Benders' Decomposition for Solving the Lagrangian Relaxation of Probabilistic Graphical Models

BENDERS' DECOMPOSITION FOR SOLVING THE LAGRANGIAN RELAXATION OF PROBABILISTIC GRAPHICAL MODELS

A Dissertation Presented

by

Ji Ah Lee

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2024

Department of Mathematics and Statistics

BENDERS' DECOMPOSITION FOR SOLVING THE LAGRANGIAN RELAXATION OF PROBABILISTIC GRAPHICAL MODELS

A Dissertation Presented

by

Ji Ah Lee

Approved as to style and content by:

Patrick Flaherty, Chair

Anna Liu, Member

Markos A. Katsoulakis, Member

Daniel Sheldon, Member

Tom Braden, Graduate Program Director
Department of Mathematics and Statistics

DEDICATION

To my family, friends, and mentors for all the support and help.

ACKNOWLEDGMENTS

First and foremost, I have to thank my advisor, Dr. Patrick Flaherty. I could not have undertaken this journey without him. Throughout my Ph.D. program, he has been the best mentor one can ask for. He shares his stories and wisdom to encourage me and guide me when I am let down or even feel lost, either academically or personally. With invaluable patience, he endlessly provides knowledge and expertise. I am truly grateful for all of his help to become a researcher.

I also appreciate Professor Anna Liu, Professor Markos Katsoulakis, and Professor Daniel Sheldon for being my committee members, reviewing the work and providing invaluable feedback.

I would like to extend my sincere thanks to my fellow labmates, especially Harsh Dubey and Vishal Sarsani, for always being there to help me with research and for moral support.

Furthermore, I would like to thank Kaitlyn O'Konis, who was highly dedicated to helping me throughout the process, from registering to filing the dissertation. Thanks should also go to the professors and staffs from the department for their support throughout my graduate studies.

Lastly, I would be remiss in not mentioning my family. Their beliefs in me started this journey and kept me going.

The research for my doctoral dissertation is based upon financial support from the University of Massachusetts Amherst.

ABSTRACT

BENDERS' DECOMPOSITION FOR SOLVING THE LAGRANGIAN RELAXATION OF PROBABILISTIC GRAPHICAL MODELS

FEBRUARY 2024

Ji Ah Lee

BA., SMITH COLLEGE

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Patrick Flaherty

The goal of this thesis is to develop a novel inference method for graphical model inference problems. Variational inference methods have been widely and successfully used for many computationally challenging statistical inference problems. However, the stopping criterion for variational inference algorithms is that the change in the objective function is small. It is unknown whether the algorithm has truly converged. Benders' decomposition is an optimization method that has the capability to provide upper and lower bounds on the optimal value and thus give a certificate of optimality for variational inference problems. Utilizing Benders' decomposition, we propose a graphical model inference method that provides an optimality guarantee.

In the first chapter, we give an introduction of the paper - what motivates our research, our contributions, and organizational overview of the paper. In the second chapter, we review the general context of the problem, comparable inference methods and techniques that we utilize in the development of the method.

In the third chapter, we try to develop an algorithm to solve a maximum a-posteriori (MAP) clustering problem under the Gaussian mixture model. We formulate the problem as a mixed-integer nonlinear problem (MINLP), turning it into an optimization problem that can accommodate side constraints. We then prove that the problem satisfies the conditions of Generalized Benders Decomposition (GBD). We explore three different derivations of GBD for the problem: (1) exploiting all explicit constraints, (2) only exploiting constraints over $\boldsymbol{\pi}$, and (3) deriving a simple linear form of optimal cuts via Taylor series. None of these approaches, yet, result in a successful algorithm; the flaws of each approach are discussed.

In the fourth chapter, we extend the idea from the third chapter, and develop a maximum a-posteriori (MAP) inference method in general Bayesian factor models. Previous study has shown that MAP assignment problem can be relaxed via Lagrangian or linear programming, and its dual form yields a computationally efficient inference algorithm. The question remains is to add back which constrains in to the relaxed problem. We use generalized Bender decomposition to sequentially adds the most optimal constraints to the fully relaxed dual problem. The method guarantees of ϵ-convergence by tightening the gap between upper and lower bounds of the problem. We also show which condition has to be met for the certificate of optimality to be at the global scale; it could be at the local level if the condition is not met. Two Bayesian models, Bayesian Gaussian mixture model (BGMM) and latent Dirichlet allocation (LDA) model, are explored using the proposed method. For each model, we implement the algorithm on standard data sets and compare the results to those of variational Bayes and Gibbs sampler. Our proposed method outperforms variational Bayes and Gibbs sampler on some of the data sets in terms of achieving the higher log(MAP) value, and always end with guaranteed optimality, which other two methods do not provide. We then discuss a few details to consider when applying the method.

Lastly, we contribute to the development of dynamic programming algorithms and related experiments in hierarchical clustering for exact inference.

CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Graphical models can describe complex dependencies among the variables with their compact description. The problem of inference in graphical models is to study the underlying state of the variables in the models. However, unlike their concise description, probabilistic inference is often computationally intractable. To find the maximum probability assignment of a graphical model (the MAP assignment) is NP-hard. Finding the MAP assignment is still challenging even if the local functions depend only on two variables. This difficulty has led to an extensive research.

Many methods used to find the MAP assignment can be seen as a specific combination of a relaxation of the problem and a search algorithm for locating a local or global minimum. Locally optimal and heuristic methods lack optimality guarantees or bounds on solution quality that indicate the potential for further improvement. The estimate produced by the *expectation-maximation (EM) algorithm* does not guarantee for the global optimality. While Balakrishnan et al. (2017) showed that the global optima of a mixture of well-separated Gaussians may have a relatively large region of attraction, Jin et al. (2016) showed that inferior local optima can be arbitrarily worse than the global optimum. *Simulated annealing* methods are theoretically guaranteed to converge to a global optimum of a nonlinear objective. However, choosing the annealing schedule for a particular problem is challenging and its guarantee of global optimality only exists in the limit of the number of steps - there is no general way to choose the annealing schedule or monitor convergence (Andrieu et al., 2003).

On the other hand, an optimization framework affords bounds that establish the degree of suboptimality for any feasible solution. Methods that use an optimization framework with a relaxation of the problem, though, do not guarantee if the solution is globally optimal, and even if they do, they only focus on certain types of graphical models. Johnson et al. (2007) propose a general framework for MAP estimation in graphical models using *Lagrangian relaxation* techniques with a new class of multi-scale relaxations. The method may reduce or eliminate the duality gap but is only developed for discrete variable and Gaussian graphical models where the problem is convex decomposable. Sontag et al. (2008) propose a framework for cluster-based *Linear programming* relaxations that tightens the problem fast enough to be used in practices. Yet, it focuses on pairwise Markov random fields rather than the general graphical models.

Besides the benefit of providing a certificate of global optimality, optimization methods has seen impressive improvements in the size of the problems that can be handled. Bertsimas and King (2016) presents a linear regression as a mixed-integer quadratic program with its stability and robustness and solved for samples of size $n \sim 1,000$. Best subset selection in the context of regression is typically approximated as a convex problem with the ℓ_1 norm penalty, but Bertsimas et al. (2016) solve it using the nonconvex ℓ_0 penalty for thousands of data points. In unsupervised learning, Bandi et al. (2019) use mixed-integer optimization to learn parameters for the Gaussian mixture model and shows that its optimization-based approach outperforms the EM algorithm.

For these reasons, we are motivated to develop an innovative method to infer the MAP assignment for *general* graphical models based on the *optimization* framework. Our proposed method guarantees of ϵ-convergence by tightening the gap between upper and lower bounds of the problem. We identify the exact condition when the

method yields a globally optimal solution for any general graphical models. We also provide tips for efficient application of the method via a thorough discussion.

The organization of the thesis is as follows. Chapter 2 reviews the general context of the problem, comparable inference methods to our proposed method and two key techniques that we use in the development of the method. Our attempts to apply the simpler version of the method to the Gaussian mixture model are presented in Chapter 3. Each attempt is illustrated in details and discusses why it is not successful. Chapter 4 extends the idea from the Chapter 3, and presents a MAP inference method in general Bayesian factor models. It also discusses the method's condition to be met for a globally optimal solution, contributions, applications and a few things to be considered before applications. In addition to the MAP inference method, Chapter 5 introduces dynamic programming algorithms in hierarchical clustering for exact inference, and shows my contribution in to the development of the algorithms and their applications in cancer genomics.

CHAPTER 2

LITERATURE REVIEW

2.1 Probabilistic Graphical Model

A probabilistic graphical model (PGM) is an encoded probability distribution in which the variables are represented as nodes and the dependence relations as edges connecting nodes. The graph is a concise representation of a set of conditional independencies that hold in distribution. It also defines a skeleton for compactly describing a high-dimensional distribution. The independence properties of the distribution allows us to divide the distribution into smaller factors. We can then define the overall joint distribution as a product of these smaller factors.

The probabilistic methods have been in use widely since late 1980s due to two key factors (Koller and Friedman, 2009). The first factor was a series of seminal theoretical developments, especially the development of the Bayesian network framework by Judea Pearl (Pearl, 1988). The second major factor was the development of large-scale, highly effective expert systems based on this framework. Applications of PGMs include causal inference, speech recognition, computer vision, analysis of genomic data, and protein structure.

2.1.1 Bayesian network

One family of probabilistic graphical models is Bayesian network (Pearl, 1988). The model consists of a directed acyclic graph and a conditional probability table for each node given its parents:

$$P(X_i|\text{pa}(X_1)). \tag{2.1}$$

The joint probability represented by Bayesian network is

$$P(X_1, X_2, \ldots, X_n) = \Pi_i P(X_i|\text{pa}(X_1)), \tag{2.2}$$

where $\text{pa}(X_i)$ is the set of parents of node X_i. The nodes could be observed variables, latent variables, unknown parameters or hypotheses, in the Bayesian perspective. One advantage of Bayesian network is that one can integrate statistical data with expert knowledge easily. Another one is that Bayesian network has possibility of working with missing data. Bayesian networks perform well with small data sets using canonical models (Oniśko et al., 2001) or even with not very precise probabilities (Uusitalo, 2007).

Three inference tasks are primarily performed over Bayesian networks. The first task is to infer unobserved variables. A Bayesian network can be used to answer probabilistic queries about its variables and their relationships because it is a complete model. The most common approximate inference algorithms are stochastic Markov chain Monte Carlo (MCMC) simulation and variational methods. The second task is to learn parameters. In order to fully represent the joint probability distribution, the probability distribution of each node X given X's parents has to be specified. These conditional distributions often include unknown parameters that must be estimated from data by the maximum likelihood method. Unobserved variables yet often make direct maximization of the likelihood complicated. A classical approach to this problem is the expectation-maximization (EM) algorithm (Dempster et al., 1977). The third task is to learn the network structure. The task of identifying the network structure without expert knowledge has to be learned from data and may be too complex. Common methods of structural learning include a recovery algorithm (Rebane and Pearl, 2008) and an optimization-based search.

2.2 Related Work

For a general problem, consider a joint density of latent variables $\mathbf{z}$ and observations $\mathbf{x}$:

$$p(z, x) = p(z)p(x|z). \tag{2.3}$$

The latent variables help shape the distribution of the data in Bayesian models. A Bayesian model draws the latent variables from a prior density $p(z)$ and ties them to the observations through the likelihood $p(x|z)$. Inference in a Bayesian model involves both conditioning on data and calculating the posterior $p(z|x)$. In complicated Bayesian models, this process often calls for approximate inference (Blei et al., 2017)

2.2.1 Variational Bayes

Variational Bayes is a machine learning method that approximates probability densities (Jordan et al., 1999; Wainwright and Jordan, 2008). It is widely used to approximate posterior densities for Bayesian models, as an alternative to MCMC sampling.

The main idea of variational inference is to use optimization. First, we postulate a family of approximate densities Q, a set of densities over the latent variables. We then focus on finding the member of the family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior (Barber, 2012),

$$q^*(\mathbf{z}) = \arg\min_{q(\mathbf{z})\in Q} \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})). \tag{2.4}$$

We at last approximate the posterior with the optimized member of the family found. Variational inference basically turns the inference problem into an optimization problem. One key idea underlying this approach is to select Q that is flexible enough to capture a density close to $p(z|x)$, but also simple enough for efficient optimization.

Variational inference can benefit from techniques like stochastic optimization (Robbins and Monro, 1951; Kushner and Yin, 1997) and distributed optimization (Welling

and Teh, 2011; Ahmed et al., 2012) thanks to its optimization framework. Compared to MCMC, another popular method used to approximate posterior densities, variational inference tends to run faster and easier to scale to large data. Thus, it has been used to solve large-scale problems such as those in document analysis, computational neuroscience, and computer vision.

2.2.2 Markov Chain Monte Carlo

For decades, the dominant paradigm for approximate inference has been MCMC (Hastings, 1970; Gelfand and Smith, 1990). In MCMC, we first set up an ergodic Markov chain on z whose stationary distribution is the posterior $p(z|x)$. We then sample from the chain in order to collect samples from the stationary distribution. We at last approximate the posterior through an estimation with the samples collected.

Although MCMC techniques are typically more computationally demanding than variational inference, they guarantee of producing (asymptotically) exact samples from the target density (Robert and Casella, 2004). MCMC sampling techniques have become a crucial tool for the modern Bayesian statistician. Notable innovations include the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), the Gibbs sampler (Geman and Geman, 1984) and its application to Bayesian statistics (Gelfand and Smith, 1990).

2.3 Lagrangian Relaxation

One of the most effective approximation approaches has been to use relaxations of the original inference problem. Relaxation methods transform the original combinatorial problem into a constrained optimization problem. They then relax some of the constraints to divide the problem into more independent subproblems, resulting in a more tractable approximation than the original. Two closely related relaxation schemes are Lagrangian relaxation, also known as dual decomposition, (Johnson, 2008;

Komodakis et al., 2011) and linear programming relaxations (Wainwright et al., 2005; Shlesinger, 1976)

Taking a set of *complicating* constraints of a general mixed integer program up into the objective function in a Lagrangean fashion with fixed multipliers yields a Lagrangean relaxation of the original program (Geoffrion, 1974). Let the general integer linear programming problem be defined as

$$\begin{aligned} \underset{x \geq 0}{\text{minimize}} \quad & \boldsymbol{c}x \\ \text{subject to} \quad & Ax \geq \boldsymbol{b}, \\ & Bx \geq \boldsymbol{d}, \\ & x_j \text{ integer}, \\ & j \in \mathcal{I}, \end{aligned} \tag{2.5}$$

where A, B are matrices of conformable dimensions and the index set $\mathcal{I}$ denotes the variables required to be integer. The second constraint, $Bx \geq \boldsymbol{d}$, is supposed to have special structure.

Lagrangean relaxation relative to $Ax \geq \boldsymbol{b}$ and a nonnegative vector $\boldsymbol{\lambda}$ to be

$$\begin{aligned} \underset{x \geq 0}{\text{minimize}} \quad & \boldsymbol{c}x + \boldsymbol{\lambda}(\boldsymbol{b} - Ax) \\ \text{subject to} \quad & Bx \geq \boldsymbol{d}, \\ & x_j \text{ integer}, \\ & j \in \mathcal{I}. \end{aligned} \tag{2.6}$$

The relaxed problem can be solved by *Lagrangean dual problem*:

$$\underset{\lambda \geq 0}{\text{maximize}} \quad P(\lambda), \tag{2.7}$$

where $P(\lambda)$ is defined as

$$\begin{aligned} \underset{x \geq 0}{\text{minimize}} \quad & \boldsymbol{c}x + \boldsymbol{\lambda}(\boldsymbol{b} - Ax) \\ \text{subject to} \quad & Bx \geq \boldsymbol{d}, \\ & x_j \text{ integer}, \\ & j \in \mathcal{I}. \end{aligned} \tag{2.8}$$

It searches the feasible range of λ values while maximizing the result returned by the inner P problem. Each value returned by P is potentially a lower bound to the problem, and we keep the highest value of those lower bounds as the best lower bound. The key idea here is that the inner P problem can be solved far more efficiently than the original inference problem 2.5, after being partially relaxed.

