settings: gamma hyper-priors \((4, 1)\) for \(\alpha\), \((3, 6)\) for \(\gamma\), and a zero mean normal distribution for the base \(H\) with the variance equal to 10\% of the empirical variance of the dataset. We also normalize the sequence to have zero mean. The number of selected samples is varied among 10, 100, and 1000 for different numbers of states, with 5 thinning iterations. (Note: there are no burn-in iterations and all samplers are initialized with a randomly initialized 20-state labeling.)

In Figure 4.3, the training accuracy and running time for the two algorithms are shown, respectively. The accuracy of the Beam sampler is given by the highest among all the samples selected. The accuracy is shown in terms of the normalized mutual information (NMI) score (in the range of \([0,1]\)), since the sampler may output different number of states than the ground truth and NMI can handle this situation gracefully. We can see that, in all datasets, our algorithm performs better than the sampling method in terms of accuracy, but with running time similar to the sampler with only 10 samples. For these datasets, we also observe that the EM algorithm for the parametric HMM (not reported in the figure) can easily output a smaller number of states than the ground truth, which yields a smaller NMI score. We also observe that the Beam sampler is highly sensitive to the initialization of hyper-parameters.
Figure 4.3: Comparisons of the training accuracy and time (log-scale) of our AsymIHMM algorithm and the Beam Sampler on the synthetic Gaussian hidden Markov model data for 3000 time steps.
Putting flat priors over the hyper-parameters can ameliorate the situation, but also substantially increases the number of samples required.

### 4.4.2 S&P Index Prediction

Our next experiment illustrates the advantages of our algorithms in a financial prediction problem. The sequence consists of 3668 values of the Standard & Poor’s 500 index on consecutive trading days from Jan 02, 1998 to July 30, 2012\(^2\). The index exhibited appreciable variability in this period, with both bull and bear runs (see Figure 4.4 for an illustration). The goal here is to predict the index value on a test sequence of trading days, and compare the accuracies of different algorithms. We use a training window of length 500. This window size is empirically chosen to provide a balance between prediction accuracy and runtime. The algorithms are trained on the sequence from index \(i\) to \(i+499\), and then the \((i+500)\)-th value is predicted and compared with the actual recorded value at that point in the sequence, where \(i\) ranged from 1 to 3168.

For this dataset, we consider our nonparametric algorithm (AsymIHMM), the asymptotic parametric algorithm with fixed number of states \(K\) (AsymHMM) which ignores the two penalties on the number of states in \((4.19)\)\(^3\), and the EM algorithm (HMM) for the parametric HMM. The Beam sampler is too slow to run over this data, as each individual prediction took on the order of minutes. For AsymHMM, the cluster means are initialized with \(K\)-means and the transition matrices are initialized with random stochastic matrices. Five latent states are empirically chosen for AsymHMM and HMM. As for AsymIHMM, we tune the parameters to get five states on average as well. To

\(^2\)[http://research.stlouisfed.org/fred2/series/SP500/downloaddata]

\(^3\)Segment \(k\)-means algorithm in [Rabiner, 1989] is a special case.
get predicted observation at time step $T + 1$ given observations up to time step $T$, we use the weighted average of the learned state means, weighted by the transition probabilities given by the state of the observation at time step $T$.

Figure 4.4 shows the predicted values from time step 501 to time step 3668 along with the true index values in that same time period. We can see that both the AsymIHMM and AsymHMM perform noticeably better than the HMM. They both are able to better follow the actual curve across all kinds of temporal fluctuations. Indeed, the difference is most stark in the areas of high-frequency oscillations. While the standard EM algorithm returns an averaged-out prediction, our algorithms latch onto the underlying behavior almost immediately and return more accurate predictions. Quantitatively, we measure the prediction accuracy using the mean absolute percentage (MAP) error, which is the average of the absolute differences between of the predicted and true values expressed as percentages of the true values. Table 4.1 presents the MAP error.
Table 4.1: Mean absolute percentage (MAP) error of AsymIHMM, AsymHMM, and HMM for the prediction error of the S&P 500 index.

<table>
<thead>
<tr>
<th></th>
<th>AsymIHMM</th>
<th>AsymHMM</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP error</td>
<td>2.78%</td>
<td>3.16%</td>
<td>6.44%</td>
</tr>
</tbody>
</table>

results. We can see that AsymHMM reduces the error by more than 50% from the HMM, which emphasizes the different impacting effects with varying $\lambda$ values. AsymIHMM in addition reduces the error by more than 10%. This manifests the great advantage of nonparametric methods when comparing with parametric ones; that is, being able to adapt the number of states with different observation sequences allows different modeling granularities conditioned on the observed data.

4.5 Conclusion

In this chapter, we have considered an asymptotic treatment of the infinite HMM. Crucially, this analysis allows us to expand the small-variance asymptotics to sequential models, obtaining non-probabilistic formulations inspired by the hidden Markov models. Our analysis utilizes the framework of deriving the objective functions from asymptotics of the joint likelihood [Broderick et al., 2013], and proposes a novel dynamic-programming-based algorithm for sequential data with a non-fixed number of states.
Chapter 5: Combinatorial Topic Models

Topic modeling has become a cornerstone of unsupervised learning on large document collections. While the roots of topic modeling date back to latent semantic indexing [Deerwester et al., 1990] and probabilistic latent semantic indexing [Hofmann, 1999], the arrival of latent Dirichlet allocation (LDA) [Blei et al., 2003] was a turning point that transformed the community’s thinking about topic modeling. LDA led to several follow-ups that address some limitations of the original model [Blei and Lafferty, 2006; Wang and Grimson, 2007], while paving the way for subsequent advances in Bayesian learning, including variational inference methods [Teh et al., 2006b], nonparametric Bayesian models [Blei et al., 2004; Teh et al., 2006a], among others.

In this chapter, we explore small-variance asymptotics for the standard latent Dirichlet allocation (LDA) [Blei et al., 2003] topic models. The SVA approach was applied to the HDP-LDA [Teh et al., 2006a] in Chapter 3, and topic modeling was briefly presented as one of the applications of this work; however, no quantitative results were ever given on this model, nor were any sophisticated algorithms developed. A crucial question is: can one obtain a combinatorial topic model that competes favorably with probabilistic LDA models quantitatively? We answer this question positively by first formulating a combinatorial topic model via asymptotics directly on the LDA
model. The resulting objective is similar to (3.26), but with no constraint on the
global number of topics which is fixed upfront.

But merely using asymptotics to obtain a combinatorial topic model is insufficient: we also need effective algorithms to optimize the model. A direct application of Algorithm 3.2 on the model fails to compete with the main probabilistic LDA methods. The first-order greedy local assignment step can succeed only under certain circumstances and often fails to minimize the objective greatly under random initialization. This setback necessitates a new idea. We develop algorithms for optimizing this combinatorial model by using ideas from facility location and incremental refinement. Moreover, we show how our procedure can be implemented in linear complexity with the number of word tokens and the number of topics. Surprisingly, as we will see, these transform the SVA approach into a competitive topic modeling algorithm.

We demonstrate that this new approach not only improves significantly over the traditional SVA algorithms, but also competes favorably with existing state-of-the-art topic modeling algorithms; in particular, it is orders of magnitude faster than sampling-based approaches, with comparable accuracy. Moreover, we show that the sampler’s mixing time improves substantially when initialized using our combinatorial method for just a few iterations. We also compare favorably against several recent theoretically-motivated algorithms [Anandkumar et al., 2012; Arora et al., 2013; Podosinnikova et al., 2015] and variational inference methods.

This chapter is organized as follows. In Section 5.1, we review the LDA model and different probabilistic inference approaches. In Section 5.2, we present the small-variance asymptotic analysis on the LDA model. In Section 5.3, we propose a new algorithm to optimize the derived objective function based on ideas from facility
Figure 5.1: The intuitions behind latent Dirichlet allocation. We assume there are some number of “topics”, which are distributions over words and exist for the whole document collection. Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then for each word, choose a topic assignment and choose the word from the corresponding topic (source: Figure 1 in [Blei, 2012]).

location and incremental refinement. Experiment results are given in Section 5.4, where we focus on comparison with probabilistic approaches. Results are shown for both synthesized and real document data.

5.1 Latent Dirichlet Allocation

The latent Dirichlet allocation model [Blei et al., 2003] is a generative probabilistic model for collections of discrete data such as text corpora. It can also be considered as a mixture model which exploits co-occurrence information, or a continuous latent variable model which can be seen as an auto-encoder with constraints on the hidden units.
The generative process of the standard Latent Dirichlet Allocation (LDA) model is as follows:

- Sample topic distribution $\psi_t \in \Delta^V$ over the vocabulary: $\psi_t \sim \text{Dirichlet}(\beta)$, where $t \in \{1, \ldots, K\}$.

- Sample document-topic mixing distribution $\theta_j \in \Delta^K$ over the topics: $\theta_j \sim \text{Dirichlet}(\alpha)$, where $j \in \{1, \ldots, M\}$.

- For each word token $i$ in document $j$:
  - Sample the corresponding topic from the document-topic mixing distribution: $z_{ji} \sim \text{Categorical}(\theta_j)$.
  - Sample the word from the corresponding topic: $w_{ji} \sim \text{Categorical}(\psi_{z_{ji}})$.

Here, $\Delta^d$ represents $d$-dimensional simplex, $V$ is the number of words in the vocabulary, $M$ is the total number of documents, $K$ is the total number of topics, $\alpha$ and $\beta$ are scalar-valued (i.e., we are using a symmetric Dirichlet distribution). Figure 5.1 presents a graphical illustration.

Denote $\mathbf{W}$ as the vector denoting all word tokens in all documents, $\mathbf{Z}$ as the topic indicators of all word tokens in all documents, $\mathbf{\theta}$ as the concatenation of all the $\theta_j$ variables, and $\mathbf{\psi}$ as the concatenation of all the $\psi_t$ variables. Also let $N_j$ be the total number of word tokens in document $j$. We can write down the full joint likelihood of the model as:

$$p(\mathbf{W}, \mathbf{Z}, \mathbf{\theta}, \mathbf{\psi}|\alpha, \beta) = \prod_{t=1}^K \text{Dirichlet}(\psi_t) \prod_{j=1}^M \text{Dirichlet}(\theta_j) \prod_{i=1}^{N_j} \text{Categorical}(z_{ji}) \cdot p(w_{ji}|\psi_{z_{ji}}). \quad (5.1)$$

Now, we can eliminate variables to simplify inference by integrating out $\mathbf{\theta}$ to obtain

$$p(\mathbf{W}, \mathbf{Z}, \mathbf{\psi}|\alpha, \beta) = \int_{\theta} p(\mathbf{W}, \mathbf{Z}, \mathbf{\theta}, \mathbf{\psi}|\alpha, \beta) d\mathbf{\theta}.$$
After simplification, we obtain

\[
p(W, Z, \psi | \alpha, \beta) = \text{data likelihood} \times \left[ \prod_{t=1}^{K} p(\psi_t | \beta) \prod_{j=1}^{M} \prod_{i=1}^{N_j} p(w_{ji} | \psi_{z_{ji}}) \right] \times \left[ \prod_{j=1}^{M} \frac{\Gamma(\alpha K)}{\Gamma(\sum_{i=1}^{K} n_{ji}^t + \alpha K) \Gamma(\alpha)} \prod_{i=1}^{K} \frac{\Gamma(n_{ji}^t + \alpha)}{\Gamma(\alpha)} \right], \tag{5.2}
\]

where \(n_{ji}^t\) is the number of word tokens in document \(j\) assigned to topic \(t\).

Many techniques have been developed for efficient inference of LDA, notably the collapsed Gibbs sampler (CGS) [Griffiths and Steyvers, 2004], and variational inference methods [Blei et al., 2003; Teh et al., 2006b]. Among MCMC and variational techniques, CGS typically yields excellent results and is guaranteed to sample from the desired posterior with sufficiently many samples. However, it can be slow and many samples may be required before mixing. Indeed, it has been shown that the mixing time of CGS is exponential in the length of the documents [Jonasson, 2017].

Method-of-moments techniques have also been applied on LDA, where tensor decomposition is employed to derive the topics. Both LDA moments [Anandkumar et al., 2012] and the discrete independent component analysis cumulants [Podosinnikova et al., 2015] have been considered. A complementary line of algorithms starts with [Arora et al., 2013], which take a “factorization” approach (see Figure 5.2 for an illustration) and consider certain separability assumptions on the input data to circumvent NP-Hardness of the nonnegative matrix factorization. These works have shown performance competitive to Gibbs sampling in some scenarios while also featuring theoretical guarantees.
5.2 Asymptotic Analysis of Latent Dirichlet Allocation

In the following, we present the asymptotic treatment for the latent Dirichlet allocation model. We will focus on examining small-variance asymptotics directly on the joint probability distribution, as done in [Broderick et al., 2013] and Chapter 4.

Recall the joint likelihood of LDA in (5.2):

$$p(W, Z, \psi | \alpha, \beta) = \left[ \prod_{t=1}^{K} p(\psi_t | \beta) \prod_{j=1}^{M} \prod_{t=1}^{N_j} p(w_{j,t} | \psi_{z_{j,t}}) \right] \times \left[ \prod_{j=1}^{M} \frac{\Gamma(\alpha K)}{\Gamma(\sum_{t=1}^{K} n_{j,t} + \alpha K)} \prod_{t=1}^{K} \frac{\Gamma(n_{j,t} + \alpha)}{\Gamma(\alpha)} \right],$$

where $K$ is the number of topics, $M$ is the number of document, $N_j$ is the number of word tokens in document $j$, and $n_{j,t}$ is the number of word tokens in document $j$ assigned to topic $t$. We can obtain the SVA objective by looking at the logarithm of the likelihood (5.2) and observing what happens when the variance goes to zero.
First, let’s look at the data likelihood part. The categorical distribution is in the exponential family, thus we can resort to Example 2.3 to represent it in the Bregman divergence form. Specifically, we can represent the categorical distribution using the KL-divergence, and write the probability of word token $w_{ji}$ as:

$$p(w_{ji} \mid \psi_{z_{ji}}) = \exp(-\text{KL}(1_{w_{ji}}, \psi_{z_{ji}})), \quad (5.3)$$

where $1_{w_{ji}} \in \Delta^V$ is the indicator vector for word $w_{ji}$. Recall the KL-divergence $\text{KL}(x, y) = \sum x_j \log(\frac{x_j}{y_j})$ and $0 \log 0 = 0$. Thus, for word token $w_{ji}$, the probability simply equals $-\log \psi_{z_{ji},w_{ji}}$.

We now resort to Section 2.2.2 to perform the asymptotic analysis. As discussed in Lemma 2.6, we can introduce another parameter, which we will call $\eta$, that scales the variance in an exponential family distribution while fixing the expectation. This new distribution may be represented, under the Bregman divergence view, as proportional to

$$\exp(-\eta \cdot \text{KL}(1_{w_{ji}}, \psi_{z_{ji}})).$$

As $\eta \to \infty$, the expectation remains fixed while the variance goes to zero, which is precisely what we require to perform small-variance analysis. Now, we have as $\beta \to \infty$,

$$-\log p(W, \psi \mid \beta, Z) \sim \eta \sum_{j=1}^{M} \sum_{i=1}^{N_j} \text{KL}(1_{w_{ji}}, \psi_{z_{ji}}), \quad (5.4)$$

where $f(x) \sim g(x)$ denotes that $f(x)/g(x) \to 1$ as $x \to \infty$. The $-\log p(\psi_t \mid \beta)$ terms vanish asymptotically since we are not scaling $\beta$.