2.4 Generalized Benders' Decomposition

A wide variety of nonlinear optimization problems include integer or discrete variables in addition to the continuous variables. These groups of optimization problems are known as Mixed-Integer Nonlinear Programming (MINLP) problems and appear in a range of applications. Key applications of MINLP approaches have emerged in chemical engineering and in engineering disciplines: the synthesis of grassroot heat recovery networks (Floudas and Ciric, 1989; Yee and Grossmann, 1990), the design and scheduling of multipurpose plants (Vaselenak et al., 1987; Papageorgaki and Reklaitis, 1990) and the optimal unit allocation in an electric power system (Bertsekas et al., 1983).

The coupling of the integer domain with the continuous domain along with their associated nonlinearities make the class of MINLP problems very challenging from the theoretical, algorithmic, and computational point of view (Floudas, 1995). The higher the number of binary variables is, the exponentially larger combinatorial problem

one has to face – the complexity analysis results describe the MINLP problems as NP-complete (Nemhauser and Wolsey, 1988). According to Murty and Kabadi (1987), finding a global solution to nonconvex MINLP problems is NP-hard. Despite the discouraging complexity analysis findings, substantial advancements have been made in the MINLP field. As a result, a number of algorithms have been presented and their convergence characteristics examined. These include Generalized Benders Decomposition (Geoffrion, 1972), Outer Approximation (Duran and Grossmann, 1986), and Feasibility Approach (Mawengkang and Murtagh, 1986).

2.4.1 Formulation

Benders (1962) developed a creative approach for utilizing the structure of mathematical programming problems with *complicating variables* (variables that, when temporarily fixed, make the remaining optimization problem significantly more tractable). Geoffrion (1972) generalized Benders' method to a larger class of programs in which the parametrized subproblem need not a linear program. When the integer variable of the problem is a binary variable, the subclass of the problems for which the GBD can be applied stated as (Floudas, 1995)

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & f(\boldsymbol{x},\boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = 0, \\ & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0, \\ & \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \\ & \boldsymbol{y} \in \mathcal{Y} = \{0,1\}^q \end{aligned} \tag{2.9}$$

under the following conditions:

C1: $\mathcal{X}$ is nonempty, convex and the functions $f : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}$ and $\boldsymbol{g} : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^p$ are convex for each fixed y, while the functions $\boldsymbol{h} : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^m$ are linear for each fixed y.

C2: The set $\boldsymbol{Z_y} = \{\boldsymbol{z} \in \mathbb{R}^p \mid \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0},\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{z}$ for some $\boldsymbol{x} \in \mathcal{X}\}$ is closed for each fixed y.

C3: For each fixed $y \in \mathcal{Y} \bigcap \mathcal{V}$, where $\mathcal{V} = \{\boldsymbol{y} \mid \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0},\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0},$ for some $\boldsymbol{x} \in \mathcal{X}\}$, one of the following two conditions holds:

1. the resulting problem has a finite solution and has an optimal Lagrange multiplier vector for the equalities and inequalities.
2. the value of objective function of the resulting Problem goes to $-\infty$ (is unbounded).

2.4.2 Basic Idea

The fundamental concept behind GBD is the generation of an upper bound and a lower bound on the solution of the MINLP model at each iteration. The primal problem yields the upper bound and the master problem yields the lower bound of the problem. The primal problem relates to problem 2.9 with fixed y so that the it is in the x-space only. Its solution provides the upper bound and the Lagrange multipliers associated with the equality and inequality constraints. The master problem is derived using nonlinear duality theory, and employs the Lagrange multipliers obtained in the primal problem. The solution of the master problem provides the lower bound and the next set of fixed y values to be used in the subsequent primal problem. As GBD iterates, the sequence of updated upper bounds is nonincreasing, the sequence of lower bounds is nondecreasing, and thus, sequences converge in a finite number of iterations.

2.4.3 Primal Problem

The primal problem is the MINLP Problem with fixed $\boldsymbol{y}$ variables – denoted $\boldsymbol{y}^{(j)}$ where j is the iteration index.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}, \boldsymbol{y}^{(j)}) \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^{(j)}) = 0, \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^{(j)}) \leq 0, \\ & \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n \end{aligned} \tag{2.10}$$

Due to conditions C1 and C3(1), the primal problem is convex and it provides the global solution. Since the primal problem is *more* constrained than the MINLP, the solution is an upper bound on the MINLP solution.

2.4.4 Master Problem

The derivation of the master problem in the GBD uses nonlinear duality theory and is characterized by the three main ideas: projection of problem 2.9 onto the y-space, dual representation of the constraints, and dual representation of the projection of problem 2.9 onto the y-space.

$$\begin{aligned} \underset{\boldsymbol{y}, \mu_B}{\text{minimize}} \quad & \mu_B \\ \text{subject to} \quad & \mu_B \geq \inf_{\boldsymbol{x} \in \mathcal{X}} L(\boldsymbol{x}, \boldsymbol{y}, \nu, \lambda) \quad \forall \nu, \forall \lambda \geq 0, \\ & \boldsymbol{y} \in \mathcal{Y} \end{aligned} \tag{2.11}$$

where

$$L(\boldsymbol{x}, \boldsymbol{y}, \nu, \lambda) = f(\boldsymbol{x}, \boldsymbol{y}) + \nu^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \lambda^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}). \tag{2.12}$$

The master problem is *equivalent* to problem 2.9, but written as an inner-outer optimization problem.

2.4.5 Relaxed Master Problem

We want to consider a relaxation of the master problem because the master problem has an infinite number of constraints – we do so by fixing Lagrange multipliers.

$$\begin{aligned} \underset{\boldsymbol{y},\,\mu_B}{\text{minimize}} \quad & \mu_B \\ \text{subject to} \quad & \mu_B \geq \xi(\boldsymbol{y}; \nu^{(j)}, \lambda^{(j)}) \quad j = 1, \ldots, J, \\ & \boldsymbol{y} \in \mathcal{Y} \end{aligned} \tag{2.13}$$

where

$$\xi(\boldsymbol{y}; \nu, \lambda) = \inf_{\boldsymbol{x} \in \mathcal{X}} L(\boldsymbol{x}, \boldsymbol{y}, \nu, \lambda), \quad \forall \nu, \forall \lambda \geq 0. \tag{2.14}$$

The relaxed master problem will provide a lower bound on the original problem.

2.4.6 General Algorithm

Step 1: Select an initial feasible point $\boldsymbol{y}^{(1)} \in \mathcal{Y} \cap \mathcal{V}$.

Step 2: Solve the primal problem with $\boldsymbol{y} = \boldsymbol{y}^{(1)}$.

From the solution, obtain the primal problem optimizer $\boldsymbol{x}^{(1)}$ and the optimal Lagrange multipliers $\nu^{(1)}$, $\lambda^{(1)}$.

Step 3: Set the counters $j = 1$.

Set the upper bound UBD $= v(\boldsymbol{y}^{(1)})$.

Step 4: Solve the relaxed master problem and let $(\hat{\boldsymbol{y}}, \hat{\mu}_B)$ be the optimal solution.

Set the current lower bound LBD $= \hat{\mu}_B$.

Step 5: If UBD $-$ LBD $\leq \epsilon$ then **terminate**.

Step 6: Solve the primal problem with $\boldsymbol{y} = \hat{\boldsymbol{y}}$.

Obtain optimizer $\hat{\boldsymbol{x}}$ and optimal Lagrange multipliers $\hat{\nu}$ and $\hat{\lambda}$. Update the upper bound UBD $= \min\{\text{UBD}, v(\hat{\boldsymbol{y}})\}$.

If UBD $-$ LBD $\leq \epsilon$ then **terminate**.

Otherwise, increment $j = j + 1$, set $\nu^{(j)} = \hat{\nu}$, and set $\lambda^{(j)} = \hat{\lambda}$.

Step 7: Return to Step 4 to obtain a new solution to the relaxed master problem.

CHAPTER 3

MAP ESTIMATION FOR THE GAUSSIAN MIXTURE MODEL VIA BENDERS' DECOMPOSITION

Abstract

We formulate the maximum a-posteriori (MAP) clustering problem under the Gaussian mixture model as a mixed-integer nonlinear optimization problem (MINLP). The problem formulated in such a way can accommodate side constraints and it preserves the combinatorial structure. We then prove that the problem satisfies the conditions of Generalized Benders Decomposition (GBD). The GBD is a global optimization approach to solve certain types of NLP and MINLP problems. We explore three different derivations of GBD for the problem: the first approach exploits all explicit constraints, the second approach only takes constraints over $\boldsymbol{\pi}$, and the third approach derives a simple linear form of optimal cuts via Taylor series. As none of these approaches turns out to be successful unfortunately, we also discuss flaws of each approach.

3.1 Motivation

In the application of clustering models to real data there is often rich prior information that constrains the relationships among the samples, or the relationships between the samples and the parameters. For example, in biological or clinical experiments, it may be known that two samples are technical replicates and should be assigned to the same cluster, or it may be known that the mean value for control samples is in a certain range. However, standard model-based clustering methods

make it difficult to enforce such hard logical constraints and may fail to provide a globally optimal clustering.

Locally optimal and heuristic clustering methods lack optimality guarantees or bounds on solution quality that indicate the potential for further improvement. In fact, even research on bounding the quality of heuristic solutions is scarce (Cochran, 2018). Conversely, a rigorous optimization framework affords bounds that establish the degree of suboptimality for any feasible solution, and a certificate of global optimality thereby guaranteeing that no better clustering exists.

Recent work on developing global optimization methods for supervised learning problems has led to impressive improvements in the size of the problems that can be handled. In linear regression, properties such as stability to outliers and robustness to predictor uncertainty have been represented as a mixed-integer quadratic program and solved for samples of size $n \sim 1,000$ (Bertsimas and King, 2016). Best subset selection in the context of regression is typically approximated as a convex problem with the ℓ_1 norm penalty, but can now be solved exactly using the nonconvex ℓ_0 penalty for thousands of data points (Bertsimas et al., 2016). In unsupervised learning, Bandi et al. (2019) use mixed-integer optimization to learn parameters for the Gaussian mixture model, showing that an optimization-based approach can outperform the EM algorithm. These examples demonstrate the great potential of optimization-based methods to significantly improve upon the state-of-the-art in machine learning. For these reasons we are motivated to develop a method for achieving a globally optimal solution for clustering under the Gaussian mixture model that allows for the incorporation of rich prior constraints.

3.2 Problem Setup

3.2.1 Finite mixture model

The probability density function of a finite mixture model is $p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \cdot p(\boldsymbol{y}|\boldsymbol{\theta}_k)$ where the observed data is $\boldsymbol{y}$ and the parameter set is $\boldsymbol{\phi} = \{\boldsymbol{\theta}, \boldsymbol{\pi}\}$. The data is an n-tuple of d-dimensional random vectors $\boldsymbol{y} = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T)^T$ and the mixing proportion parameter $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)^T$ is constrained to the probability simplex $\mathcal{P}_K = \{\boldsymbol{p} \in \mathbb{R}^K | \boldsymbol{p} \succeq 0 \text{ and } \mathbf{1}^T\boldsymbol{p} = 1\}$. When the component density, $p(\boldsymbol{y}|\boldsymbol{\theta}_k)$, is a Gaussian density function, $p(\boldsymbol{y}|\boldsymbol{\phi})$ is a Gaussian mixture model with parameters $\boldsymbol{\theta} = (\{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\}, \ldots, \{\boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\})$. Assuming independent, identically distributed (iid) samples, the Gaussian mixture model probability density function is $p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\pi}) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \; p\left(\boldsymbol{y}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$.

3.2.2 Generative model

A generative model for the Gaussian mixture density function is

$$\begin{aligned} Z_i &\overset{\text{iid}}{\sim} \text{Categorical}(\boldsymbol{\pi}) \quad \text{for } i = 1, \ldots, n, \\ Y_i|z_i, \boldsymbol{\theta} &\sim \text{Gaussian}(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}), \end{aligned} \tag{3.1}$$

where $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ and $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K)$. To generate data from the Gaussian mixture model, first draw $z_i \in \{1, \ldots, K\}$ from a categorical distribution with parameter $\boldsymbol{\pi}$. Then, given z_i, draw $\boldsymbol{y}_i$ from the associated Gaussian component distribution $p(\boldsymbol{y}_i|\boldsymbol{\theta}_{z_i})$.

3.2.3 MAP clustering

The posterior distribution function for the generative Gaussian mixture model is

$$p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\pi}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{\pi})p(\boldsymbol{\theta}, \boldsymbol{\pi})}{p(\boldsymbol{y})}. \tag{3.2}$$

The posterior distribution requires the specification of a prior distribution $p(\boldsymbol{\theta}, \boldsymbol{\pi})$, and if $p(\boldsymbol{\theta}, \boldsymbol{\pi}) \propto 1$, then MAP clustering is equivalent to maximum likelihood clustering. The MAP estimate for the component membership configuration can be obtained by solving the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\pi}}{\text{maximize}} \quad & \log p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\pi} | \boldsymbol{y}) \\ \text{subject to} \quad & z_i \in \{1, \ldots, K\} \ \forall i, \\ & \boldsymbol{\pi} \in \mathcal{P}_K. \end{aligned} \tag{3.3}$$

Assuming iid sampling, the objective function comprises the following conditional density functions:

$$\begin{aligned} p(\boldsymbol{y}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{z}_i) &= \prod_{k=1}^{K} \Big[(2\pi)^{-m/2} |\boldsymbol{\Sigma}_k|^{-1/2} \\ &\quad \cdot \exp\left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_k) \right) \Big]^{z_{ik}}, \\ p(\boldsymbol{z}_i | \boldsymbol{\pi}) &= \prod_{k=1}^{K} [\pi_k]^{z_{ik}}, \qquad p(\boldsymbol{\Sigma}, \boldsymbol{\pi}, \boldsymbol{\mu}) \propto 1, \end{aligned}$$

where $z_i \in \{1, \ldots, K\}$ is recast using binary encoding. In the case of one-dimensional data ($d = 1$) and equivariant components ($\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \sigma^2$), the MAP optimization problem can be written as

$$\begin{aligned} \underset{\mu, \pi, \mathbf{Z}}{\text{minimize}} \quad & \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} (y_i - \mu_k)^2 - \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k \\ \text{subject to} \quad & \sum_{k=1}^{K} z_{ik} = 1, \quad i = 1, \ldots, n \end{aligned} \tag{3.4}$$

where z_i is recast using binary encoding ($\mathbf{z} \in \mathcal{Z} = \{0, 1\}^{n \times K}$), $\mu \in \mathcal{M} := \{\mu | M_k^L \leq \mu_k \leq M_k^U, \ \forall k\} \subseteq \mathbb{R}^K$, and $\pi \in \Delta_K := \{\pi | 0 \leq \pi_k \leq 1, \forall k \text{ and } \sum_k \pi_k = 1\} \subseteq \mathbb{R}^K$. For simplicity, the rest of Chapter 3 focuses on the problem 3.4.

3.3 Applicability of Generalized Benders' Decomposition

Theorem 1. Maximum a-posteriori estimation for the Gaussian mixture model satisfies the Generalized Benders Decomposition conditions as defined in section 2.4.

Definition 3.3.1 (Main Problem)**.**

$$
\begin{aligned}
\underset{\boldsymbol{z},\boldsymbol{\mu},\boldsymbol{\pi}}{\text{minimize}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}(y_i-\mu_k)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\log\pi_k \\
\text{subject to} \quad & \sum_{k=1}^{K}\pi_k - 1 = 0, \\
& \sum_{k=1}^{K} z_{ik} - 1 = 0, \quad i=1,\dots,n, \\
& -z_{ik} \le 0, \quad i=1,\dots,n;\ k=1,\dots,K, \\
& z_{ik} - 1 \le 0, \quad i=1,\dots,n;\ k=1,\dots,K, \\
& (\boldsymbol{\mu},\boldsymbol{\pi}) \in \mathcal{D} = \mathcal{M}\cap\Delta_K, \\
& \boldsymbol{z} \in \mathcal{Z}.
\end{aligned}
\tag{3.5}
$$

Definition 3.3.2. A twice differentiable function $f:\mathbb{R}^n\to\mathbb{R}$ is convex if $\mathbf{dom}f$ is a convex set and its Hessian is positive semidefinite, $\nabla^2 f(x)\succcurlyeq 0 \quad \forall x\in\mathbf{dom}f$.