Next, we consider the “regularization” part arising from the Dirichlet-multinomial distribution. We will choose to scale $\alpha$ appropriately as well; this will ensure that the hierarchical form of the model is retained asymptotically.
Lemma 5.1. Consider the likelihood
\[
\frac{\Gamma(\alpha K)}{\Gamma(\sum_{t=1}^{K} n_{j}^t + \alpha K)} \prod_{t=1}^{K} \frac{\Gamma(n_{j}^t + \alpha)}{\Gamma(\alpha)}.
\] (5.5)

If \( \alpha = \exp(-\lambda \cdot \eta) \), then asymptotically as \( \eta \to \infty \) we have (5.5) \( \sim \eta \lambda (K_j - 1) \), where \( K_j \) is the number of topics \( t \) utilized in document \( j \). Therefore, we have
\[
- \log p(Z|\alpha) \sim \eta \lambda \sum_{j=1}^{M} (K_j - 1).
\] (5.6)

Proof. Note that \( N_j = \sum_{t=1}^{K} n_{j}^t \). Using standard properties of the \( \Gamma \) function (\( \Gamma(z + 1) = z\Gamma(z) \)), we have that the negative logarithm of the (5.5) is equal to
\[
\sum_{n=0}^{N_j - 1} \log(\alpha K + n) - \sum_{i=1}^{K} \sum_{n=0}^{n_{j}^t - 1} \log(\alpha + n).
\]

All of the logarithmic summands converge to a finite constant whenever they have an additional term besides \( \alpha \) or \( \alpha K \) inside. The only terms that asymptotically diverge are those of the form \( \log(\alpha K) \) or \( \log(\alpha) \), that is, when \( n = 0 \). The first term always occurs. Terms of the type \( \log(\alpha) \) occur only when, for the corresponding \( t \), we have \( n_{j}^t > 0 \). Recalling that \( \alpha = \exp(-\lambda \cdot \eta) \), we can conclude that the negative log of the Dirichlet multinomial term becomes asymptotically \( \eta \lambda (K_j - 1) \), where \( K_j \) is the number of topics \( t \) in document \( j \) where \( n_{j}^t > 0 \), i.e., the number of topics currently utilized by document \( j \). (The maximum value for \( K_j \) is \( K \), the total number of topics.)

Combining results from (5.4) and (5.6), we can conclude that the negative log-likelihood asymptotically yields
\[
- \log p(Z, W, \psi|\alpha, \beta) \sim \eta \left[ \sum_{j=1}^{M} \sum_{i=1}^{N_j} \text{KL}(1_{w_{ji}}, \psi_{z_{ji}}) + \lambda \sum_{j=1}^{M} (K_j - 1) \right],
\] 81
as $\eta \to \infty$. This leads to the objective function

$$\min_{Z,\psi} \sum_{j=1}^{M} \sum_{i=1}^{N_j} \text{KL}(1_{w_{ji}}, \psi_{z_{ji}}) + \lambda \sum_{j=1}^{M} (K_j - 1).$$  \hspace{1cm} (5.7)

Recall that $\text{KL}(1_{w_{ji}}, \psi_{z_{ji}})$ is just $-\log \psi_{z_{ji},w_{ji}}$. Thus we obtain a $K$-means-like term that says that any word should be “close” to its assigned topic in terms of KL-divergence under the word co-occurrence constraint enforced with reasonable $\lambda$ value. Note that (5.7) reduces to the document-level $K$-means problem with $\lambda \to \infty$, and the token-level $K$-means with $\lambda \to 0$.

Note that we did not scale $\beta$, to obtain a simpler objective with only one parameter (other than the total number of topics), but let us say a few words about scaling $\beta$. A natural approach is to further integrate out $\psi$ of the joint likelihood, as is done with the collapsed Gibbs sampler. One would obtain additional Dirichlet-multinomial distributions, and properly scaling as discussed above would yield a simple objective that places penalties on the number of topics per document as well as the number of words in each topic. Optimization would be performed only with respect to the topic assignment matrix. Future work would consider the effectiveness of such an objective function for topic modeling.

### 5.3 Algorithms

In this section, we will discuss algorithms that optimize the combinatorial objective function (5.7). First, we discuss a locally-convergent algorithm similar to $K$-means and the hard topic modeling algorithm discussed in Section 3.2.4. Then, we introduce two more powerful techniques: (i) a word-level assignment method that arises from connections between our proposed objective function and the facility location problem [Korte and Vygen, 2012]; and (ii) an incremental topic refinement method.
Despite the apparent complexity of our algorithms, we show that the per-iteration
time matches that of the collapsed Gibbs sampler (while empirically converging in just
a few iterations, as opposed to the thousands typically required for Gibbs sampling).

Algorithm 5.1 Basic Batch Algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>Words: ( W ), number of topics: ( K ), Topic penalty: ( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Topic Assignments ( Z ).</td>
</tr>
</tbody>
</table>

1: Initialize \( Z \) and topic vectors \( \psi_1, ..., \psi_K \).
2: Compute initial objective function (5.7) using \( Z \) and \( \psi \).
3: repeat
   //Update assignments:
   4: for every word token \( i \) in every document \( j \) do
   5:   Compute distance \( d(j, i, t) \) to topic \( i \): \(-\log(\psi_{t,w,ji})\).
   6:   If \( z_{ji} \neq t \) for all tokens \( i \) in document \( j \), add \( \lambda \) to \( d(j, i, t) \).
   7:   Obtain assignments via \( z_{ji} = \arg\min_t d(j, i, t) \).
   8: end for
   //Update topic vectors:
   9: for every element \( \psi_{tw} \) do
   10:   \( \psi_{tw} = \frac{\text{number of occurrences of word } w \text{ in topic } t}{\text{total number of word tokens in topic } t} \).
   11: end for
12: Recompute objective (5.7) using updated \( Z \) and \( \psi \).
13: until no change in objective function.

5.3.1 The Basic Batch Algorithm

We first describe a basic iterative algorithm for optimizing the combinatorial hard
LDA objective (5.7). The basic algorithm follows the \( K \)-means style – we perform
alternate optimization by first minimizing with respect to the topic assignments for
each word token (the \( Z \) values) and then minimizing with respect to the topics (the
\( \psi \) vectors). We continue this alternating strategy until convergence. Algorithm 5.1
shows the basic iterative algorithm.
First, consider the minimization with respect to $\psi$, with $Z$ fixed. In this case, the penalty term in (5.7) for the number of topics per document is not relevant to the minimization. Therefore the minimization can be performed in closed form by computing means based on the assignments, due to nice properties of the KL-divergence [Banerjee et al., 2005]. The topic vectors will be computed as follows:

$$
\psi_{tu} = \frac{\sum_{j=1}^{M} n_{jtu}}{\sum_{j=1}^{M} n_{jt}},
$$

(5.8)

where $n_{jtu}$ is the number of word token $u$ in document $j$ assigned to topic $t$, and $n_{jt}$ is the number of tokens in document $j$ assigned to topic $t$. That is, the topic vector entry $\psi_{tu}$ will simply be equal to the number of occurrences of word $u$ assigned to topic $t$ normalized by the total number of word tokens assigned to topic $t$.

Next, consider the minimization with respect to $Z$ with fixed $\psi$. Because of the presence of the penalty terms for the number of topics per document, this assignment step can be shown to be NP-hard [Vazirani, 2001]. We follow a strategy similar to Algorithm 3.1. In particular, we compute the KL-divergence between each word token $w_{ji}$ and every topic $t$. For any topic $t$ that is not currently occupied by any word token in document $j$ (i.e., $z_{ji} \neq t$ for any token $i$ in document $j$), we penalize the divergence by $\lambda$, which corresponds to the additional contribution to the objective function. We then obtain new assignments by re-assigning each word token to the “nearest” topic based on the divergences (including any penalties).

The running time of this algorithm can be shown to be $O(NK)$ per iteration since both steps run in linear time, where $N$ is the total number of word tokens. It can also be shown that this algorithm is guaranteed to converge to a local optimum, similar to $K$-means and DP-means [Kulis and Jordan, 2012]. The argument follows that each updating step cannot increase the objective function (5.7).
5.3.2 Improved Word Assignments

The basic algorithm has the advantage that it achieves local convergence, and can be implemented to run efficiently. However, as we will see in Section 5.4, it is also prone to falling into poor local optima due to the inability of the assignment step to find the right associations. In this section, we discuss and analyze an alternative assignment technique for \( Z \), which may be used as an initialization to the locally-convergent basic algorithm or to replace it completely.

Take a closer look at the assignment problem within (5.7):

\[
\min_Z \sum_{j=1}^{M} \left[ \sum_{i=1}^{N_j} KL(1_{w_{ji}}, \psi_{z_{ji}}) + \lambda(K_j - 1) \right].
\]  

(5.9)

With \( \psi \) fixed, (5.9) reduces to \( M \) independent assignment problems. The assignment problem for one document has a natural connection with the facility location problem.

**Facility Location problem.** The uncapacitated facility location (UFL) problem [Korte and Vygen, 2012] consists of: a finite set \( D \) of clients, a finite set \( F \) of potential facility locations, a fixed cost \( f_i \in \mathbb{R}_{+} \) for opening each facility \( i \in F \), and a distance cost \( d_{ij} \in \mathbb{R}_{+} \) for each \( i \in F \) and \( j \in D \). The task is trying to find a subset \( S \) of open facilities and an assignment \( z : D \rightarrow S \) of the clients to open facilities, such that the sum of facility costs and distance costs

\[
\sum_{i \in S} f_i + \sum_{j \in D} d_{z(j)j}.
\]  

(5.10)

is the minimum (see Figure 5.3 for an illustration). The UFL problem can be transformed into an instance of a set cover problem \((S, cost)\):

- Let \( S = \{(i, A) : i \in F, A \subset D\} \). That is, \((i, A)\) covers elements of the set \( A \).
- The cost of the set \((i, A)\) is \( f_i + \sum_{j \in A} d_{ij} \).
Figure 5.3: The facility location problem. There are a pool of fixed clients and a set of fixed potential facility locations (grey dots and dark-green factory symbols). We try to find a subset of facilities which gives the smallest combined facility cost and distance cost. Here, we have 4 open facilities (dark-green factory symbol), and 6 unopened facilities (grey dots) (source: Discrete Optimization Group DISOPT, École polytechnique fédérale de Lausanne).
It can then be solved using the greedy algorithm for set cover problems [Vazirani, 2001]: for every iteration, chooses the set \((i, A)\) which minimizes

\[
cost(i, A) \frac{|A \cap Uncovered|}{|A|}.
\]

**Connection with UFL.** Consider the topic assignment problem for document \(j\):

\[
\min_{Z_j} \sum_{i=1}^{N_j} KL(1_{w_{ji}}, \psi_{z_{ji}}) + \lambda(K_j - 1).
\]

We can now map it to the UFL problem as follows:

- \(N_j\) word tokens in document \(j\) as the pool of clients,
- \(K\) global topics as potential facility locations,
- \(\lambda\) as the fixed facility cost for opening one facility,
- \(KL(1_{w_{ji}}, \psi_t)\) as the distance cost.

Then the UFL objective (5.10) corresponds exactly to the assignment problem (5.11) for combinatorial topic modeling (ignoring the \(-1\) constant).

**Algorithm 5.2** Improved Word Assignments for \(Z\)

**Input:** Words: \(W\), number of topics: \(K\), Topic penalty: \(\lambda\), Topics: \(\psi\)

**Output:** Topic Assignments \(Z\).

1. **for** every document \(j\) **do**
2. Let \(f_t = \lambda\) for all topics \(t\).
3. Initialize all word tokens to be unmarked.
4. **while** there are unmarked tokens **do**
5. Pick the topic \(t\) and set of unmarked tokens \(W\) that minimizes (5.12).
6. Let \(f_t = 0\) and mark all tokens in \(W\).
7. Assign \(z_{ji} = t\) for all \(i \in W\).
8. **end while**
9. **end for**
**Improved Assignment Algorithm.** Algorithm 5.2 details the improved assignment strategy for word tokens based on the simple greedy algorithm for the UFL problem. The algorithm, must select, among all topics and all unmarked tokens $W$, the minimizer to

$$f_t + \sum_{i \in W} KL(1_{w_{ji}}, \psi_t).$$

(5.12)

This algorithm appears to be computationally expensive, requiring multiple rounds of marking where each round requires us to find a minimizer over exponentially-sized sets. Surprisingly, under mild assumptions we can use the structure of our problem to derive an efficient implementation of this algorithm that runs in total time $O(NK)$. For particular instances of UFL, this greedy strategy is known to produce a provably approximate solution to the UFL problem. For instance, if the distances between clients and facilities is a metric, it is known that the greedy algorithm can achieve a constant factor approximation guarantee [Jain et al., 2003]. Unfortunately, the KL-divergence does not constitute a metric, and therefore these guarantees do not directly carry over to our case. But one may resort to the $\mu$-similar Bregman divergences [Ackermann and Blömer, 2009] to prove similar guarantees.

**Efficient Implementation.** Now, we describe the details of the efficient $O(NK)$ implementation of Algorithm 5.2. Note that, for a fixed size of $W$ and a given topic $t$, the best choice of $W$ is obtained by selecting the $|W|$ closest tokens to $\psi_t$ in terms of the KL-divergence. Thus, as a first pass, we can obtain the correct points to mark by appropriately sorting KL-divergences of all tokens to all topics, and then searching over all sizes of $W$ and topics $t$.

We make several observations about the sorting procedure. First, the KL-divergence between a word and a topic can be performed purely on counts of words within topics.
Thus, for a given topic, the sorted words are obtained exactly by sorting word counts within a topic in decreasing order. Second, we can use a linear-time sorting algorithm such as counting sort or radix sort to efficiently sort the items since the word counts are just integers. In the case of counting sort, for instance, if we have \( n \) integers whose maximum value is \( k \), the total running time is \( O(n + k) \). In our case, we perform many sorts. Each sort considers, for a fixed document \( j \), sorting word counts to some topic \( t \). Suppose there are \( n_{jt} \) tokens with non-zero counts to the topic, and the maximum word count is \( m_{jt} \). Then the running time of this sort is \( O(n_{jt} + m_{jt}) \). Across the document, we do this for every topic, making the running time scale as \( O(\sum_t (n_{jt} + m_{jt})) = O(N_j K) \), where \( N_j \) is the number of word tokens in document \( j \). Across all documents this sorting then takes \( O(NK) \) time.

Last, we note that sorting only once per run of the algorithm will be sufficient. Once we have sorted lists for words to topics, if we mark some set \( \mathcal{W} \), we can efficiently remove these words from the sorted lists and keep the updated lists in sorted order. Removing an individual word from a single sorted list can be done in constant time. Since each word token is removed exactly once during the algorithm, and must be removed from each topic, the total time to update the sorted lists during the algorithm is \( O(NK) \).

At this point, we still do not have a procedure that runs in \( O(NK) \) time. In particular, we must find the minimum of (5.12) at each round of marking. Naively this is performed by traversing the sorted lists and accumulating the value of the above score via summation. In the worst case, each round would take a total of \( O(NK) \) time across all documents, so if there are \( R \) rounds on average across all the documents,
the total running time would be $O(NKR)$. However, we do not need to traverse entire sorted lists in general. Consider a fixed document, where we try to find the minimizer for some fixed topic $t$ of the above expression. We can show that the value monotonically decreases until hitting the minimum value, and then monotonically increases. This can be formalized as follows:

**Proposition 5.2.** Let $g_t(n)$ be the value of (5.12) for the candidate set $W$ of size $n$ for topic $t$. For any reasonably large $\lambda > 0$, $g_t(n)$ will first decrease monotonically hitting the minimum value, then monotonically increase.

**Proof.** Recall that the KL-divergence is equal to the negative logarithm of the number of occurrences of the corresponding word token divided by the total number occurrences of tokens in the topic. Write this as $\log c_t - \log c_{tj}$, where $c_t$ is the total number of occurrences of tokens in topic $t$ and $c_{tk}$ is the count of word $k$ in topic $t$ (assuming word is ordered by the decreasing order of occurrences). Now, we can write $g_t(n)$ as

$$g_t(n) = \frac{f_t + n \log c_t - \sum_{j=1}^{n} \log c_{tj}}{n} = \log c_t + \frac{f_t - \sum_{j=1}^{n} \log c_{tj}}{n}.$$  

Let $g_t(x) = \log c_t + \frac{f_t - \int_{0}^{x} \log c_{tj} dj}{x}$, $x > 0$, we have

$$g_t'(x) = 0 - \frac{f_t}{x^2} - \left[ -\frac{1}{x^2} \int_{0}^{x} \log c_{tj} dj + \frac{1}{x} \log c_{tx} \right]$$

$$= \frac{1}{x^2} \left[ \int_{0}^{x} \log c_{tj} dj - x \log c_{tx} - f_t \right].$$

Thus $g_t'(0^+) = -f_t < 0$, and

$$g_t'(1) = \int_{0}^{1} \log c_{tj} dj - \log c_{t1} - f_t < 0.$$
Since the positivity of $g'_t(x)$ only depends on the positivity of $\int_{0^+} x \log c_{tj} dj - x \log c_{tx} - f_t = h_t(x)$, we have

$$h'_t(x) = \log c_{tx} - [\log c_{tx} + x \nabla \log c_{tx}]$$

$$= -x \nabla \log c_{tx} > 0,$$

where the last inequality follows that $\log c_{tx}$ is decreasing as $x$ increases; that is, $\nabla \log c_{tx} < 0$. This shows that $h_t(x)$ is monotonically increasing for $x > 0$. Since $g'_t(1) < 0$, we know that $g'_t(x)$ will be monotonically increasing until it hits zero, where $g_t(x)$ will be monotonically decreasing. Then $g'_t(x)$ will become positive, where $g_t(x)$ changes to increase monotonically.