Lemma 1. Maximum a-posteriori estimation for the Gaussian mixture model satisfies Generalized Benders Decomposition condition **C1**.

Proof. **C1** requires that $\mathcal{X}$ is nonempty, convex and the functions $f:\mathbb{R}^n\times\mathbb{R}^q\to\mathbb{R}$ and $\boldsymbol{g}:\mathbb{R}^n\times\mathbb{R}^q\to\mathbb{R}^p$ are convex for each fixed y, while the functions $\boldsymbol{h}:\mathbb{R}^n\times\mathbb{R}^l\to\mathbb{R}^m$ are linear for each fixed y, as defined in Equation 2.9.

First, we establish the following equivalences between the GBD form (Equation 2.9) and the main problem (Definition 3.3.1):

- $\mathcal{X}\equiv\mathcal{D}$
- $f\equiv\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}(y_i-\mu_k)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\log\pi_k$

- $\boldsymbol{g} \equiv (-z_{ik}, z_{ik} - 1)$ for $i = 1, \ldots, n;\ k = 1, \ldots, K$
- $\boldsymbol{h} \equiv (\sum_{k=1}^{K} \pi_k - 1, \sum_{k=1}^{K} z_{ik} - 1)$ for $i = 1, \ldots, n$.

Our task now is to prove that $\mathcal{D}$ is nonempty and convex, the functions f and $\boldsymbol{g}$ are convex for each fixed $\boldsymbol{z}$, and the functions $\boldsymbol{h}$ is linear for each fixed z. It is easy to show that $\mathcal{D}$ is nonempty and convex. The function f is twice-differentiable and $\mathbf{dom} f = \mathcal{D}$ is convex, so we prove f is convex for each fixed z by showing that its Hessian matrix is positive semidefinite for all $\pi_k, \mu_k \in \mathcal{D}$ using Definition 3.3.2. The Hessian, $\nabla^2 f(\boldsymbol{\mu}, \boldsymbol{\pi})$, is a $2K \times 2K$ block symmetric matrix:

$$\nabla^2 f(\boldsymbol{\mu}, \boldsymbol{\pi}) = \left[\begin{array}{c|c} \frac{\partial^2 f}{\partial \boldsymbol{\mu} \partial \boldsymbol{\mu}^T} & \frac{\partial^2 f}{\partial \boldsymbol{\mu} \partial \boldsymbol{\pi}^T} \\ \hline \frac{\partial^2 f}{\partial \boldsymbol{\pi} \partial \boldsymbol{\mu}^T} & \frac{\partial^2 f}{\partial \boldsymbol{\pi} \partial \boldsymbol{\pi}^T} \end{array}\right]$$

where

$$\frac{\partial^2 f}{\partial \mu_k \partial \mu_k'} = \begin{cases} 2\sum_{i=1}^{n} \bar{z}_{ik} & k = k' \\ 0 & k \neq k' \end{cases}, \quad \frac{\partial^2 f}{\partial \pi_k^2} = \begin{cases} \frac{n}{\pi_k} & k = k' \\ 0 & k \neq k' \end{cases}, \quad \text{and} \quad \frac{\partial^2 f}{\partial \mu_k \partial \pi_k'} = 0 \qquad \forall k, k'.$$

Since the Hessian, $\nabla^2 f(\mu, \pi)$, is diagonal and all of the elements on the diagonal are nonnegative in $\mathcal{D}$, it is positive semidefinite. Therefore, f is convex.

It remains to show that $\boldsymbol{g}$ is convex and $\boldsymbol{h}$ is linear for fixed z. There can only be two cases of $\boldsymbol{g}$ for fixed z: if $z_{ik} = 0\ \ \forall i, k$, then $\boldsymbol{g} = (0, -1)$ and if $z_{ik} = 1\ \ \forall i, k$, then $\boldsymbol{g} = (-1, 0)$. Therefore, $\boldsymbol{g}$ is convex for fixed z. $\boldsymbol{h}$ is linear for each fixed z, as the summation is a linear operator. □

Lemma 2. Maximum a-posteriori estimation for the Gaussian mixture model satisfies the Generalized Benders Decomposition condition **C2**.

Proof. **C2** requires that the set $\boldsymbol{Z_y} = \{\boldsymbol{z} \in \mathbb{R}^p \mid \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{0},\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{z} \text{ for some } \boldsymbol{x} \in \mathcal{X}\}$ is closed for each fixed y, as defined in Equation 2.9.

It is equivalent to show that the set $\mathcal{S}_z = \{\boldsymbol{s} \in \mathbb{R}^2 \mid (\sum_{k=1}^{K} \pi_k - 1, \sum_{k=1}^{K} z_{ik} - 1) = \mathbf{0}, (-z_{ik}, z_{ik} - 1) \leq \boldsymbol{s} \text{ for } i = 1, \ldots, n;\ k = 1, \ldots, K \text{ and some } \boldsymbol{\mu}, \boldsymbol{\pi} \in \mathcal{D}\}$ is closed for each fixed z. There can only be two cases of $\boldsymbol{s}$ for fixed z satisfying $(\sum_{k=1}^{K} \pi_k - 1, \sum_{k=1}^{K} z_{ik} - 1) = \mathbf{0}$: if $z_{ik} = 0\ \ \forall i, k$, then $(0, -1) \leq \boldsymbol{s}$, and if $z_{ik} = 1\ \ \forall i, k$,

then $(-1, 0) \leq \boldsymbol{s}$. Two possible sets, therefore, are $\mathcal{S}_{z_{ik}=0} = \{\boldsymbol{s} \in \mathbb{R}^2 \mid 0 \leq s_1, -1 \leq s_2\}$, and $\mathcal{S}_{z_{ik}=1} = \{\boldsymbol{s} \in \mathbb{R}^2 \mid -1 \leq s_1, 0 \leq s_2\}$, which are both closed. □

Lemma 3. Maximum a-posteriori estimation for the Gaussian mixture model satisfies the Generalized Benders Decomposition condition **C3**.

Proof. **C3** requires that for each fixed $y \in \mathcal{Y} \bigcap \mathcal{V}$, where $\mathcal{V} = \{\boldsymbol{y} \mid \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0},\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0},\ \text{for some } \boldsymbol{x} \in \mathcal{X}\}$, one of the following two conditions hold:

1. the resulting problem has a finite solution and has an optimal Lagrange multiplier vector for the equalities and inequalities
2. the value of objective function of the resulting Problem goes to $-\infty$ (is unbounded).

It is equivalent to show that for each fixed $z \in \mathcal{Z} \cap \mathcal{V}$, where $\mathcal{V} = \{\boldsymbol{z} \mid \boldsymbol{h}(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) = \boldsymbol{0},\ \boldsymbol{g}(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) \leq \boldsymbol{0},\ \text{for some } \boldsymbol{\mu}, \boldsymbol{\pi} \in \mathcal{D}\}$, one of the following two conditions hold: (1) the resulting problem has a finite solution and has an optimal Lagrange multiplier vector for the equalities and inequalities, or (2) the value of objective function of the resulting problem goes to $-\infty$. Since we are able to analytically solve optimal $\mu, \pi, (\nu, \lambda)$ in the main text, the problem has a finite solution and an optimal Lagrange multiplier vector with fixed $z_{ik} \in \mathcal{Z} \cap \mathcal{V}$, in which case the first condition holds. The second condition only applies to when $\pi_k = 0$ for any k because $\log \pi_k$ term goes to $-\infty$. Either one condition always holds for each fixed $z \in \mathcal{Z} \cap \mathcal{V}$. □

Corollary 1. The main problem, as defined in Definition 3.3.1, always has feasible solutions except the case when $\pi_k = 0$ for any k.

This is a corollary from lemma 3.

Theorem 2. Maximum a-posteriori estimation for the Gaussian mixture model satisfies the Generalized Benders Decomposition conditions.

Proof. By lemma 1, 2, 3, maximum a-posteriori estimation for the Gaussian mixture model satisfies the Generalized Benders Decomposition conditions. □

3.4 Derivations of GBD for the problem

3.4.1 A derivation of GBD for GMM with Full Constraints

3.4.1.1 Primal Problem

In Generalized Benders Decomposition, the primal problem is formed by fixing the integer variables and minimizing over the non-integer variables,

$$
\begin{aligned}
\underset{\boldsymbol{\mu},\boldsymbol{\pi}}{\text{minimize}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{K} \bar{z}_{ik}^{(j)}(y_i-\mu_k)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} \bar{z}_{ik}^{(j)} \log \pi_k \\
\text{subject to} \quad & \sum_{k=1}^{K} \pi_k - 1 = 0, \\
& \sum_{k=1}^{K} \bar{z}_{ik}^{(j)} - 1 = 0, && i = 1,\dots,n, \\
& -\bar{z}_{ik}^{(j)} \le 0, && i = 1,\dots,n;\ k = 1,\dots,K, \\
& \bar{z}_{ik}^{(j)} - 1 \le 0, && i = 1,\dots,n;\ k = 1,\dots,K, \\
& (\boldsymbol{\mu},\boldsymbol{\pi}) \in \mathcal{D} = \mathcal{M} \cap \Delta_K,
\end{aligned}
\tag{3.6}
$$

with fixed $\boldsymbol{z}^{(j)}$ where j stands for the iteration counter.

3.4.1.2 Master Problem

The master problem of the Gaussian Mixture model is

$$
\begin{aligned}
\underset{\boldsymbol{z},\,\mu_B}{\text{minimize}} \quad & \mu_B \\
\text{subject to} \quad & \mu_B \ge \inf_{\boldsymbol{\mu},\boldsymbol{\pi}\in\mathcal{D}} L(\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{z},\nu,\lambda) \quad \forall\nu,\forall\lambda \ge 0, \\
& \boldsymbol{z} \in \mathcal{Z},
\end{aligned}
\tag{3.7}
$$

where

$$\begin{aligned} L(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}, \nu, \lambda) &= f(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) + \nu^T \boldsymbol{h}(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) + \lambda^T \boldsymbol{g}(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} (y_i - \mu_k)^2 - \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k \\ &\quad + \nu_1 \left(\sum_{k=1}^{K} \pi_k - 1 \right) + \sum_{i=1}^{n} \nu_{2i} \left(\sum_{k=1}^{K} z_{ik} - 1 \right) \\ &\quad + \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_{1ik} (-z_{ik}) + \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_{2ik} (z_{ik} - 1) . \end{aligned} \tag{3.8}$$

We can solve $\inf_{\boldsymbol{\mu}, \boldsymbol{\pi} \in \mathcal{D}} L(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}, \nu, \lambda)$ analytically for the optimizer $(\hat{\mu_k}, \hat{\pi_k})$ using derivatives:

$$\hat{\pi} = \frac{1}{n} \sum_{k=1}^{K} z_{ik} \tag{3.9}$$

and

$$\hat{\mu} = \frac{\sum_{i=1}^{n} z_{ik} y_i}{\sum_{i=1}^{n} z_{ik}}. \tag{3.10}$$

Substituting these optimizers into the master problem gives lower bound function:

$$\begin{aligned} L^*(\boldsymbol{z}; \nu, \lambda) &= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \left(y_i - \frac{\sum_{i=1}^{n} z_{ik} y_i}{\sum_{i=1}^{n} z_{ik}} \right)^2 - \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \left(\frac{1}{n} \sum_{i=1}^{n} z_{ik} \right) \\ &\quad + \sum_{i=1}^{n} \nu_{2i} \left(\sum_{k=1}^{K} z_{ik} - 1 \right) \\ &\quad + \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_{1ik} (-z_{ik}) + \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_{2ik} (z_{ik} - 1) \end{aligned} \tag{3.11}$$

where the term associated with ν_1 is always equal to zero and is eliminated.

The relaxed master problem is then

$$\underset{\boldsymbol{z}}{\text{minimize}} \quad \mu_B \tag{3.12a}$$

$$\text{subject to} \quad \mu_B \geq L^* \left(\boldsymbol{z}; \nu^{(j)}, \lambda^{(j)} \right) \quad j = 1, \ldots \tag{3.12b}$$

where $(\nu^{(j)}, \lambda^{(j)})$ are the Lagrange multipliers associated with the j-th subproblem solution.

3.4.1.3 Flaw

The value of $L^*\left(\boldsymbol{z}; \nu^{(j)}, \lambda^{(j)}\right)$ is maximized if and only if $\lambda_{1ik} \in \mathcal{R}$ and $\lambda_{2ik} = -\infty$ for i, k such that $z_{ik} = 0$, or $\lambda_{1ik} = -\infty$ and $\lambda_{2ik} \in \mathcal{R}$ for i, k such that $z_{ik} = 1$. These values of $\lambda^{(j)}$, undefined and $-\infty$, are not providing new information to solve the primal problem in the next iteration.

3.4.2 A derivation of GBD for GMM with Constraints over π_k

3.4.2.1 Primal Problem

The primal problem of the Gaussian mixture model with constraints restricted to π_k is

$$\begin{aligned} \underset{\boldsymbol{\mu}, \boldsymbol{\pi}}{\text{minimize}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}(y_i - \mu_k)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik} \log \pi_k \\ \text{subject to} \quad & \sum_{k=1}^{K} \pi_k = 1, \\ & 0 \leq \pi_k \leq 1, \qquad k = 1, \ldots, K, \\ & M_k^L \leq \mu_k \leq M_k^U, \quad k = 1, \ldots, K. \end{aligned} \tag{3.13}$$

The optimal Lagrange dual variables are passed from the primal problem to the master problem, so we proceed as follows:

1. Form the Lagrange function.

2. Form the dual function by minimizing the Lagrange function with respect to the primal variables.

3. Find the optimal Lagrange dual variables by maximizing the dual function with respect to the dual variables.

3.4.2.2 Lagrange Function

The Lagrange function for Problem (3.13) is

$$\begin{aligned} L(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{z}, \boldsymbol{\lambda}, \nu) = & \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik}(y_i - \mu_k)^2 - \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k \\ & + \nu \left(\sum_{k=1}^{K} \pi_k - 1 \right) + \sum_{k=1}^{K} \lambda_{1k} \left(\pi_k - 1 \right) + \sum_{k=1}^{K} \lambda_{2k} \left(-\pi_k \right) \\ & + \sum_{k=1}^{K} \lambda_{3k} \left(\mu_k - M_k^U \right) + \sum_{k=1}^{K} \lambda_{4k} \left(M_k^L - \mu_k \right) \end{aligned} \tag{3.14}$$

3.4.2.3 Dual Function

The dual function is obtained by minimizing the Lagrange function over the primal variables

$$\begin{aligned} \xi(\boldsymbol{z}, \boldsymbol{\lambda}, \nu) = \underset{\boldsymbol{\mu}, \boldsymbol{\pi}}{\text{minimize}} \quad & L(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq 0. \end{aligned} \tag{3.15}$$

Optimal μ Since the Lagrange function is convex in $\boldsymbol{\mu}$ the optimizing value is found by setting the derivative equal to zero. The partial derivative of the Lagrange function with respect to μ_k is

$$\frac{\partial L}{\partial \mu_k} = \sum_{i=1}^{n} z_{ik}(y_i - \mu_k)(-2) + \lambda_{3k} - \lambda_{4k}.$$

Setting the partial derivative equal to zero gives the minimizer,

$$\hat{\mu}_k = \frac{2 \sum_{i=1}^{n} z_{ik} y_i + \lambda_{4k} - \lambda_{3k}}{2 \sum_{i=1}^{n} z_{ik}}. \tag{3.16}$$

Optimal π Since the Lagrange function is convex in $\boldsymbol{\pi}$ the optimizing value is found by setting the derivative equal to zero. The partial derivative of the Lagrange function with respect to π_k is

$$\frac{\partial L}{\partial \pi_k} = -\frac{1}{\pi_k} \sum_{i=1}^{n} z_{ik} + \nu + \lambda_{1k} - \lambda_{2k}$$

Setting the partial derivative equal to zero gives the minimizer,

$$\hat{\pi}_k = \frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu}. \tag{3.17}$$

The dual function can now be formed by substituting the optimal π and μ into the Lagrange function,

$$\begin{aligned}\xi(\boldsymbol{z}, \boldsymbol{\lambda}, \nu) = &\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\left(y_i - \hat{\mu}_k\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\log\hat{\pi}_k \\ &+ \nu\left(\sum_{k=1}^{K}\hat{\pi}_k - 1\right) + \sum_{k=1}^{K}\lambda_{1k}\left(\hat{\pi}_k - 1\right) + \sum_{k=1}^{K}\lambda_{2k}(-\hat{\pi}_k) \\ &+ \sum_{k=1}^{K}\lambda_{3k}\left(\hat{\mu}_k - M_k^U\right) + \sum_{k=1}^{K}\lambda_{4k}\left(M_k^L - \hat{\mu}_k\right)\end{aligned} \tag{3.18}$$

3.4.2.4 Optimal Dual Variables

Generalized Benders' Decomposition passes the optimal dual variables to the master problem. Thus, here, we solve for the optimal dual variables.