In words, the above proof demonstrates that, once the $g_t(n)$ stops decreasing, it will not decrease any further, i.e., the minimum has been found. Thus, once $g_t(n)$ starts to increase as $W$ gets larger, we can stop and the best score (i.e., the best set $W$) for that topic $t$ has been found. We do this for all topics $t$ until we find the best set $W$. Under the mild assumption that the size of the chosen minimizer $W$ is similar (specifically, within a constant factor) to the average size of the best candidate sets $W$ across the other topics (an assumption which holds in practice), then it follows that the total time to find all the sets $W$ also takes $O(NK)$ time.

### 5.3.3 Incremental Topic Refinement

In this section, we try to refine the assignments and the topics simultaneously exploring the hierarchical structure in topic modeling: we have both word-level assignments and “mini-topics” (formed by word tokens in the same document which are assigned to the same topic. See Figure 5.4 for an illustration). Unlike for the word tokens
themselves, explicitly refining the assignments of mini-topics should help in achieving better word-coassignment within the same document. This can be considered as analogously to the block coordinate descent algorithm [Bertsekas, 1999] in the continuous optimization and is also similar to the local search techniques [Dhillon et al., 2002] in the clustering literature.

Specifically, we consider an incremental topic refinement scheme that works as follows. For a given document, we consider the possibility of swapping all word tokens assigned to the same topic within that document to another topic. Different from the computation in Algorithm 5.1, we compute the true change in the objective function; that is the difference between the objective functions if we updated the topic assignments for those tokens and also updated the resulting topic vectors simultaneously. This true objective function change computation would make crucial differences here. At first glance, it seems that this computation may be computationally expensive because we try to find the true difference. However, we will show in the following that the change in objective function can be computed with only the mini-topic and
the two involving topics. Specifically, given a topic $t$ and all the word tokens $U_t$ associated with it, we have

$$\sum_{w \in U_t} \text{KL}(1_w, \psi_t) = \sum_{w \in U_t} [\phi(1_w) - \phi(\psi_t) - (1_w - \psi_t)^T \nabla \phi(\psi_t)]$$

$$= -n'_t \cdot \phi(\psi_t) + \sum_{w \in U_t} \phi(1_w),$$

where $\phi(\psi_t) = \sum_w \psi_{tw} \log \psi_{tw}$, and $n'_t$ is the total number of word tokens assigned to topic $t$. The last equality follows that $\psi_t = \sum_{w \in U_t} 1_w / n'_t$. Then, for document $j$ and its mini-topic $S^t_j$ formed by its word tokens assigned to topic $t$, the change of removing $S^t_j$ from topic $t$ is

$$\sum_{w \in U^-_t} \text{KL}(1_w, \psi^-_t) - \sum_{w \in U^+_t} \text{KL}(1_w, \psi_t)$$

$$= -(n'_t - n^t_j \cdot) \phi(\psi^-_t) + \sum_{w \in U^-_t} \phi(1_w) + n'_t \cdot \phi(\psi_t) - \sum_{w \in U^+_t} \phi(1_w)$$

$$= -(n'_t - n^t_j \cdot) \phi(\psi^-_t) + n'_t \cdot \phi(\psi_t) - \sum_{w \in S^t_j} \phi(1_w), \quad (5.13)$$

and similarly, the change of adding $S^t_j$ to topic $t' \neq t$ is

$$\sum_{w \in U^+_t} \text{KL}(1_w, \psi^+_t) - \sum_{w \in U^-_t} \text{KL}(1_w, \psi_t)$$

$$= -(n''_t - n^t_j \cdot) \phi(\psi^+_t) + n''_t \cdot \phi(\psi_t) + \sum_{w \in S^t_j} \phi(1_w), \quad (5.14)$$

where $n^t_j$ is the number of tokens in document $j$ assigned to topic $t$, $U^-_t$ and $U^+_t$ are the updated sets of word tokens, $\psi^-_t$ and $\psi^+_t$ are the updated topics.

Adding (5.13) and (5.14), and considering the possible effect of $\lambda$, the change in objective function (5.7) of assigning $S^t_j$ to topic $t'$ can be computed by

$$\Delta(S, t, t') = -(n'_t - n^t_j \cdot) \phi(\psi^-_t) - (n''_t + n^t_j \cdot) \phi(\psi^+_t)$$

$$+ n'_t \cdot \phi(\psi_t) + n''_t \cdot \phi(\psi_{t'}) - \lambda \mathbb{1}[t' \in T_j], \quad (5.15)$$

93
where $\mathcal{T}_j$ is the set of all the topics used in document $j$. It still appears that this computation be computationally expensive. However, computing $\Delta(S^t_j, t, t')$ can be computed in $O(|S^t_j|)$ time, if the topics are maintained by count matrices. Only the counts involving the words in the mini-topic and the total counts are affected. Since we compute the change across all topics, and across all mini-topics $S^t_j$, the total running time of the incremental topic refinement is $O(NK)$, as for the basic batch algorithm and the improved word assignment algorithm.

**Algorithm 5.3 Incremental Topic Refinements for $Z$**

**Input:** Words: $W$, number of topics: $K$, Topic penalty: $\lambda$, Assignment: $Z$, Topics: $\psi$

**Output:** Topic Assignments $Z$ and Topics $\psi$.

1: randomly permute the documents.
2: for every document $j$ do
3: for each mini-topic $S^t_j$, where $z_{js} = t \ \forall s \in S^t_j$ for some topic $t$ do
4: for every other topic $t' \neq t$ do
5: Compute $\Delta(S^t_j, t, t')$, the change in the obj. function when re-assigning $z_{js} = t' \ \forall s \in S^t_j$.
6: end for
7: Let $t^* = \arg \min_{t'} \Delta(S^t_j, t, t')$.
8: Reassign tokens in $S^t_j$ to $t^*$ if it yields a smaller obj.
9: Update topics $\psi$ and assignments $Z$.
10: end for
11: end for

We accept the move if it leads to smaller value of the objective function; that is, $\min_{t' \neq t} \Delta(S^t_j, t, t') < 0$ and update the topics $\psi$ and assignments $Z$ accordingly. Then we continue to the next mini-topic, hence the term “incremental”. Since $\psi$ and $Z$ are updated in every objective-decreasing move, we randomly permute the processing order of the documents in each iteration. This usually helps in obtaining better results in practice. Algorithm 5.3 gives the detailed description.
Note this incremental refinement step can also be applied to the word-level, where we iterate through all the word tokens, similar to the iterated conditional modes method [Besag, 1986] and the local search techniques [Dhillon and Guan, 2003] in clustering. An important difference from [Dhillon and Guan, 2003] is that here we need to accept all objective-decreasing moves, not just the single best one. However, this step has overlapping effect with the facility location based assignment algorithm and in practice, it does not work well as either a stand-alone assignment step or a refinement step. As a stand-alone assignment step, it is still quite sensitive to the initialization even though it could optimize the objective function with relatively large λ values. It still tries to assigning topic associations to word tokens based solely on one word token other than considering multiple words altogether. As a refinement step, it has limited ability to refine the topics and assignments.