Theorem 3. ξ is concave in $\boldsymbol{\lambda}$ and ν (see the Appendix A.1 for a proof).

Optimal $(\lambda_{1k},\ \lambda_{2k},\ \nu)$ The partial derivative of the dual function with respect to λ_{1k} is found by the chain rule

$$\begin{aligned}\frac{\partial\xi}{\partial\lambda_{1k}} = \frac{\partial\xi}{\partial\hat{\pi}_k}\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}} = &-\sum_{i=1}^{n} z_{ik}\frac{1}{\hat{\pi}_k}\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}} + \nu\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}} + \left(\hat{\pi}_k - 1\right) \\ &+ \lambda_{1k}\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}} - \lambda_{2k}\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}}\end{aligned}$$

The partial derivative of $\hat{\pi}_k$ with respect to λ_{1k} is

$$\frac{\partial\hat{\pi}_k}{\partial\lambda_{1k}} = \frac{\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2}.$$

Substituting gives

$$\frac{\partial \xi}{\partial \lambda_{1k}} = -\sum_{i=1}^{n} z_{ik} \frac{\lambda_{2k} - \lambda_{1k} - \nu}{\sum_{i=1}^{n} z_{ik}} \frac{\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} + \left(\frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu} - 1 \right) + (\lambda_{1k} - \lambda_{2k} + \nu) \frac{\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2}.$$

Collecting terms gives

$$\frac{\partial \xi}{\partial \lambda_{1k}} = -\frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu} + \left(\frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu} - 1 \right) - \frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu}.$$

The first and second terms cancel leaving

$$\frac{\partial \xi}{\partial \lambda_{1k}} = -1 - \frac{\sum_{i=1}^{n} z_{ik}}{\lambda_{2k} - \lambda_{1k} - \nu}.$$

Setting the derivative equal to zero and multiplying through $(\lambda_{2k} - \lambda_{1k} - \nu)$ gives

$$0 = -\lambda_{2k} - \lambda_{1k} - \nu - 1.$$

Solving for λ_{1k} gives

$$\lambda_{1k} = -\lambda_{2k} - \nu - 1.$$

Using a similar procedure to solve for the optimal λ_{4k} gives

$$\lambda_{2k} = 0.$$

Using a similar procedure to solve for the optimal ν gives

$$\nu = -\lambda_{2k} - \lambda_{1k} - 1.$$

Optimal $(\lambda_{3k},\ \lambda_{4k})$ The partial derivative of the dual function with respect to λ_{3k} is found by the chain rule

$$\begin{aligned}\frac{\partial \xi}{\partial \lambda_{3k}} = \frac{\partial \xi}{\partial \hat{\mu}_k}\frac{\partial \hat{\mu}_k}{\partial \lambda_{3k}} = &-2\sum_{i=1}^{n} z_{ik}\left(y_i - \hat{\mu}_k\right)\frac{\partial \hat{\mu}_k}{\partial \lambda_{3k}} \\ &+ \left(\hat{\mu}_k - M_k^U\right) + \lambda_{3k}\frac{\partial \hat{\mu}_k}{\partial \lambda_{3k}} - \lambda_{4k}\frac{\partial \hat{\mu}_k}{\partial \lambda_{3k}}.\end{aligned}$$

The partial derivative of $\hat{\mu}_k$ with respect to λ_{3k} is

$$\frac{\partial \hat{\mu}_k}{\partial \lambda_{3k}} = \frac{-1}{2\sum_{i=1}^{n} z_{ik}}.$$

Substituting gives

$$\begin{aligned}\frac{\partial \xi}{\partial \lambda_{3k}} = &\left(\frac{1}{\sum_i z_{ik}}\right)\sum_{i=1}^{n} z_{ik}\left(y_i - \frac{2\sum_i z_{ik}y_i + \lambda_{4k} - \lambda_{3k}}{2\sum_i z_{ik}}\right) \\ &+ \frac{2\sum_i z_{ik}y_i + \lambda_{4k} - \lambda_{3k}}{2\sum_i z_{ik}} - M_k^U \\ &- \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} + \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{4k}.\end{aligned}$$

Collecting terms gives

$$\begin{aligned}\frac{\partial \xi}{\partial \lambda_{3k}} = &\left(\frac{1}{\sum_i z_{ik}}\right)\sum_i z_{ik}y_i \\ &- \left(\frac{\left(\sum_i z_{ik}\right)}{2\left(\sum_i z_{ik}\right)^2}\right)\cdot\left(2\sum_i z_{ik}y_i + \lambda_{4k} - \lambda_{3k}\right) \\ &+ \left(\frac{1}{2\left(\sum_i z_{ik}\right)}\right)\cdot\left(2\sum_i z_{ik}y_i + \lambda_{4k} - \lambda_{3k}\right) \\ &- M_k^U - \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} + \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{4k}.\end{aligned}$$

The second and third terms cancel leaving

$$\frac{\partial \xi}{\partial \lambda_{3k}} = \left(\frac{1}{\sum_i z_{ik}}\right)\sum_i z_{ik}y_i - M_k^U - \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} + \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{4k}.$$

Setting the derivative equal to zero and multiplying through by $(2\sum_i z_{ik})$ gives

$$0 = 2\sum_i z_{ik} y_i - 2M_k^U \sum_i z_{ik} - \lambda_{3k} + \lambda_{4k}.$$

Solving for λ_{3k} gives

$$\lambda_{3k} = 2\sum_i z_{ik}\left(y_i - M_k^U\right) + \lambda_{4k}. \tag{3.19}$$

Using a similar procedure to solve for the optimal λ_{4k} gives

$$\lambda_{4k} = 2\sum_i z_{ik}\left(M_k^L - y_i\right) + \lambda_{3k}. \tag{3.20}$$

Equations (3.19) and (3.20) are a system of linearly dependent equations

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_{3k} \\ \lambda_{4k} \end{bmatrix} = \begin{bmatrix} 2\sum_i z_{ik}\left(y_i - M_k^U\right) \\ 2\sum_i z_{ik}\left(y_i - M_k^L\right) \end{bmatrix}. \tag{3.21}$$

3.4.2.5 Flaw

The system 3.21 does not have a feasible solution, and thus there does not exist a local/global maximum. When there is not local maximum point within the boundary, it should exist at the boundary. As $\lambda_{3k}, \lambda_{4k} \geq 0$, boundaries are at $\lambda_{3k}, \lambda_{4k} = 0$. First, we fix $\lambda_{4k} = 0$ and take derivative over λ_{3k}.

$$\begin{aligned} \frac{\partial \xi_{\lambda_{4k}=0}}{\partial \lambda_{3k}} &= \left(\frac{1}{\sum_i z_{ik}}\right)\sum_i z_{ik} y_i - M_k^U - \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} \\ \lambda_{3k} &= 2\sum_i z_{ik}\left(y_i - M_k^U\right) \quad \text{for} \quad \lambda_{4k} = 0. \end{aligned} \tag{3.22}$$

However, due to the constraint that $\lambda_{3k} \geq 0$ and $y_i \leq M_k^U$,

$$\lambda_{3k} = \max\left(2\sum_i z_{ik}\left(y_i - M_k^U\right), 0\right) = 0 \quad \text{for} \quad \lambda_{4k} = 0. \tag{3.23}$$

Using a similar procedure to solve for the optimal λ_{4k} gives $\lambda_{4k} = 0$ for $\lambda_{3k} = 0$. Constant zero values of $(\lambda_{3k},\ \lambda_{4k})$ are not providing new information to solve the primal problem in the next iteration.

3.4.3 A derivation of GBD for GMM with Taylor Series

3.4.3.1 Theory

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} \quad & G(x, y) \geq 0, \\ & \boldsymbol{x} \in \mathcal{X}, \\ & \boldsymbol{y} \in \mathcal{Y} \end{aligned} \tag{3.24}$$

is equivalent to

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & \underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} \quad & G(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \\ & \boldsymbol{x} \in \mathcal{X}, \\ & \boldsymbol{y} \in \mathcal{Y}. \end{aligned} \tag{3.25}$$

We can substitute $f(\boldsymbol{x}, \boldsymbol{y})$ with $v(\boldsymbol{y})$:

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & v(\boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{y} \in \mathcal{V} \cap \mathcal{Y} \end{aligned} \tag{3.26}$$

where $\mathcal{V} = \{\text{for } \boldsymbol{y},\ \exists \boldsymbol{x} \in \mathcal{X} \text{ s.t. } G(\boldsymbol{x}, \boldsymbol{y}) \geq 0\}$ and $v(\boldsymbol{y}) = \min_{\boldsymbol{x} \in \mathcal{X}} \{f(\boldsymbol{x}, \boldsymbol{y}) | G(\boldsymbol{x}, \boldsymbol{y}) \geq 0\}$. By duality:

$$v(\boldsymbol{y}) = \sup_{u \geq 0} \inf_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}) + u^t G(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$$

for all $\boldsymbol{y} \in \mathcal{V} \cap \mathcal{Y}$. The problem is equivalent to:

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & \theta \\ \text{subject to} \quad & \theta \geq \inf_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}) + u^t G(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \quad \forall u \geq 0. \end{aligned} \tag{3.27}$$

By Lagrangian duality, at optimal $\boldsymbol{y}$:

$$\nabla_{\boldsymbol{y}} v(\boldsymbol{y}) = \nabla_{\boldsymbol{y}} f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \hat{u}^t \nabla_{\boldsymbol{y}} G(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}).$$

Hence, optimality cuts can be written in the form:

$$\theta \geq v(\hat{\boldsymbol{y}}) + \nabla_{\boldsymbol{y}} v(\boldsymbol{y})^t (\boldsymbol{y} - \hat{\boldsymbol{y}}).$$

3.4.3.2 Application

In the standard form, $\boldsymbol{x}$ represents the set of continuous, non-complicating variables. In our main problem, the GMM, these variables are $(\boldsymbol{\mu}, \boldsymbol{\pi})$ because the problem is convex over $(\boldsymbol{\mu}, \boldsymbol{\pi})$ when $\boldsymbol{z}$ is fixed. Likewise, in the standard form, $\boldsymbol{y}$ represents the set of complicating variables. In the GMM, these variables are $\boldsymbol{z}$ as they are the integer valued variables. The function f can be reformulated with such equivalences as

$$f(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} (y_i - \mu_k)^2 - \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k.$$

The optimal cuts can be rewriten in the form:

$$\begin{aligned} \theta \geq & v(\hat{\boldsymbol{z}}) + \nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t (\boldsymbol{z} - \hat{\boldsymbol{z}}) \\ & = \inf_{\boldsymbol{\mu}, \boldsymbol{\pi}} f(\boldsymbol{\mu}, \boldsymbol{\pi}, \hat{\boldsymbol{z}}) + u^t G(\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{z}) + \left[\nabla_{\boldsymbol{z}} f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{z}}) + \hat{u}^t \nabla_{\boldsymbol{z}} G(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{z}}) \right]^t (\boldsymbol{z} - \hat{\boldsymbol{z}}). \end{aligned}$$

As our main problem does not have G, the constraints that involve both $(\boldsymbol{\mu}, \boldsymbol{\pi})$ and $\boldsymbol{y}$, the optimal cuts can be simplified in the form:

$$\begin{aligned} \theta \geq & v(\hat{\boldsymbol{z}}) + \nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t (\boldsymbol{z} - \hat{\boldsymbol{z}}) \\ & = \inf_{\boldsymbol{\mu}, \boldsymbol{\pi}} f(\boldsymbol{\mu}, \boldsymbol{\pi}, \hat{\boldsymbol{z}}) + \nabla_{\boldsymbol{z}} f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{z}})^t (\boldsymbol{z} - \hat{\boldsymbol{z}}). \end{aligned}$$

Solve for $v(\hat{\boldsymbol{z}}) = \inf_{\boldsymbol{\mu}, \boldsymbol{\pi}} f(\boldsymbol{\mu}, \boldsymbol{\pi}, \hat{\boldsymbol{z}})$

3.4.3.2.0.1 Optimal μ_k

$$\hat{\mu_k} = \frac{\sum_i^n z_{ik} y_i}{\sum_i^n z_{ik}}$$

3.4.3.2.0.2 Optimal π_k

$$\hat{\pi_k} = \frac{\sum_i^n z_{ik}}{n}$$

With optimal values of μ_k and π_k, we get

$$v(\boldsymbol{z}) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\left(y_i - \frac{\sum_{i'}^n z_{'ik} y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\log\left(\frac{\sum_{i'}^n z_{i'k}}{n}\right),$$
$$v(\hat{\boldsymbol{z}}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{z}_{ik}\left(y_i - \frac{\sum_{i'}^n \hat{z}_{i'k} y_{i'}}{\sum_{i'}^n \hat{z}_{i'k}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{z}_{ik}\log\left(\frac{\sum_{i'}^n \hat{z}_{i'k}}{n}\right).$$

Solve for $\nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t(\boldsymbol{z} - \hat{\boldsymbol{z}}) = \nabla_{\boldsymbol{z}} f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{z}})^t(\boldsymbol{z} - \hat{\boldsymbol{z}})$

$$\nabla_{\boldsymbol{z}} v(\boldsymbol{z}) = \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \boldsymbol{z}} = \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \boldsymbol{\mu}}\frac{d\boldsymbol{\mu}}{d\boldsymbol{z}} + \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \boldsymbol{\pi}}\frac{d\boldsymbol{\pi}}{d\boldsymbol{z}}.$$

We treat $\boldsymbol{z}$ as a vector of length $i \times k$ instead of a matrix of $i \times k$, then we get

$$\nabla_{\boldsymbol{z}} v(\boldsymbol{z})_{ik} = \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \mu_k}\frac{d\mu_{k'}}{dz_{ik}} + \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \pi_k}\frac{d\pi_{k'}}{dz_{ik}}.$$

There are two cases: $k = k'$ and $k \neq k'$. If $k \neq k'$, then $\frac{d\mu_k}{dz_{ik}} = 0$ and $\frac{d\pi_k}{dz_{ik}} = 0$ leaving a non-zero derivative only if $k = k'$:

$$\begin{aligned}\nabla_{\boldsymbol{z}} v(\boldsymbol{z})_{ik} &= \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \mu_k}\frac{d\mu_k}{dz_{ik}} + \frac{\partial f(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\pi}}, \boldsymbol{z})}{\partial \pi_k}\frac{d\pi_k}{dz_{ik}} \\ &= -2z_{ik}\left(y_i - \hat{\mu}_k\right)\frac{d\hat{\mu}_k}{dz_{ik}} + \left(y_i - \hat{\mu}_k\right)^2 - \frac{1}{\hat{\pi}_k} z_{ik}\frac{d\hat{\pi}_k}{dz_{ik}} - \log(\hat{\pi}_k).\end{aligned}$$

The derivatives of $\hat{\mu}_k$ and $\hat{\pi}_k$ with respect to z_{ik} are

$$\frac{d\hat{\mu}_k}{dz_{ik}} = \frac{y_i(\sum_{i'}^n z_{i'k}) - (\sum_{i'}^n z_{i'k} y_{i'})}{(\sum_{i'}^n z_{i'k})^2} \quad \text{and} \quad \frac{d\hat{\pi}_k}{dz_{ik}} = \frac{1}{n}.$$