On the other hand, the hierarchical structure is critical here. One noticeable thing about the proposed mini-topic based refinement step is that it will merge several mini-topics into one mini-topic and decrease the number of topics utilized for the document. This is not true for the word based refinement. Thus, the proposed Improved Word Assignment with Incremental Refinement provides a natural “split-and-merge” strategy, where the assignment step tries to split the document into multiple partitions belonging to different topics, while the refinement step attempts to merge different partitions into one exploiting the co-occurrence information. Our refinement step can also be seen as a “guided” merge strategy, where the guidance is from the improved assignments instead of considering word tokens randomly.
5.4 Experiments

In this section, we compare the algorithms proposed above with the basic algorithm and their probabilistic counterparts.

Methodology. We compare two versions of our algorithms—Improved Word Assignment (\texttt{Word}), and Improved Word Assignment with Incremental Refinement (\texttt{Word + Refine})—with the traditional Basic Batch algorithm (\texttt{Basic}), the collapsed Gibbs sampler (CGS) [Griffiths and Steyvers, 2004], the standard variational inference algorithm (VB) [Blei \textit{et al.}, 2003], the spectral algorithm (Spectral) [Anandkumar \textit{et al.}, 2012], the orthogonal joint diagonalization (JD) [Podosinnikova \textit{et al.}, 2015], the tensor power method (TPM) [Podosinnikova \textit{et al.}, 2015] and the Anchor method\textsuperscript{4} [Arora \textit{et al.}, 2013].

5.4.1 Synthetic Documents

Our first set of experiments is on simulated data. Due to a lack of ground truth data for topic modeling, we benchmark on synthetic data. We train all algorithms on the following data sets. (A) documents sampled from an LDA model with $\alpha = 0.04, \beta = 0.05$, with 20 topics and having vocabulary size 2000. Each document has length 150. (B) documents sampled from an LDA model with $\alpha = 0.02, \beta = 0.01$, 50 topics and vocabulary size 3000. Each document has length 200.

For the collapsed Gibbs sampler, we collect 10 samples with 30 iterations of thinning after 3000 burn-in iterations. The variational inference algorithm runs for 100 iterations. The Basic algorithm runs towards convergence. The Word algorithm replaces

\textsuperscript{4}All the codes used are provided by the authors.
basic word assignment with the improved word assignment step within the batch algorithm, and \texttt{Word+Refine} further alternates between improved word and incremental topic refinement steps. The \texttt{Word} and \texttt{Word+Refine} are run for 20 and 10 iterations, respectively. All the algorithms are initialized by randomly assigning each word to one of the topics, whenever applicable. For \texttt{Basic}, \texttt{Word} and \texttt{Word+Refine}, we run experiments with $\lambda \in \{6, 7, 8, 9, 10, 11, 12\}$, and the best results are presented if not stated otherwise. In contrast, the true $\alpha, \beta$ parameters are provided as inputs to the CGS, VB, Spectral, JD and TPM algorithms. We note that we are heavily handicapped by this setup, since these algorithms are designed specifically for data from the LDA model. Here, we consider both the latent Dirichlet allocation moments [Anandkumar et al., 2012] and the discrete independent analysis cumulants (DICA) [Podosinnikova et al., 2015] for the Spectral, JD and TPM algorithms.

\textbf{Objective optimization.} First we look at the abilities of the proposed algorithms on optimizing the objective function (5.7). Table 5.1 shows the resulting objective function values optimized with all three proposed algorithms. We can see that the \texttt{Word} algorithm significantly reduces the objective value when compared with the \texttt{Basic} algorithm, and the \texttt{Word+Refine} algorithm reduces further. As pointed out in [Yen et al., 2015] in the context of other SVA models, the \texttt{Basic} algorithm is very sensitive to initializations and $\lambda$ values. With relatively large $\lambda$ values, it almost has no ability to optimized it further with random initialization. However, this is not the case for the \texttt{Word} and \texttt{Word+Refine} algorithms and they are quite robust to initializations, thanks to the facility location algorithm. From the objective values, the improvement from \texttt{Word+Refine} to \texttt{Word} seems to be marginal. But we will show
<table>
<thead>
<tr>
<th>Objective Value</th>
<th>SynthA</th>
<th>SynthB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>5074939.616</td>
<td>5453889.128</td>
</tr>
<tr>
<td>Word</td>
<td>4055091.759</td>
<td>3790071.752</td>
</tr>
<tr>
<td>Word+Refine</td>
<td>3975536.098</td>
<td>3609980.107</td>
</tr>
</tbody>
</table>

Table 5.1: The resulting combinatorial topic modeling objective function values optimized with Basic, Word, and Word+Refine algorithms with $\lambda = 10$ (smaller the better).

in the following that the incorporation of the incremental refinement is crucial for learning good topic models.

**Assignment accuracy.** We now consider the accuracy of the topic assignments for the word tokens. Both the Gibbs sampler and our proposed algorithms provide word-level topic assignments. Thus we can compare the training accuracy of these assignments, which is shown in Figure 5.5. The result of the Gibbs sampler is given by the highest among all the samples selected. The accuracy is shown in terms of the normalized mutual information (NMI) score and the adjusted Rand index (ARand), which are both in the range of $[0,1]$ and higher values represents better accuracy. Both metrics are standard evaluation metrics for clustering accuracy.

We can see that the Basic algorithm, despite the similarity with the Gibbs sampler and has the same assignment update strategy as with existing SVA algorithms, performs poorly. It has NMI scores less than 0.05 with random assignment initialization. This shows that the Basic algorithm is very sensitive to the initialization and the $\lambda$ value. Basically, it tries to optimize an approximation of the objective and can not optimize the objective much with relatively large $\lambda$. Even with careful assignment initializations, it can only achieve NMI scores of 0.4 to 0.5. Unlike Basic, the Word algorithm greatly boosts the assignment accuracy with NMI scores of 0.7
Figure 5.5: Comparisons of the normalized mutual information (NMI) scores and adjusted rand index (ARand) for the topic assignments of word tokens with the Basic, Word, and Word+Refine algorithms for Synthetic A (top) and Synthetic B (bottom) datasets with 5000 documents (higher the value, better the accuracy). The results of CGS are shown in dashed line.
to 0.8, which shows the importance of the better assignment algorithm. Also, there is no explicit dependence on the topic assignment initializations. With further help from incremental refinement, we achieve NMI scores of 0.8 to 0.9, which match or marginally exceed the performance of the Gibbs sampler for a wide range of \( \lambda \). This validates the usefulness of the refinement step and manifests the insensitivity with respect to the \( \lambda \) values.

**Topic reconstruction error.** Now we look at the reconstruction error between the true topic-word distributions and the learned distributions. In particular, given a learned topic matrix \( \hat{\psi} \) and the true matrix \( \psi \), we use the Hungarian algorithm [Kuhn, 1955] to align topics, and then evaluate the \( \ell_1 \) distance between each pair of topics. Table 5.2 presents the mean reconstruction errors per topic of different learning algorithms for varying number of documents. As a baseline, we also include the results from the \( K \)-means algorithm with KL-divergence [Banerjee et al., 2005] where each document is assigned to a single topic. This can also be considered as an extreme case of our objective (5.7) as \( \lambda \to \infty \).

Among the three proposed algorithms, similar to the situation in assignment accuracy, the **Basic** algorithm performs the worst in all the data settings, even worse than the \( K \)-means algorithm. This is due to its inability to optimize the objective with relatively large \( \lambda \) values, and the topic matrix is still very similar to the random initialization. The **Word** algorithm itself has already reduced the error by more than 45% from the **Basic** algorithm. But it still cannot compete with the probabilistic approaches. Finally, the topic refinement step provides a significant improvement, which helps to reduce the \( \ell_1 \) error at least 60% from the **Word** algorithm only. Among the probabilistic approaches, variational inference method perform the worst, possibly
Table 5.2: Comparisons of topic reconstruction $\ell_1$ errors of different algorithms for the synthesized datasets with varying number of documents. Here, “KMeans-KL” is the $K$-means algorithm with KL-divergence and LDA moments are used for method-of-moments style algorithms, namely Spectral, JD, and TPM. Top two results are shown in bold.
due to its independence assumption of the posterior distributions. The only difference among Spectral, JD, and TPM is the algorithm used to recover the topic matrix, and these results again illustrate the importance of better algorithms.