Substitution gives

$$\begin{aligned}
\nabla_{\boldsymbol{z}} v(\boldsymbol{z})_{ik} &= -2z_{ik}\left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)\left(\frac{y_i(\sum_{i'}^n z_{i'k}) - (\sum_{i'}^n z_{i'k}y_{i'})}{(\sum_{i'}^n z_{i'k})^2}\right) \\
&\quad + \left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 - \left(\frac{n}{\sum_{i'}^n z_{i'k}}\right) z_{ik}\left(\frac{1}{n}\right) - \log\left(\frac{\sum_{i'}^n z_{i'k}}{n}\right) \\
&= -2z_{ik}\left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 \frac{1}{\sum_{i'}^n z_{i'k}} - \frac{z_{ik}}{\sum_{i'}^n z_{i'k}} \\
&\quad + \left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 - \log\left(\frac{\sum_{i'}^n z_{i'k}}{n}\right) \\
&= -\frac{z_{ik}}{\sum_{i'}^n z_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 + 1\right) \\
&\quad + \left(y_i - \frac{\sum_{i'}^n z_{i'k}y_{i'}}{\sum_{i'}^n z_{i'k}}\right)^2 - \log\left(\frac{\sum_{i'}^n z_{i'k}}{n}\right).
\end{aligned}$$

At fixed $\boldsymbol{z}$, it becomes

$$\begin{aligned}
\nabla_{\boldsymbol{z}} v(\hat{\boldsymbol{z}})_{ik} &= -\frac{\hat{z}_{ik}}{\sum_{i'}^n \hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^n \hat{z}_{i'k}y_{i'}}{\sum_{i'}^n \hat{z}_{i'k}}\right)^2 + 1\right) \\
&\quad + \left(y_i - \frac{\sum_{i'}^n \hat{z}_{i'k}y_{i'}}{\sum_{i'}^n \hat{z}_{i'k}}\right)^2 - \log\left(\frac{\sum_{i'}^n \hat{z}_{i'k}}{n}\right).
\end{aligned}$$

Multiplication with $(\boldsymbol{z} - \hat{\boldsymbol{z}})$ gives

$$\begin{aligned}
\nabla_{\boldsymbol{z}} v(\hat{\boldsymbol{z}})^t(\boldsymbol{z} - \hat{\boldsymbol{z}}) &= \sum_{i=1}^n \sum_{k=1}^K \nabla_{\boldsymbol{z}}\nabla_{\boldsymbol{z}} v(\hat{\boldsymbol{z}})_{ik}(z_{ik} - \hat{z}_{ik}) \\
&= -\sum_{i=1}^n \sum_{k=1}^K \left[(z_{ik} - \hat{z}_{ik})\frac{\hat{z}_{ik}}{\sum_{i'}^n \hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^n \hat{z}_{i'k}y_{i'}}{\sum_{i'}^n \hat{z}_{i'k}}\right)^2 + 1\right)\right] \\
&\quad + \sum_{i=1}^n \sum_{k=1}^K (z_{ik} - \hat{z}_{ik})\left(y_i - \frac{\sum_{i'}^n \hat{z}_{i'k}y_{i'}}{\sum_{i'}^n \hat{z}_{i'k}}\right)^2 \\
&\quad - \sum_{i=1}^n \sum_{k=1}^K (z_{ik} - \hat{z}_{ik}) \log\left(\frac{\sum_{i'}^n \hat{z}_{i'k}}{n}\right).
\end{aligned}$$

Optimal Cuts Recall that the simplified form of optimal cuts is

$$\theta \geq v(\hat{\boldsymbol{z}}) + \nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t(\boldsymbol{z} - \hat{\boldsymbol{z}}).$$

Substitution of $v(\hat{\boldsymbol{z}})$ and $\nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t(\boldsymbol{z}-\hat{\boldsymbol{z}})$ with expressions above gives

$$
\begin{aligned}
\theta \geq & v(\hat{\boldsymbol{z}}) + \nabla_{\boldsymbol{z}} v(\boldsymbol{z})^t(\boldsymbol{z}-\hat{\boldsymbol{z}}) \\
= & \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{z}_{ik}\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K}\hat{z}_{ik}\log\left(\frac{\sum_{i'}^{n}\hat{z}_{i'k}}{n}\right) \\
& - \sum_{i=1}^{n}\sum_{k=1}^{K}\left[(z_{ik}-\hat{z}_{ik})\frac{\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 + 1\right)\right] \\
& + \sum_{i=1}^{n}\sum_{k=1}^{K}(z_{ik}-\hat{z}_{ik})\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K}(z_{ik}-\hat{z}_{ik})\log\left(\frac{\sum_{i'}^{n}\hat{z}_{i'k}}{n}\right) \\
= & \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{K}z_{ik}\log\left(\frac{\sum_{i'}^{n}\hat{z}_{i'k}}{n}\right) \\
& - \sum_{i=1}^{n}\sum_{k=1}^{K}\left[(z_{ik}-\hat{z}_{ik})\frac{\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 + 1\right)\right]
\end{aligned}
$$

The cut can be rewritten in linear form of $a \times z_{ik} + b$ over z_{ik} as

$$
\begin{aligned}
\theta \geq \sum_{i=1}^{n}\sum_{k=1}^{K}\Bigg[& z_{ik}\Bigg[\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2\left(1 - \frac{2\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right) - \log\left(\frac{\sum_{i'}^{n}\hat{z}_{i'k}}{n}\right) - \frac{\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\Bigg] \\
& + \frac{(\hat{z}_{ik})^2}{\sum_{i'}^{n}\hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 + 1\right)\Bigg],
\end{aligned}
$$

where a is

$$
\left[\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2\left(1 - \frac{2\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right) - \log\left(\frac{\sum_{i'}^{n}\hat{z}_{i'k}}{n}\right) - \frac{\hat{z}_{ik}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right],
$$

and b is

$$
\frac{(\hat{z}_{ik})^2}{\sum_{i'}^{n}\hat{z}_{i'k}}\left(2\left(y_i - \frac{\sum_{i'}^{n}\hat{z}_{i'k}y_{i'}}{\sum_{i'}^{n}\hat{z}_{i'k}}\right)^2 + 1\right).
$$

3.4.3.3 Flaw

The linear form of optimal cuts implies that the cuts are getting updated instead of added. In that way, the solver may stop at local optima instead of global optimum.

3.5 Discussion

We formulate the MAP clustering problem under the Gaussian mixture model as an optimization problem. This allows the problem to accommodate hard logical constraints, preserve the combinatorial structure, and afford a certificate of global optimality. Yet, the coupling of the integer domain with the continuous domain along with their associated nonlinearities make the MINLP problems very challenging from the theoretical, algorithmic, and computational point of view (Floudas, 1995). The higher the number of binary variables is, the exponentially larger combinatorial problem one has to face – finding a global solution to nonconvex MINLP problems is NP-hard (Murty and Kabadi, 1987). Despite the discouraging complexity analysis findings, substantial advancements have been made in the MINLP field including the Generalized Benders Decomposition (Geoffrion, 1972).

Unfortunately, GBD has not always been effective in all of its applications. Bazaraa and Sherali (1980) noticed that GBD required a large number of iterations to solve their MILP model for quadratic assignment problems of realistic size. Sahinidis et al. (1989) discovered that branch and bound was substantially faster than GBD for their multiperiod MILP for long range planning in the chemical industries. Lastly, in regards to its application on nonconvex problems, Sahinidis et al. (1989) have seen instances in which the global optima are not produced for their MINLP model for the planning of chemical processes. In addition to findings in literature, experiments in this chapter suggest that for GBD to work, 'meaningful' conditions are needed that generate iterations effectively.

CHAPTER 4

INFERENCE FOR FACTOR GRAPHS VIA BENDERS DECOMPOSITION

Abstract

We study a variational inference method for general factor graphs. Maximum a-posteriori (MAP) inference is widely used when making decisions in Bayesian models. Previous research has found that MAP inference can be relaxed through Lagrangian or linear programming, and its dual form yields a computationally-efficient inference algorithm. However, the decision as to which constraints to add back in to the relaxed problem is often heuristic or specific to the model structure. We present a method for MAP inference in general Bayesian factor models that sequentially adds optimal constraints to the fully relaxed dual problem using Benders decomposition. The algorithm guarantees ϵ-convergence through iterating between upper and lower bounds. We also present a theorem showing when the algorithm yields an ϵ-globally optimal solution and when it may yield a locally optimal solution. We apply our proposed method to two Bayesian factor models: Bayesian Gaussian mixture model (BGMM) and latent Dirichlet allocation (LDA) model. For each model, we implement the algorithm on standard data sets and compare the results to those of variational Bayes and Gibbs sampler.. Our proposed method outperforms variational Bayes and Gibbs sampler on some of the data sets in terms of achieving the higher log(MAP) value, and always end with guaranteed optimality, which other two methods do not provide. Lastly, we discuss a few details to consider when applying the method.

4.1 Motivation

Graphical models can describe complex dependencies among the variables despite their concise description. These dependencies are results of all the local interactions. The problem of probabilistic inference is to reason about the underlying state of the variables in the model. However, probabilistic inference is often computationally intractable due to the dependencies. Finding the maximum probability assignment is challenging even if the local functions depend only on two variables. To find the MAP assignment of a discrete graphical model such as the Ising model is NP-hard (Kalinin and Berloff, 2022). This has led to extensive research into approximation algorithms.

Many methods used to find the MAP assignment can be seen as a specific combination of a relaxation of the problem and a search algorithm for locating a local or global minimum. Locally optimal and heuristic methods lack optimality guarantees or bounds on solution quality that indicate the potential for further improvement. Methods that use an optimization framework do not guarantee if the solution is globally optimal, and also only focus on certain types of graphical models. For these reasons, we are motivated to develop a method to find the MAP assignment for general graphical models that provides optimality guarantees.

4.2 Problem Setup

As section 3.2 shows in detail with the example of Gaussian mixture model, the MAP assignment is a solution to

$$\max_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y};\phi), \tag{4.1}$$

where $\mathbf{x}$ is the vector of latent variables, $\mathbf{y}$ is the observed data, and ϕ is the vector of parameters of a Bayesian model. Latent variables, $\mathbf{z}$, that are in the model, but not subjects of inference are marginalized, $p(\mathbf{x}|\mathbf{y};\phi) = \int_{\mathbf{z}} p(\mathbf{x},\mathbf{z}|\mathbf{y};\phi)d\mathbf{z}$.

First, we can decompose the MAP inference problem as a factor graph (Sontag et al., 2012),

$$\text{MAP}(\theta) = \max_{\mathbf{x}} \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \theta_f(\mathbf{x}_f), \tag{4.2}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathcal{V}$ is the set of latent variables, $\mathcal{F}$ is the set of factors, and $\mathbf{x}_f$ is the subset of the latent variables that are involved in factor f. This decomposition is accomplished by expanding the posterior distribution function and collecting terms that involve multiple latent variables. The functions θ incorporate the data, $\mathbf{y}$, and the parameters, ϕ.

Second, the model is augmented by duplicating all the latent variables that are involved in factors $\mathbf{x}_f \; \forall f \in \mathcal{F}$; we denote this set of factor variables $\mathbf{x}^{\mathcal{F}} = \{\mathbf{x}_f : f \in \mathcal{F}\}$, where $\mathbf{x}_f = \{x_v : x_v \in f\}$. Constraints are added to tie the newly created factor variables to the original latent variables.

$$\begin{aligned} \text{MAP}(\theta) \quad &= \quad \max_{\mathbf{x}, \mathbf{x}^{\mathcal{F}}} \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \theta_f(\mathbf{x}_v^f) && (4.3) \\ \text{subject to} \qquad & \qquad x_v = x_v^f, \quad \forall v \in f, f \in \mathcal{F} && (4.4) \end{aligned}$$

This MAP inference problem is equivalent to (4.2), but is useful because it enables a relaxation of the constraints. Without the constraints, the problem would simply decompose into independent maximization for each factor, each of which can be done efficiently.

Third, to remove these "complicating" constraints, we may use the technique of Lagrangian relaxation ((Geoffrion, 1974), (Shlesinger, 1976), (Lemarechal, 2001), (Guignard, 2003)). The Lagrange function for the factor graph MAP inference problem is

$$L(\delta, \mathbf{x}, \mathbf{x}^{\mathcal{F}}) = \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \theta_f(\mathbf{x}_v^f) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \delta_{f_v}(x_v - x_v^f), \tag{4.5}$$

where δ_{f_v} is the Lagrange variables associated with the constraint $g_{f_v}(x_v, x_v^f) := x_v - x_v^f = 0$. The MAP estimation problem can be solved by relaxing (removing) all

of the constraints and solving the dual problem. The resulting optimization problem completely separates and the size of the largest optimization problem is the largest factor. Note that the fully relaxed solution may be non-feasible for the original problem.

The problem is then how should we select constraints to add back in to the fully relaxed Lagrange dual problem. Here, we present a method based on Benders' decomposition that optimally selects violated constraints and iteratively tightens the relaxation while maintaining rigorous bounds on the value of the optimum.

4.3 Related Work

Many methods of finding the MAP assignment can be interpreted as a specific combination of a relaxation of problem (4.1) and a search algorithm for finding a local or global minimum.

Simulated annealing Simulated annealing methods are theoretically guaranteed to converge to a global optimum of a nonlinear objective. However, choosing the annealing schedule for a particular problem is challenging and the guarantee of global optimality only exists in the limit of the number of steps; there is no general way to choose the annealing schedule or monitor convergence (Andrieu et al., 2003). Even so, modern simulated annealing-type methods such as basin hopping (Wales and Doye, 1997) and genetic algorithms (Kuo and Lin, 2013) have shown promise in practical applications.

Sequential Least Squares Programming Sequential Least Squares Programming (SLSQP) is a popular general-purpose constrained nonlinear optimization method that uses a quadratic surrogate function to approximate the Lagrangian (Nocedal and Wright, 2006). In SLSQP, the surrogate function is a quadratic approximation of the Lagrangian of the original problem. The domain of the original problem is also

relaxed so that the constraint cuts it generates are approximated by linear functions. Like variational Bayes, SLSQP iterates between fitting the surrogate function and optimizing over the decision variables. Quadratic surrogate functions have also been investigated in the context of variational Bayes for nonconjugate models (Braun and McAuliffe, 2010; Wang and Blei, 2013).

Lagrangian Relaxation with Convex Optimization Johnson et al. (2007) propose a general framework for MAP estimation in graphical models using Lagrangian relaxation techniques. The approach combines convex optimization with dynamic programming techniques applicable for thin graphs. The popular tree-reweighted max-product (TRMP) method is used to solve such relaxations, where the intractable graph is relaxed to a set of spanning trees. Johnson et al. (2007) also propose a new class of multi-scale relaxations that introduce *summary* variables. The potential benefits of such generalizations are reducing or eliminating the *duality gap*, reducing the number of Lagrange multipliers in the dual problem, and accelerating the convergence of the iterative optimization procedure. However, the approach is developed for discrete variable and Gaussian graphical models only, where the problem is *convex decomposable*, and is not able to tell when the duality gap exists or not.

Linear Programming Relaxations Linear programming (LP) relaxations are a popular approach to approximate combinatorial optimization problems, and have become powerful tools for MAP configuration in graphical models. LP relaxations come with an optimality guarantee – if the solution to the linear program is integral, it is guaranteed to provide the global optimum of the MAP problem. These relaxations can be solved efficiently using message-passing algorithms such as belief propagation and its generalizations (Globerson and Jaakkola, 2007; Wainwright et al., 2005; Weiss et al., 2007). The standard LP relaxation is not tight enough in most practices, however, and this has led to the use of higher order cluster-based LP relaxations. The

computational cost grows exponentially with the size of the clusters and limits the number and type of clusters we can use. Sontag et al. (2008) propose to solve the cluster selection problem monotonically in the dual LP, iteratively selecting clusters with guaranteed improvement and quickly re-solving with the added clusters by reusing the existing solution. Yet, Sontag et al. (2008) focuses on *pairwise* Markov random fields rather than the general graphical models.

4.4 Our Contributions

- We provide an algorithm for maximum a-posteriori inference in *general* Bayesian models.
- The algorithm guarantees of ϵ-convergence by tightening the gap between upper and lower bounds of the problem.
- We present a theorem showing when the algorithm yields an ϵ-globally optimal solution, and when it may yield a locally optimal solution.
- The algorithm enables the incorporation of hard domain constraints appropriate to the model and produces feasible points in the posterior.