The Gibbs sampler has the lowest $\ell_1$ error on smaller corpora, where Word+Refine and Anchor come next. However, for the larger corpora, the sampler needs to run much longer to reach a lower $\ell_1$ error, and can not compete with Word+Refine and Anchor for even 3000 iterations. These manifest the consistency of the our Word+Refine algorithm with respect to the size of the document collection. We again want to emphasize here that CGS, VB, Spectral, JD and TPM are given the true parameters as input.

As observed above, the Gibbs sampler can easily become trapped in a local optima area and needs many iterations to get out of it on large data sets, which can be seen from Figure 5.6. It takes the sampler more than 3000 steps to jump to the lower-error area. Since our algorithm outputs the topic assignment $Z$, we can use this assignment as initialization to the sampler. In Figure 5.6, we show the evolution of topic reconstruction $\ell_1$ error initialized with the Word+Refine optimized assignment for only 3 iterations with varying values of $\lambda$. With these semi-optimized initializations, we observe more than a 5-fold speed-up compared to random initialization. For example, it takes about 500 steps to reach an average $\ell_1$ error of 0.05 initialized using the assignment learned using Word+Refine algorithm with $\lambda = 6$, while it takes about 5000 Gibbs steps to reach the same accuracy with random initialization. In addition, this speed-up happens with no special choice of $\lambda$. Finding the perfect $\lambda$ value may not be easy, but with this result we can simply use a reasonably large $\lambda$ value to get the

5The $\ell_1$ distance used in [Podosinnikova et al., 2015] is the normalized version, which is half of what we report here.
Figure 5.6: The evolution of topic reconstruction $\ell_1$ errors of Gibbs sampler with different initializations: “Random” means random initialization, and “lambda=6” means initializing with the assignment learned using Word+Refine algorithm with $\lambda = 6$ for only 3 iterations.

semi-optimized assignment initialization for the sampler which will largely improve the efficiency of the sampler.

To provide a comprehensive analysis, we also consider generating documents with varied number of word tokens similar to [Podosinnikova et al., 2015] in addition to the fixed document-length synthetic datasets. The document length is sampled from a Gaussian distribution with the mean same as the fixed document length and standard deviation 10 and 30 respectively. In addition to the results with the LDA moments [Anandkumar et al., 2012] reported in Table 5.2, we also consider the Spectral, JD, TPM with the discrete independent analysis cumulants (DICA) [Podosinnikova et al., 2015]. Table 5.3 presents the mean reconstruction errors with their standard deviations for 10K documents. Intuitively, the performance of the algorithms should
not be impacted by the variation of document lengths in the collection. And this is indeed true for most of the LDA based approaches, including our Word+Refine algorithm. However, this is not the case for the DICA based methods, especially the JD and TPM method. For example, with synthetic dataset A, the TPM method produces an average $\ell_1$ error of 1.717 with fixed document-length collection, but an error of 0.077 with varied document-length collection. JD produces a little better average $\ell_1$ error with DICA than LDA moments for synthetic dataset B with both fixed and varied document-length, but with significantly large standard deviations (0.26 for DICA v.s. 0.02 for LDA). These highlight the importance of the appropriateness of the model assumption to the observed data.

**Running Time.** For our current implementation, an iteration of the Refine step is roughly equivalent to one Gibbs step, while an iteration of the Word step is roughly equivalent to two Gibbs iterations. Since one typically runs thousands of Gibbs iterations as shown in Figure 5.6 (while ours runs in 10 iterations even on very large data sets, yielding a running time equivalent to approximately 30 Gibbs iterations$^6$), we can observe several orders of magnitude improvement in speed by our algorithm. Further, running time could be significantly enhanced by noting that the Word algorithm trivially parallelizes.

### 5.4.2 Real Documents

In this section, we consider two real-world data sets with different properties: a random subset of the Enron emails (8K documents, vocabulary size 5000), and a subset

---

$^6$We observe better performance with two Refine steps instead of one in the first few iterations.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SynthA same length</th>
<th>SynthA varied length</th>
<th>SynthB same length</th>
<th>SynthB varied length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>0.220 (0.428)</td>
<td>0.283 (0.471)</td>
<td>0.504 (0.676)</td>
<td>0.363 (0.559)</td>
</tr>
<tr>
<td>VB</td>
<td>0.059 (0.010)</td>
<td>0.317 (0.543)</td>
<td>0.392 (0.663)</td>
<td>0.401 (0.662)</td>
</tr>
<tr>
<td>Spectral-DICA</td>
<td>0.557 (0.430)</td>
<td>0.475 (0.268)</td>
<td>0.707 (0.495)</td>
<td>0.635 (0.461)</td>
</tr>
<tr>
<td>Spectral-LDA</td>
<td>0.112 (0.020)</td>
<td>0.120 (0.041)</td>
<td>0.314 (0.199)</td>
<td>0.274 (0.170)</td>
</tr>
<tr>
<td>JD-DICA</td>
<td>0.259 (0.336)</td>
<td>0.153 (0.007)</td>
<td>0.115 (0.270)</td>
<td>0.113 (0.267)</td>
</tr>
<tr>
<td>JD-LDA</td>
<td>0.099 (0.005)</td>
<td>0.102 (0.006)</td>
<td>0.161 (0.022)</td>
<td>0.162 (0.022)</td>
</tr>
<tr>
<td>TPM-DICA</td>
<td>1.717 (0.091)</td>
<td>0.077 (0.004)</td>
<td>0.099 (0.005)</td>
<td>0.033 (0.002)</td>
</tr>
<tr>
<td>TPM-LDA</td>
<td>0.102 (0.018)</td>
<td>0.103 (0.019)</td>
<td>0.105 (0.028)</td>
<td>0.111 (0.023)</td>
</tr>
<tr>
<td>Anchor</td>
<td>0.080 (0.005)</td>
<td>0.080 (0.004)</td>
<td>0.105 (0.271)</td>
<td>0.051 (0.004)</td>
</tr>
<tr>
<td>Word+Refine</td>
<td>0.197 (0.440)</td>
<td>0.059 (0.003)</td>
<td>0.197 (0.440)</td>
<td>0.059 (0.003)</td>
</tr>
<tr>
<td>CGS</td>
<td>0.276 (0.556)</td>
<td>0.160 (0.421)</td>
<td>0.276 (0.556)</td>
<td>0.160 (0.421)</td>
</tr>
</tbody>
</table>

Table 5.3: Comparisons of topic reconstruction errors of different algorithms for the synthesized datasets, where the number inside the parentheses is the standard deviation. Here, “same length” means all the generated documents have the same number of word tokens, while “varied length” means the generated documents have varied number of word tokens. The “same length” and “varied length” documents share the same mean length. Also, “LDA” is the latent Dirichlet allocation moments [Anandkumar et al., 2012], and “DICA” is the discrete independent component analysis cumulants [Podosinnikova et al., 2015].
Table 5.4: Comparisons of topic reconstruction $\ell_1$ errors of different algorithms on the semi-synthetic documents generated from the Enron and NYTimes datasets. Here, LDA moments are used for method-of-moments style algorithms, namely *Spectral*, *JD*, and *TPM*.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>semiEnron</th>
<th>semiNYTimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>1.881</td>
<td>1.890</td>
</tr>
<tr>
<td>Word</td>
<td>0.529</td>
<td>0.721</td>
</tr>
<tr>
<td>VB</td>
<td>0.375</td>
<td>0.468</td>
</tr>
<tr>
<td>Spectral</td>
<td>0.340</td>
<td>0.510</td>
</tr>
<tr>
<td>JD</td>
<td>0.219</td>
<td>0.325</td>
</tr>
<tr>
<td>TPM</td>
<td>0.215</td>
<td>0.328</td>
</tr>
<tr>
<td>Anchor</td>
<td>0.199</td>
<td>0.313</td>
</tr>
<tr>
<td>Word+Refine</td>
<td>0.201</td>
<td>0.297</td>
</tr>
<tr>
<td>CGS</td>
<td>0.202</td>
<td>0.283</td>
</tr>
</tbody>
</table>


**Semi-synthetic corpora.** We consider semi-synthetic corpora following [Arora et al., 2013]. Specifically, we generate semi-synthetic corpora from topics trained with $K = 50$ from Enron and NYTimes using the Gibbs sampler for 3000 steps, with document lengths set to 200. This setup gives a clear expected advantage to the performance of the Gibbs sampler; the main interest here is in comparisons to other methods. In view of the unstable performance of the DICA methods as shown in Table 5.3, we only consider LDA-moments based spectral methods here. Table 5.4 shows the mean topic reconstruction $\ell_1$ errors for 10K documents. The Gibbs sampler has the lowest reconstruction $\ell_1$ error as expected. And similar to the situations in the synthetic case, the *Word+Refine* and *Anchor* methods perform quite well and consistently. Figure 5.7 shows the density of the topic reconstruction $\ell_1$ error for the
Figure 5.7: Comparisons of the densities of the topic reconstruction $\ell_1$ error on the semi-NYTimes documents for the TPM, Anchor, Word+Refine, and CGS algorithms.

top few methods. Although CGS delivers the lowest average error, it also have the widest error range. On the contrary, our proposed Word+Refine shows the smallest error range while still delivering very good average error. Again, the true parameters are provided as input to the probabilistic approaches when applicable.