4.5 Generalized Benders' Decomposition for Factor Graphs

Generalized Benders' decomposition(Geoffrion, 1972) is used for solving problems of the form

$$\begin{aligned} &\max_{x,y} f(x,y) \\ &\text{subject to } G(x,y) \geq 0, \quad x \in \mathcal{X}, y \in \mathcal{Y}, \end{aligned} \tag{4.6}$$

to obtain the globally optimal solution where the following conditions on the objective and constraints hold:

1. for a fixed y, $f(x,y)$ separates into independent optimization problems each involving a different subvector of x,

2. for a fixed y, $f(x, y)$ is of a special structure that can be solved efficiently, and

3. fixing y renders the optimization problem convex in x.

The constraint function $G(x, y)$ captures all of the constraints that involve *both* x and y, while the constraints $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ capture the constraints that involve either variable but not both simultaneously.

Theorem 4. If the posterior distribution is convex in duplicated factor variables for fixed original latent variables, maximum a-posteriori inference for Bayesian models in factor graph form satisfies the conditions of generalized Benders' decomposition, and thus global optimality can be achieved.

Proof. The factor graph representation of maximum a-posteriori inference for a Bayesian model is

$$\begin{aligned} \text{MAP}(\theta) &= \max_{\mathbf{x}, \mathbf{x}^{\mathcal{F}}} \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \theta_f(\mathbf{x}_v^f) \\ \text{subject to} \quad & x_v = x_v^f, \quad \forall v \in f, f \in \mathcal{F}. \end{aligned}$$

Let $x^{\mathcal{F}}$ in the MAP inference problem be x in the generalized Benders' decomposition problem (equation 4.6), and let $\mathbf{x}$ in the MAP inference problem be y in the generalized Benders' decomposition problem.

Condition 1 is satisfied because if $\mathbf{x}$ in the MAP problem is held fixed, the maximizatation is over a sum of factors, each of which is a subvector of $\mathbf{x}^{\mathcal{F}}$. If $\mathbf{x}$ is fixed, the optimization problem is trivial because the constraints $x_v = x_v^f, \quad \forall v \in f, f \in \mathcal{F}$ set the value of $\mathbf{x}^{\mathcal{F}}$; finally, if the posterior is convex in the duplicated latent variables $\mathbf{x}^{\mathcal{F}}$ for fixed $\mathbf{x}$, condition 3 is satisfied. □

Corollary 2. If the posterior is not convex in duplicated factor variables for fixed original latent variables, maximum a-posteriori inference for Bayesian models in factor

graph form do not satisfy the conditions of generalized Benders' decomposition, and thus local optimality can be achieved.

Proof. The dual problem is equivalent to the projection of the Lagrangian form only when the condition 3 of GBD holds true. If false, a dual gap may exist between the optimal value of the Lagrangian form and the dual, and the dual problem can only provide a lower bound on the optimum. □

Generalized Benders' decomposition separates out the *complicating* variables in the optimization problem with the idea that holding these variables fixed renders the optimization problem much simpler. In the Bayesian factor graph model formulation, the original latent variables $\mathbf{x}$ are construed as complicating variables.

4.5.1 Branch-and-Bound

When GBD is not guaranteed to provide the global optimum according to Theorem 4 and provides a local optimum, the solution of the method may be heavily influenced by the initial points when fixing x in equation 4.6. To obtain favorable initial points, we use branch-and-bound method.

Branch-and-bound method, first proposed by Land and Doig (1960), has seen the most success in many practical optimization problems featuring combinatorial structure. It performs an exhaustive search over the space of assignments. It works by iteratively updating a nonincreasing upper bound and a nondecreasing lower bound until the bounds are within ϵ, when a certificate of optimality is obtained. The upper bound is updated by a feasible solution of problem and the lower bound is updated by solving a relaxation of problem. The branch-and-bound strategy fixes integer decision variables or bounds continuous variables at each node of the branch-and-bound tree. At a given node of the branch-and-bound search tree, if the lower bound is greater than the best available upper bound (for minimization problems), the subtree is fathomed, that is, excluded from the search space. Branch-and-bound solves the original problem

with integer variables by solving a sequence of relaxations that partition the search space. The branch-and-bound algorithm provides for the opportunity to exclude large portions of the search space at each iteration, and if the branching strategy is well-suited to the problem it can substantially reduce the actual computation time for real problems (Grötschel, 2012).

4.5.2 Algorithm

The algorithm of our proposed method is described in Algorithm 1. This algorithm selects the *most violated constraint* as assessed by the Lagrange multiplier and adds that constraint to the master problem. In this way, Benders' decomposition addresses the challenge of selecting constraints in the variational approximation of factor graph models.

4.6 Bayesian Gaussian Mixture Model

4.6.1 Problem Formulation

A Bayesian Gaussian mixture model (BGMM) has the form:

$$
\begin{aligned}
y_i|z_i,\mu,\Lambda &\sim \text{Gaussian}(\mu_{z_i},\Lambda_{z_i}^{-1}) && (4.9)\\
z_i|\pi &\sim \text{Categorical}_K(\pi) && (4.10)\\
\pi|\alpha_0 &\sim \text{Dirichlet}_K(\alpha_0) && (4.11)\\
\mu_k|\Lambda_k &\sim \text{Gaussian}(\mu_0,(\beta_0\Lambda_k)^{-1}) && (4.12)\\
\Lambda_k &\sim \text{Wishart}(W_0,\nu_0). && (4.13)
\end{aligned}
$$

The object of inference is the full set of latent variables $x = (z_1,\ldots,z_N,\mu_1,\ldots,\mu_K,$ $\Lambda_1,\ldots,\Lambda_K,\pi_1,\ldots,\pi_K)$. The value of z_i gives the cluster assignment for observation y_i, the values of μ_k and Λ_k give the cluster location and spread, and the value of π_k gives

Algorithm 1 GBD for factor graph models

1: Run a branch-and-bound (BNB) for a short, limited time starting from initial points.
2: Set an initial feasible point $\bar{\mathbf{x}}$ from the result of BNB
3: Solve the primal problem $(1 - \bar{\mathbf{x}})$,

$$\max_{\mathbf{x}^{\mathcal{F}}} \sum_{f \in \mathcal{F}} \theta_f(\mathbf{x}_f^f), \quad \text{s.t.} \quad \bar{x}_v = x_v^f, \ \forall v \in f, f \in \mathcal{F}. \tag{4.7}$$

Obtain the optimal multiplier vector $\bar{\mathbf{u}}$ and the function $L^*(\mathbf{x}, \bar{\mathbf{u}})$, where

$$L^*(\mathbf{x}; \bar{\mathbf{u}}) = \max_{\mathbf{x}^{\mathcal{F}}} \left\{ \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \theta_f(\mathbf{x}^f) + \bar{\mathbf{u}}^T G(\mathbf{x}, \mathbf{x}^{\mathcal{F}}) \right\}.$$

Since the constraints are all of the form $x_v = x_v^f$, the function L^* has the following decomposition

$$\sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \sum_{v \in f} \bar{u}_{x_v^f} x_v + \max_{\mathbf{x}^{\mathcal{F}}} \left\{ \sum_{f \in \mathcal{F}} \theta_f(\mathbf{x}^f) - \sum_{f \in \mathcal{F}} \sum_{v \in f} \bar{u}_{x_v^f} x_v^f \right\}. \tag{4.8}$$

Since the objective is separable, the optimization problem can be solved efficiently.
Set $p = 1$, $u^1 = \bar{u}$, and LBD $= v(\bar{x})$.
4: **for** *epoch* $= 1$ to N **do**
5: **for** *iter* $= 1$ to N **do**
6: Solve the *relaxed master problem*

$$\begin{aligned} \max_{\mathbf{x}, x_0} \quad & x_0 \\ \text{subject to} \quad & x_0 \ \leq L^*(\mathbf{x}, \mathbf{u}^j), \quad j = 1, \ldots, p \end{aligned}$$

Let $(\hat{\mathbf{x}}, \hat{x}_0)$ be an optimal solution; $\hat{x}_0$ is an upper bound on the optimal value of (4.6). If LBD $\geq \hat{x}_0 - \epsilon$, then terminate.
Integer cuts may be added to avoid previous solutions if $\mathbf{x}$ includes binary variables.
7: Solve the revised *primal problem* $(1 - \hat{\mathbf{x}})$.
If $v(\hat{\mathbf{x}}) \geq x_0 - \epsilon$, then terminate.
Otherwise, determine the optimal Lagrange multiplier $\hat{\mathbf{u}}$, and the function $L^*(\mathbf{x}, \hat{\mathbf{u}})$. Increment p by 1 and set $\mathbf{u}^p = \hat{\mathbf{u}}$. If $v(\hat{\mathbf{x}}) >$ LBD, update the lower bound, LBD $= v(\hat{\mathbf{x}})$.

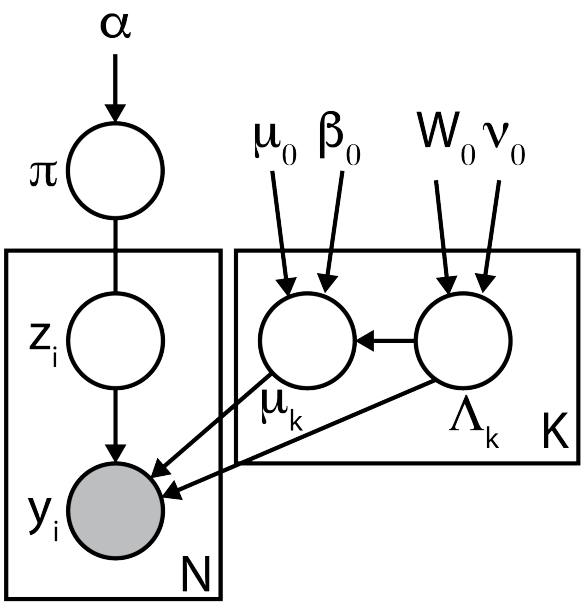

Figure 4.1: Graphical model representation of the BGMM model.

the relative proportion of the data in cluster k. The prior, $p(x|\phi)$, is parameterized by $\phi = (\alpha_0, \mu_0, \beta_0, W_0, \nu_0)$. Given this model, the posterior density function factorizes as

$$p(x|y,\phi) \propto p(x,y,\phi) = p(y|z,\mu,\Lambda)p(z|\pi)p(\pi;\alpha_0)p(\mu|\Lambda;\mu_0,\beta_0)p(\Lambda;W_0,\nu_0), \quad (4.14)$$

and can be represented as a factor graph (for full derivations, see Appendix B.1).

A graphical representation of the factor graph is shown in Figure 4.2.

Theorem 5. The posterior of BGMM as a factor graph is not convex in the latent variables.

Proof. Let $x^{\mathcal{F}}$ and $\mathbf{x}$ in the MAP inference problem be x and y in the generalized Benders' decomposition problem respectively. Assume that $\mathbf{x}$ is fixed. If $\mathbf{x}$ is fixed, the Hessian matrix, $\mathbf{H}$, is

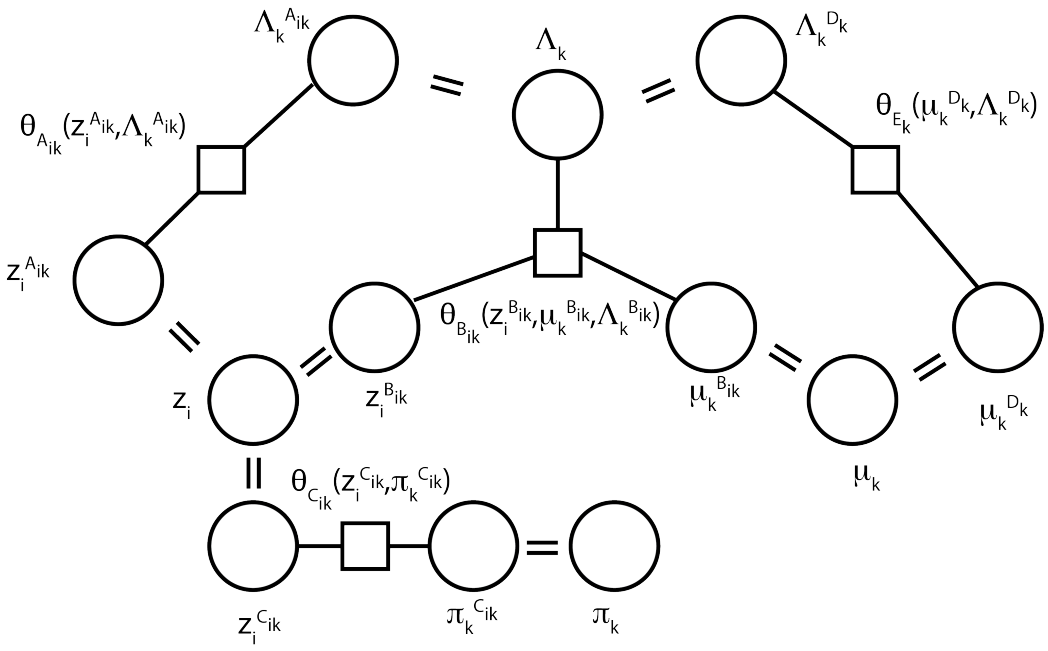

Figure 4.2: Factor graph representation of the BGMM model.

$$
\begin{array}{c}
 \\
z_{ik}^{A_{ik}} \\
\Lambda_k^{A_{ik}} \\
z_{ik}^{B_{ik}} \\
\mu_k^{B_{ik}} \\
\Lambda_k^{B_{ik}} \\
z_{ik}^{C_{ik}} \\
\pi_k^{C_{ik}} \\
\mu_k^{D_{ik}} \\
\Lambda_k^{D_{ik}}
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
z_{ik}^{A_{ik}} & \Lambda_k^{A_{ik}} & z_{ik}^{B_{ik}} & \mu_k^{B_{ik}} & \Lambda_k^{B_{ik}} & z_{ik}^{C_{ik}} & \pi_k^{C_{ik}} & \mu_k^{D_{ik}} & \Lambda_k^{D_{ik}}
\end{array} \\
\begin{pmatrix}
0 & (1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
(1) & (2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & (3) & (5) & 0 & 0 & 0 & 0 \\
0 & 0 & (3) & (4) & (6) & 0 & 0 & 0 & 0 \\
0 & 0 & (5) & (6) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & (7) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & (7) & (8) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & (9) & (10) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & (10) & 0
\end{pmatrix}
\end{array}
\text{, where}
$$

$$
\begin{aligned}
(1) &= \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\log(adj\Lambda_k^{A_{ik}}) - (y_i y_i^T) \\
(2) &= \frac{\partial}{\partial\Lambda_k^{A_{ik}}}\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}^{A_{ik}}(\log(adj\Lambda_k^{A_{ik}}) - (y_i y_i^T)) \\
(3) &= \sum_{i=1}^{N}\sum_{k=1}^{K}\Lambda_k^{B_{ik}} y_i - {\mu_k^{B_{ik}}}^T\Lambda_k^{B_{ik}} \\
(4) &= -\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}^{B_{ik}}\Lambda_k^{B_{ik}} \\
(5) &= \sum_{i=1}^{N}\sum_{k=1}^{K}{\mu_k^{B_{ik}}}^T y_i - \frac{1}{2}(\mu_k^{B_{ik}}{\mu_k^{B_{ik}}}^T) \\
(6) &= \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}^{B_{ik}}(y_i - \mu_k^{B_{ik}}) \\
(7) &= \sum_{i=1}^{N}\sum_{k=1}^{K}\frac{1}{\pi_k^{C_{ik}}} \\
(8) &= -\sum_{i=1}^{N}\sum_{k=1}^{K}\frac{{z_{ik}^{C_{ik}}}^2}{\pi_k^{C_{ik}}} \\
(9) &= -\beta_0\sum_{k=1}^{K}\Lambda_k^{D_{ik}} \\
(10) &= -\beta_0\sum_{k=1}^{K}\mu_k^{D_{ik}} - \mu_0.
\end{aligned}
$$

Considering $\mathbf{H}$ as a block matrix, the eigenvalues of $\mathbf{H}$ are collections of each block's eigenvalues. Consider the first block as

$$
\mathbf{H_1} = \{\mathbf{H_{ij}} : \mathbf{i} \in \mathbf{1}, \mathbf{2}, \mathbf{j} \in \mathbf{1}, \mathbf{2}\} = \begin{bmatrix} 0 & (1) \\ (1) & (2) \end{bmatrix}.
$$

Then the characteristic equation is

$$
|\mathbf{H_1} - \lambda\mathbf{I}| = \left|\begin{bmatrix} 0 & (1) \\ (1) & (2) \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right| = \left|\begin{bmatrix} -\lambda & (1) \\ (1) & (2)-\lambda \end{bmatrix}\right| = \lambda^{\mathbf{2}} - (\mathbf{2})\lambda - (\mathbf{1})^{\mathbf{2}} = \mathbf{0},
$$

and the two eigenvalues are $\frac{(2)\pm\sqrt{(2)^2+4(1)^2}}{2}$. $\mathbf{H}$ is not negative definite, because one of the eigenvalues, $\frac{(2)+\sqrt{(2)^2+4(1)^2}}{2}$, is always greater than 0. Thus this violates the condition 3 of the generalized Bender's decomposition. □

Thus, by Corollary 2, we note that our proposed method may achieve a local optimality.

4.6.2 Experiments

We compare our proposed approach to variational Bayes and Gibbs sampler on standard clustering data sets over the MAP value and labels provided with the data sets. However, we note that labels are typically generated by human classification and they may be erroneous when the difference between groups is difficult to ascertain based on features alone. Our primary interest lies in achieving a measure of convergence to the global/local optimum while being comparable with other methods. Hence, we also include the variation of information (Meila, 2007) as a clustering comparison metric.

Variation of information. Let a clustering solution is $C = \{C_1, \ldots, C_K\}$ where $C_k \subset \{1, \ldots, K\}$, $\bigcup_{k=1}^{K} C_k = \{1, \ldots, K\}$, and $C_i \cap C_j = \emptyset$ for $i \neq j$. Meila (2007) proposed an information theoretic criterion for comparing two clustering solutions. Variation of information (VI) measures the amount of information lost and gained when cluster changes from clustering C to clustering C':

$$VI(C, C') = H(C) + H(C') - 2I(C, C'), \tag{4.15}$$

where $H(C) = -\sum_{k=1}^{K} P_C(k) \log P_C(k)$ is the entropy associated with clustering C, and $I(C, C') = \sum_{k=1}^{K} \sum_{k'=1}^{K} P_{C,C'}(k, k') \log \frac{P_{C,C'}(k,k')}{P_C(k) P_{C'}(k')}$ is the mutual information between the clustering solutions. In the entropy and mutual information computations, $P_{C,C'}(k, k') = \frac{|C_k \cap C'_{k'}|}{n}$ is the joint probability that a point belongs to C_k and $C'_{k'}$, and $P_C(k) = \frac{|C_k|}{n}$ is proportion of points in cluster k. The higher the VI, the more distance there is between two clustering solutions. The VI can be bounded and normalized by the number of elements: $NVI(C, C') = \frac{VI(C,C')}{log(n)}$, where n is the number of elements clustered.

4.6.2.1 Data collection and preprocessing.

We obtained the Iris (`iris`, $n = 150, d = 4$), Wine Quality (`wine`, $n = 178, d = 13$), and Wisconsin Breast Cancer (`brca`, $n = 569, d = 30$) data sets from the UCI Machine

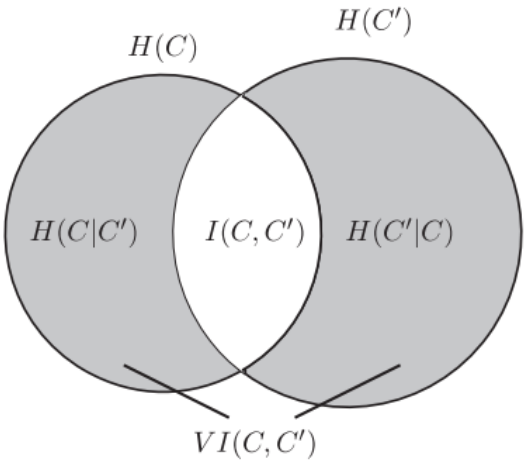

Figure 4.3: The variation of information (represented by the sum of the shaded areas) and related quantities (Meila, 2007).

Learning Repository (Dua and Graff, 2017). A 3-d projection of `iris` is obtained by projecting on the first three principal components c (designated `iris-3d`), a 6-d projection of `wine` is obtained by projecting on the first six principal components (designated `wine-6d`), and only the following features are employed for the `brca` data set: worst area, worst smoothness, and mean texture, then standardized (designated `brca-3d`). The reduction in dimensions is conducted to obtain not highly correlated (< 0.7) features but to keep the good portion of overall variability (> 0.85). Since our goal is to obtain the MAP clustering given the data set rather than a prediction, all of the data was used for clustering.

4.6.2.2 Experimental protocol.

We use K-means for initial points for BNB. We use the `scikit-learn` (Pedregosa et al., 2011) implementation of K-means and variational Bayes. We implement the Gibbs sampler for the Bayesian Gaussian mixture model using `bayesm` (Rossi, 2022). The K-means and variational Bayes experiments are run using algorithms defined in `python/scikitlearn`, and the Gibbs sampler experiments are run using algorithms defined in `R/bayesm`. The sampler is run for 10,000 iterations. Our approach was

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
variational	-403.403	0.071	0.667	(N/A)
Gibbs	-289.570	11.431	0.747	(N/A)
GBD	-291.215	847.491	0.827	epoch2, iter6

(a) `iris-3d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
variational	-1,322.345	0.077	0.927	(N/A)
Gibbs	-1,335.183	13.167	0.815	(N/A)
GBD	-1,384.721	3,018.171	0.972	epoch4, iter4

(b) `wine-6d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
variational	-1,836.016	0.073	0.900	(N/A)
Gibbs	-1,889.171	31.246	0.935	(N/A)
GBD	-1,881.576	24,181.185	0.905	epoch3, iter3

(c) `brca-3d`

Table 4.1: Comparisons among clustering solutions.

implemented in GAMS (Bisschop and Meeraus, 1982; GAMS Development Corporation, 2017), a standard optimization problem modeling language, with a solver BARON (Tawarmalani and Sahinidis, 2005; Sahinidis, 2017). For each problem solved using BARON, we limit the process to end in 600 seconds. Due to the complexity involved in calculating the determinant of the matrix in GAMS, we limit Λ_k to be a diagonal matrix.

4.6.2.3 Comparison to other algorithms.

Table 4.1 shows comparisons of our proposed method (GBD) with standard methods; variational Bayes and Gibbs sampler. It contains the log MAP value, running time (in seconds), match to labels (in proportion), and if the optimality is guaranteed.

log MAP. The value of log(MAP) is calculated with the estimated latent variables at the last step for variational Bayes and GBD. For Gibbs sampler, the value takes an expectation over 10,000 iterations after discarding the first 1,000 iterations as burn-in.

Looking at the first column in the Table 4.1, the clustering solution by GBD results in the second highest log(MAP) in `iris-3d` and `brca-3d`, and the last in `wine-6d`. Yet, GBD consistently shows log(MAP) values close to the highest log(MAP) achieved among all methods. We also want to highlight that the performance is achieved when Λ_k is limited to be a diagonal matrix when other methods are run without such a limit. On the other hand, variational Bayes seems to be stuck at a local optimum in `iris-3d` even though it performs well on other data sets.

Running time. The running time taken by each algorithm is recorded on the third column in the Table 4.1. In all data sets, the running time by algorithm increases in order of variational Bayes, Gibbs sampler and GBD. GBD takes a lot longer than other algorithms as it tries to solve each problem to its optimality. However, we want to emphasize that it takes far less than to prove optimality from equation 4.2 without using GBD. In fact, when we run simple branch-and-bound algorithm for 86,400 seconds on all three data sets, the gap between the lower bound and the upper bound does not close under 99%.

Match to labels. The proportion of clustering solution matched to labels provided by the data set is calculated on the third column in the Table 4.1. GBD scores the second place for `iris-3d`, the first place for `wine-6d` and the third place for `brca-3d`. In all cases, the difference to the first place is quite minimal (≤ 0.3). This helps us make the assertion that the GBD algorithm shows comparable performance to the variational Bayes and Gibbs sampler.

Optimality guaranteed. Whether the optimality is achieved by each algorithm and when if applicable is shown on the fourth column in the Table 4.1. Gibbs sampler

	K-means	variational	Gibbs	GBD
Label	0.149	0.092	0.193	0.152
K-means		0.092	0.208	0.035
variational			0.208	0.092
Gibbs				0.209

(a) iris-3d

	K-means	variational	Gibbs	GBD
Label	0.052	0.095	0.166	0.048
K-means		0.128	0.173	0.026
variational			0.173	0.125
Gibbs				0.187

(b) wine-6d

	K-means	variational	Gibbs	GBD
Label	0.093	0.101	0.075	0.080
K-means		0.084	0.074	0.110
variational			0.074	0.085
Gibbs				0.080

(c) brca-3d

Table 4.2: Comparisons among clustering solutions by normalized variation of information.

and variational Bayes do not provide finite global or the local optimality guarantee. On the other hand, our GBD method guarantees locally optimal solution by achieving an optimality gap of less than 0.01% for all data sets, and the column shows in which part of algorithm it is achieved.

Variation of information. Table 4.2 shows comparisons of our proposed method (GBD) with standard methods; K-means, variational Bayes, and Gibbs sampler. It contains the normalized variation of information (NVI) on three standard data sets for four algorithms under consideration. The first row where we compare the algorithms to the provided labels indicates that the information exchange - loss and gain of information as we go from the label to either of the four algorithms - is mostly similar.

The values in the fourth column of the Table 4.2 are the normalized variation of information between GBD and other algorithms, helping us make the assertion that the GBD shows similar performance to the K-means, variational Bayes, and Gibbs sampler. This is especially true for the `brca-3d` data set.

4.7 Latent Dirichlet Allocation

4.7.1 Problem Formulation

A latent Dirichlet allocation (LDA) model has the form:

$$\beta_k|\eta_0 \sim \text{Dirichlet}_V(\eta_0) \tag{4.16}$$

$$\Theta_d|\alpha_0 \sim \text{Dirichlet}_K(\alpha_0) \tag{4.17}$$

$$z_{dn}|\Theta_d \sim \text{Categorical}_K(\Theta_d) \tag{4.18}$$

$$w_{dn}|z_{dn}, \beta_k \sim \text{Categorical}_V(\beta_{z_{dn}}). \tag{4.19}$$

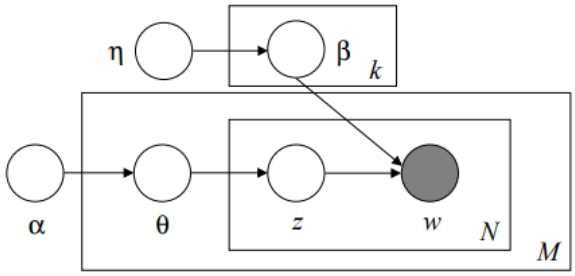

Figure 4.4: Graphical model representation of the smoothed LDA model.

The object of inference is the full set of latent variables $x = (z_{11}, \ldots, z_{MN}, \Theta_1, \ldots, \Theta_M, \beta_1, \ldots, \beta_K)$. The values of β_k are the distribution of words in topic k, the values of Θ_d are the probability distribution of topics in document d, and the value of z_{dn} gives the topic assignment of an observed n-th word in document d, w_{dn}. The prior, $p(x|\phi)$

is parameterized by $\phi = (\alpha_0, \eta_0)$. Given this model, the posterior density function factorizes as

$$p(x|w,\phi) \propto p(x,w,\phi) = p(w|z,\beta)p(z|\Theta)p(\Theta;\alpha_0)p(\beta;\eta_0) \tag{4.20}$$

and can be represented as a factor graph (for full derivations, see Appendix B.2).

A graphical representation of the factor graph is shown in Figure 4.5.

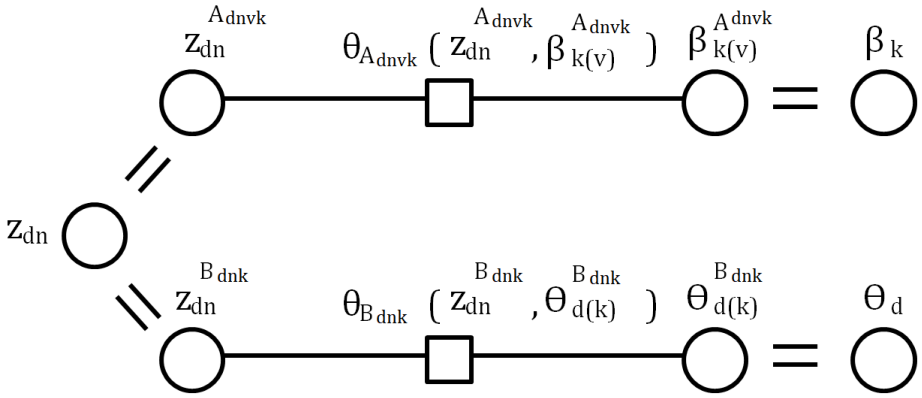

Figure 4.5: Factor graph representation of the LDA model.

Theorem 6. The posterior of LDA as a factor graph is not convex in the latent variables.

Proof. Let $x^{\mathcal{F}}$ and $\mathbf{x}$ in the MAP inference problem be x and y in the generalized Benders' decomposition problem respectively. Assume that $\mathbf{x}$ is fixed. If $\mathbf{x}$ is fixed, the Hessian matrix, $\mathbf{H}$, is

$$
\begin{array}{c}
 \\
z_{dn}^{A_{dnvk}} \\
\beta_{kv}^{A_{dnvk}} \\
\Theta_{dk}^{B_{dnk}} \\
z_{dn}^{B_{dnk}}
\end{array}
\begin{array}{c}
\begin{array}{cccc}
z_{dn}^{A_{dnvk}} & \beta_{kv}^{A_{dnvk}} & \Theta_{dk}^{B_{dnk}} & z_{dn}^{B_{dnk}}
\end{array} \\
\begin{pmatrix}
(1) & (2) & 0 & 0 \\
(2) & (3) & 0 & 0 \\
0 & 0 & (4) & (5) \\
0 & 0 & (5) & 0
\end{pmatrix}
\end{array}
\text{, where}
$$

$$
\begin{aligned}
(1) &= \sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{v=1}^{V} w_{dn}\left[\frac{z''_{dn}}{\beta_{z_{dn},v}} - (z'_{dn})^2\beta_{z_{dn},v}^{-2}\right] \quad \text{(For simplicity '}A_{dnk}\text{' is omitted)} \\
(2) &= -\sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{v=1}^{V} w_{dn}z'_{dn}\beta_{z_{dn},v}^{-2} \quad \text{(For simplicity '}A_{dnk}\text{' is omitted)} \\
(3) &= -\sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{v=1}^{V} w_{dn}\beta_{z_{dn},v}^{-2} \quad \text{(For simplicity '}A_{dnk}\text{' is omitted)} \\
(4) &= -\sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{z_{dn}^{B_{dnk}}}{{\Theta_{dk}^{B_{dnk}}}^2} \\
(5) &= \sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{1}{\Theta_{dk}^{B_{dnk}}}.
\end{aligned}
$$

Considering $\mathbf{H}$ as a block matrix, the eigenvalues of $\mathbf{H}$ are collections of each block's eigenvalues. Consider the second block as

$$
\mathbf{H_2} = \{\mathbf{H_{ij}} : \mathbf{i} \in \mathbf{3}, \mathbf{4}, \mathbf{j} \in \mathbf{3}, \mathbf{4}\} = \begin{bmatrix} (4) & (5) \\ (5) & 0 \end{bmatrix}.
$$

Then the characteristic equation is

$$
|\mathbf{H_2} - \lambda\mathbf{I}| = \left| \begin{bmatrix} (4) & (5) \\ (5) & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| = \left| \begin{bmatrix} (4)-\lambda & (5) \\ (5) & -\lambda \end{bmatrix} \right| = \lambda^{\mathbf{2}} - (\mathbf{4})\lambda - (\mathbf{5})^{\mathbf{2}} = \mathbf{0},
$$

and the two eigenvalues are $\frac{(4)\pm\sqrt{(4)^2+4(5)^2}}{2}$. $\mathbf{H}$ is not negative definite, because one of the eigenvalues, $\frac{(4)+\sqrt{(4)^2+4(5)^2}}{2}$, is always greater than 0. Thus this violates the condition 3 of the generalized Bender's decomposition. □

Thus, by Corollary 2, we note that our proposed method may achieve a local optimality.