**Predictive performance.** We consider the held-out word log-likelihood for predictive performance: fifty percent of the word tokens of the test documents are used to inference the document-topic distribution $\theta_j$s, and then are used to compute the word log-likelihood of the other fifty percent which forms the prediction set. This is similar to the document completion evaluation metric in [Wallach et al., 2009]. Computing directly from the predictive distribution requires computationally demanding sampling procedures. As pointed out in [Ranganath et al., 2015], it only allows testing of a smaller number of documents. In addition, we also consider a discrete version...
Table 5.5: The discrete and held-out word log-likelihood on new documents for Enron ($K = 100$ topics) and NYTimes ($K = 100$ topics) datasets with fixed $\alpha = 0.01$ and varying $\beta$ values using both Gibbs sampler and our proposed \textit{Word+Refine} algorithm. See text for details.

<table>
<thead>
<tr>
<th></th>
<th>$\beta = 0.1$</th>
<th>$\beta = 0.01$</th>
<th>$\beta = 0.001$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>discrete</td>
<td>held-out</td>
<td>discrete</td>
</tr>
<tr>
<td>CGS</td>
<td>-5.932</td>
<td>-8.604</td>
<td>-5.484</td>
</tr>
<tr>
<td>NYTimes</td>
<td>$\beta = 0.1$</td>
<td>$\beta = 0.01$</td>
<td>$\beta = 0.001$</td>
</tr>
<tr>
<td></td>
<td>discrete</td>
<td>held-out</td>
<td>discrete</td>
</tr>
<tr>
<td>Word+Refine</td>
<td>-6.105</td>
<td>-10.666</td>
<td>-5.941</td>
</tr>
</tbody>
</table>

Figure 5.8: The evolution of the held-out word log-likelihood of Gibbs sampler with different initializations for the Enron dataset: “Random” means random initialization, and “W+R Init” means initializing with the assignment learned using the \textit{Word+Refine} algorithm for only 3 iterations.
of the word log-likelihood similar to the evaluation of $k$-means, where we directly infer the word-topic assignments $Z$ using the Word algorithm and compute the word likelihood based on the learned topic and the inferred topic assignments. Note this is not actually a “test” likelihood since we infer the assignments by looking at the test documents. We include this result only for possible alignment with traditional clustering problems.

Table 5.5 shows the results on the Enron and NYTimes datasets with $K = 100$ topics. We can see that our approach excels in the discrete log-likelihood while lags in the held-out log-likelihood. Note here we tune the $\lambda$ value such that the resulting number of topics per document is comparable to that of the sampler. This would put our method at disadvantage since smaller number of topics per document is needed for comparable performance. This is because our method treat every including topic in each document almost equally, while the probabilistic approaches put different weights on different topics. Therefore, in order to include those “small-weight” topics from the probabilistic approaches, our method needs to put a noticeable amount of word tokens to those topics which may result worse parameter inference results.

With a higher $\lambda$ value, we can get comparable held-out predictive performance as the sampler. For example, with $\beta = 0.01$, the held-out log-likelihood of our method comes at -10.784 (v.s. -10.870 of the sampler) for Enron and -11.449 (v.s. -11.431 of the sampler) for NYTimes.

Similar to the topic reconstruction case, we also consider the speed-up of the sampler in term of the held-out word log-likelihood initialized with the topic assignments optimized using our proposed Word+Refine method for 3 iterations only. This costs the time of less than that of 10 Gibbs steps. Figure 5.8 shows the results. We
Table 5.6: Some example topics with top 10 words learned from NYTimes dataset using both the Gibbs sampler and our proposed Word+Refine algorithm. The topics are aligned using the Hungarian algorithm [Kuhn, 1955].

again observe quite a speed-up over the random initialization. We can see that the held-out word log-likelihood quickly surpasses that from the random initialization, and reaches a log-likelihood in less than 1000 steps which requires 3000 steps with random initialization. Combining with results shown in Figure 5.6, we can see that the proposed algorithm can serve as a nice initialization strategy for the sampler with no special choice of $\lambda$ values, which would substantially improve the sampler’s mixing time, in addition to its service as a well-performed stand-alone inference algorithm for topic models with right $\lambda$ value. It also provides a quick solution and more flexibility exploring different granularities with varying $\lambda$ values. Table 5.6 further shows some sample topics generated by CGS and our Word+Refine method.
5.5 Conclusion

In this chapter, we have considered an asymptotic treatment of the LDA model, which has laid the groundwork for a combinatorial optimization view of topic modeling as an alternative to the standard probabilistic framework. The traditional SVA algorithm performs poorly when compared quantitatively to probabilistic approaches. However, using ideas from facility location and incremental refinement, we obtain an algorithm that compares favorably, while being efficient and robust to initializations and parameter selection. Moreover, the sampler’s mixing time improves substantially when initialized using our method.
Chapter 6: Conclusions and Future Works

Probabilistic approaches, particularly Bayesian models, are flexible from a modeling perspective. But inference in large-scale probabilistic models remains a challenge. Existing sampling-based and variational inference techniques still leave some applications out of reach. Small-variance asymptotics provide an emerging technique for obtaining combinatorial models and designing scalable algorithms from rich probabilistic models.

In this dissertation, we have considered the small-variance asymptotics on three standard learning problems: infinite mixture models with exponential family distributions, infinite hidden Markov models, and latent Dirichlet allocation topic models. The goal is to obtain non-probabilistic formulations inspired by those Bayesian models, and lay the groundwork for a combinatorial optimization framework as an alternative to the standard probabilistic framework. Small-variance asymptotics provides a natural way to obtain an underlying objective function, using the $K$-means connection to Gaussian mixtures as an analogy. We have developed combinatorial algorithms that compare favorably with probabilistic approaches, while being efficient. In Chapter 5, we also showed that the algorithm is robust to initializations and parameter selection. Moreover, the sampler’s mixing time improves substantially when initialized using our method.
Small-variance asymptotics have been quite successful recently, applications include clustering [Kulis and Jordan, 2012], feature learning [Broderick et al., 2013], evolutionary clustering [Campbell et al., 2013], Markov jump processes [Huggins et al., 2015], infinite SVMs [Wang and Zhu, 2014], and hierarchical clustering methods [Lee and Choi, 2015]. However, almost all the algorithms proposed in the SVA literature share the same greedy local assignment step as in Algorithm 3.1, which has been shown in Chapter 5 to succeed only under certain circumstances and often fails to minimize the objective greatly under random initialization. We have discussed several improvements in Section 3.4 and presented some preliminary quantitative results. A split-merge framework will be most effective, but a good split method is critical. We could apply the facility location algorithm described in Section 5.3 as a potential cluster assignment method with all the utilized cluster centroids and all the data points as potential cluster means. But this would introduce quadratic time complexity per iteration. How to design efficient split and merge steps would be an interesting question to explore. Distributed implementation of the combinatorial topic modeling algorithm would be another interesting question to explore. The facility location based assignment step is embarrassingly parallel, since the assignment for word tokens in each document is independent. But the incremental refinement part is essentially sequential since it relies on the exact objective function value change. We have tried the “delayed” parameter updates similar to [Agarwal and Duchi, 2011], where we use the “stale” topic matrix information sometimes for computation. For limited delay, the algorithm seems work fine, but would deteriorate once the parameter information gets older. Some modification of the refinement step may be needed to make it insensitive to the delay.
References