4.7.2 Experiments

We compare our proposed approach to variational Bayes and Gibbs sampler on a standard topic modeling data set.

4.7.2.1 Data collection and preprocessing.

We obtain the 20 newsgroups (`news20`, $n = 18,846$, $d = 1$, type = text) dataset from the UCI KDD Archive (Hettich and Bay, 1999). We only look at randomly selected 300 text data from the original dataset, and run all methods given 5 topics and 150 most frequently used vocabulary within the selected data.

4.7.2.2 Experimental protocol.

We use variationaly Bayes for initial points for BNB, and use the `scikit-learn` (Pedregosa et al., 2011) implementation of variational Bayes. We implement the Gibbs sampler for latent Dirichlet allocation model based on (Griffiths and Steyvers, 2002). The variational Bayes experiment is run using algorithms defined in `python/scikitlearn`, and the Gibbs sampler experiments are run via `python`. The sampler is run for 1,000 iterations due to a fast convergence. Our approach was implemented in GAMS (Bisschop and Meeraus, 1982; GAMS Development Corporation, 2017), a standard optimization problem modeling language, with a solver BARON (Tawarmalani and Sahinidis, 2005; Sahinidis, 2017). For each problem solved using BARON, we limit the process to end in 600 seconds.

4.7.2.3 Comparison to other algorithms.

Top words assigned. Table 4.3 shows top 10 words assigned to 5 topics for each method: our proposed method (GBD), variational Bayes and Gibbs sampler.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
god	think	year	aids	edu
good	key	just	new	graphics
people	like	years	said	mail
just	government	car	health	send
like	chip	don	april	128
know	know	insurance	children	3d
gm	way	ve	10	com
does	don	windows	care	file
don	use	drive	1993	ftp
Israel	keys	state	people	objects

(a) Top 10 words assigned to each topic by variational Bayes

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
god	like	year	aids	edu
just	use	just	gm	graphics
good	key	problem	health	mail
like	Israel	years	children	send
people	government	don	april	128
think	bit	car	said	3d
don	chip	insurance	10	com
know	really	drive	new	file
doew	way	sure	care	ftp
better	want	ve	1993	objects

(b) Top 10 words assigned to each topic by Gibbs sampler

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
god	think	year	aids	edu
good	key	just	new	graphics
people	like	years	said	mail
just	government	car	health	send
like	chip	don	april	128
know	know	insurance	children	3d
gm	way	ve	10	com
does	don	windows	care	file
don	use	drive	1993	ftp
Israel	keys	state	people	objects

(c) Top 10 words assigned to each topic by GBD

Table 4.3: Comparisons of top 10 words assigned to each topic.

Variational Bayes and GBD show the same results; this is the case when the initial points given by variational Bayes are also the local optimum our algorithm finds (the optimality is found when epoch1, iter1). Gibbs sampler shows a bit different but similar results. Most words are assigned similarly yet ranked a bit differently within topics.

Running time. Running time comparison shows the same trend as in section 4.6. The running time by algorithm increases in order of variational Bayes (0.231s), Gibbs sampler (338.868s) and GBD (2,673.272s). GBD takes a lot longer than other algorithms as it tries to solve each problem to its optimality.

4.8 Discussion

In this section, we summarize the main contributions of the work, discuss some limitations, and suggest future work. First, We provide a computationally efficient algorithm for maximum a-posteriori inference in Bayesian models that yields an ϵ-global/local optimal value. The method sequentially adds optimal constraints to the fully relaxed dual problem using Benders' decomposition. Second, show when the algorithm yields an ϵ-globally optimal value and when it yields a locally optimal value. Third, we apply the method to two Bayesian factor models (BGMM and LDA) with incorporation of hard domain constraints to reflect the properties of the models. For each model, we implement the algorithm on standard data sets and compare the results to other methods.

4.8.1 Trade-off of adding hard constraints

As our problem is turned into an optimization problem, it allows the modeler to add hard constraints appropriately. For example, the modeler can force the solution has two data points being clustered into the same cluster, or into the different clusters based on prior knowledge. In such case, these constraints come useful as it is not

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD `min3`	-83.234	805.200	0.827	epoch2, iter46
GBD	-83.234	847.491	0.827	epoch2, iter6

(a) `iris-3d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD `min3`	-632.388	2,482.007	0.966	epoch4, iter40
GBD	-632.388	3,018.171	0.972	epoch4, iter4

(b) `wine-6d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD `min3`	-657.521	34,217.748	0.910	epoch3, iter41
GBD	-657.521	24,181.185	0.905	epoch3, iter3

(c) `brca-3d`

Table 4.4: Comparisons between with and without the `min3` constraint applied.

always easy to enforce such constrains using other standard methods. Yet, adding constraints may limit the domain space in a way that the solver runs less efficiently.

Table 4.4 shows how the method performs differently when the constraint that each cluster must have at least 3 points assigned (`min3`) is applied. For all three data sets, the method eventually reaches the same solution whether it has `min3` applied or not. With `brca-3d`, the method with `min3` takes about 42% more time and also more iterations to reach the local optimum. With `iris-3d` and `wine-6d`, though the method with `min3` takes less running time, it needs more iterations to reach the same solution of the model without `min3`.

As Table 4.4 illustrates the need of careful implementation of hard constraints when not required, the modeler needs to design the problem cautiously.

4.8.2 Initialization

In the algorithm described in section 4.5.2, we provide initial data points at two different steps: one by using branch-and-bound for generalized Benders' decomposition, and the other one for branch-and-bound. We discuss merits and limitations of each initialization here.

4.8.2.1 Initialization using branch-and-bound

When GBD is not guaranteed to provide the global optimum according to Theorem 4 and provides a local optimum, the solution of the method may be heavily influenced by the initial points when fixing x in equation 4.6). Thus it is important to provide good initial points to start with. We use branch-and-bound method for the purpose. Flaherty et al. (2019) shows that when running branch-and-bound, its upper bound converges very quickly to the global optimum, and it takes the majority of the time to (computationally) prove optimality within a predetermined ϵ threshold. GBD is more computationally efficient to prove the optimality so we use branch-and-bound to find the best solution it can possibly find within given relatively short time.

Table 4.5 shows how the method performs differently when GBD is run with and without an initialization through branch-and-bound. For all three data sets, the method with BNB provides more optimal solution with higher log MAP value. With `iris-3d` and `brca-3d`, the method with BNB takes fewer iterations and less time (excluding the time spent on BNB, 600 seconds).

While the modeler is welcomed to choose a different method for the initialization of GBD, as Table 4.5 illustrates the importance of initial points, the modeler needs to choose initial points with care.

4.8.2.2 Initialization of branch-and-bound

An upper bound of branch-and-bound relatively converges quickly to the global optimum (Flaherty et al., 2019). Yet, as we only run BNB for a relatively short time,

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD w/o BNB	-109.435	384.459	0.833	epoch2, iter26
GBD	-83.234	847.491	0.827	epoch2, iter6

(a) `iris-3d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD w/o BNB	-691.357	1817.026	0.966	epoch3, iter18
GBD	-632.388	3,018.171	0.972	epoch4, iter4

(b) `wine-6d`

	log MAP	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD w/o BNB	-733.567	36,927.063	0.910	epoch4, iter47
GBD	-657.521	24,181.185	0.905	epoch3, iter3

(c) `brca-3d`

Table 4.5: Comparisons between with and without the initialization using branch-and-bound.

it does not guarantee the upper bound it is at, at the end of the given time, is the global optimum. To run BNB more efficiently within the given time, therefore, we provide the initial solution. We use results of K-means for Bayesian Gaussian mixture model and those of variational Bayes for latent Dirichlet allocation model.

For relatively small data sets such as `iris-3d`, `wine-6d`, and `brca-3d`, Table 4.6 compares performance of K-means and BNB solutions. Though BNB method starts at the solution of K-means, within 600 seconds, it finds more optimal solutions, improving log MAP value from -270 to -83 for `iris-3d`, from -1,296 to 632 for `wine-6d`, and from -1,885 to -657 for `brca-3d`.

For relatively big data set with more variables such as `news20`, BNB fails to find a better solution than the initial points within the given time, and thus GBD also starts at the same initial points. We show how important initial points are in section 4.8.2.2.

	log MAP	Running time (s)	Match to labels
K-means	-270.289	0.028	0.833
BNB	-83.237	600	0.827

(a) `iris-3d`

	log MAP	Running time (s)	Match to labels
K-means	-1,296.599	0.038	0.966
BNB	-632.388	600	0.972

(b) `wine-6d`

	log MAP	Running time (s)	Match to labels
K-means	-1,885.084	0.048	0.910
BNB	-657.522	600	0.905

(c) `brca-3d`

Table 4.6: Comparisons among clustering solutions.

While the modeler is welcomed to choose a different method for the initialization of BNB, the initialization of BNB can also influence where GBD starts, the modeler needs to choose initial points with care.

4.8.3 Iterations vs Iterations of iterations

In the algorithm described in section 4.5.2, we sequentially add optimal constraints to the problem 50 times to the fully relaxed problem and repeat the process 5 times. Here, we discuss why we choose to repeat smaller number of iterations instead of running one set of longer iterations.

Table 4.7 shows how the method performs differently when GBD is run with 150 iterations and with 5 sets of 50 iterations. For all three data sets, both methods find the same solution but the method with 150 iterations fails to close the optimality gap. When all iterations are run, the optimality gaps range from 15.4% to 67.5% across the data sets. It also takes longer time to run. For `brca-3d`, the method with 150 iterations is terminated when its running time goes over 50,000 seconds.

	log MAP	Running time (s)	(Local) optimality guaranteed at
GBD, 150 iterations	-83.234	40,458.203	(optimality gap: 0.180)
GBD, 5 x 50 iterations	-83.234	847.491	epoch2, iter6

(a) `iris-3d`

	log MAP	Running time (s)	(Local) optimality guaranteed at
GBD, 150 iterations	-632.388	16,693.464	(optimality gap: 0.154)
GBD, 5 x 50 iterations	-632.388	3,018.171	epoch4, iter4

(b) `wine-6d`

	log MAP	Running time (s)	(Local) optimality guaranteed at
GBD, 150 iterations	-657.521	50,000+	(optimality gap: 0.675)
GBD, 5 x 50 iterations	-657.521	24,181.185	epoch3, iter3

(c) `brca-3d`

Table 4.7: Comparisons between 150 iterations and 5 sets of 50 iterations.

The problem becomes more complex when more constraints are added. That means each iteration takes longer and longer to be completed, or in some cases, the problem cannot be solved within the given time limit. Thus instead of continuously adding more of new constraints to the problem, after 50 iterations, we update existing constraints. Also, for problems where integer cuts are used, we remove all the integer cuts from the previous iterations (except the one with the best solution in the set) and start over. The updated constraints and newly added integer cuts (when applicable) are likely to be more relevant to the current stage of problem as well.

While the modeler is welcomed to adjust the number of iterations, as Table 4.7 illustrates the benefits of running sets of iterations, we encourage modeler to run sets of iterations instead of running one set of iterations.

	Running time (s)	Match to labels	(Local) optimality guaranteed at
GBD, $\beta_0 = 0.1$	384.459	0.833	epoch2, iter26
GBD, $\beta_0 = 1$	491.391	0.833	epoch2, iter27
GBD, $\beta_0 = 10$	1,393.085	0.666	(optimality gap: 0.159)

(a) `iris-3d`

Table 4.8: Comparisons among clustering solutions with different values of β_0. For this set of experiments, GBD is run without BNB initialization.

4.8.4 Regularization

In Bayesian mixture models, it is important to set appropriate values for parameters of prior probability distributions for parameter estimation and inference on posterior probability distribution. We can determine them with prior knowledge or based on common practices in literature. Yet, when neither is available or when the data set is significantly different from the one in literature, the modeler needs to choose the prior parameter values that are appropriate for the model and for the data set.

Table 4.8 shows how the method performs differently for different values of β_0. With `iris-3d`, compared to the method with $\beta_0 = 0.1$, the method with $\beta_0 = 1$ takes longer time and more iterations to reach a similarly performing solution (in terms of matching rate to labels given by the data set). The method with $\beta_0 = 10$ takes longer to run over all iterations, as it fails to close the optimality gap, resulting in a poor solution.

While the modeler is welcomed to choose different values of model parameters, as Table 4.5 illustrates how different values of a model parameter impacts on model performance, the modeler needs to choose parameter values with care and is advised to adjust them through experiments when no prior knowledge is available.

CHAPTER 5

CLUSTER TRELLIS: DATA STRUCTURES & ALGORITHMS FOR EXACT INFERENCE IN HIERARCHICAL CLUSTERING

Note: This chapter is part of the published work in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (Macaluso et al., 2021). Macaluso et al. and I are co-authors of the paper.

Section 5.2, section 5.3 and section 5.4 provide theoretical background of the work. They were mainly written by other co-authors of the paper but were included in the thesis for better understanding of my contributions. Section 5.5 contains my contributions in to the development of algorithms and their applications in cancer genomics, which was also published in that paper.

5.1 Abstract

Hierarchical clustering is a fundamental task often used to discover meaningful structures in data, such as phylogenetic trees, taxonomies of concepts, subtypes of cancer, and cascades of particle decays in particle physics. Typically approximate algorithms are used for inference due to the combinatorial number of possible hierarchical clusterings. In contrast to existing methods, we present novel dynamic-programming algorithms for *exact* inference in hierarchical clustering based on a novel trellis data structure. Our algorithms scale in time and space proportional to the powerset of N elements, which is super-exponentially more efficient than explicitly considering each of the $(2N-3)!!$ possible hierarchies. Exact methods are relevant to data analyses in particle physics and for finding correlations among gene expression in cancer genomics.

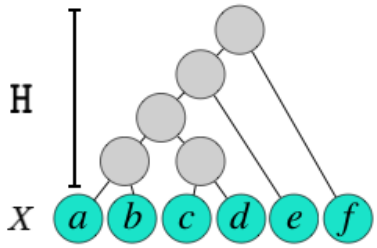

Figure 5.1: **Schematic representation of a hierarchical clustering**. H denotes the hierarchical clustering and X the dataset.

5.2 Introduction

Hierarchical clustering is often used to discover meaningful structures, such as phylogenetic trees of organisms (Kraskov et al., 2005), taxonomies of concepts (Cimiano and Staab, 2005), subtypes of cancer (Sørlie et al., 2001), and jets in particle physics (Cacciari et al., 2008). Among the reasons that hierarchical clustering has been found to be broadly useful is that it forms a natural data representation of data generated by a Markov tree, i.e., a tree-shaped model where the state variables are dependent only on their parent or children.

We define a hierarchical clustering as a recursive splitting of a dataset of N elements, $X = \{x_i\}_{i=1}^N$ into subsets until reaching singletons. This can equivalently be viewed as starting with the set of singletons and repeatedly taking the union of sets until reaching the entire dataset. We show a schematic representation in Figure 5.1, where we identify each x_i with a leaf of the tree and the hierarchical clustering as H. Formally,

Definition 5.2.1. (*Hierarchical Clustering*[1]) Given a dataset of elements, $X = \{x_i\}_{i=1}^N$, a **hierarchical clustering**, H, is a set of nested subsets of X, s.t. $X \in$ H,

[1]We limit our exposition to binary hierarchical clustering. Binary structures encode more tree-consistent clusterings than k-ary (Blundell et al., 2010). Natural extensions may exist for k-ary clustering, which are left for future work.

$\{\{x_i\}\}_{i=1}^N \subset \mathtt{H}$, and $\forall X_i, X_j \in \mathtt{H}$, either $X_i \subset X_j$, $X_j \subset X_i$, or $X_i \bigcap X_j = \emptyset$. Further, $\forall X_i \in \mathtt{H}$, if $\exists X_j \in \mathtt{H}$ s.t. $X_j \subset X_i$, then $\exists X_k \in \mathtt{H}$ s.t. $X_j \bigcup X_k = X_i$.

Given a subset $X_L \in \mathtt{H}$, then X_L is referred to as a cluster in $\mathtt{H}$. When $X_P, X_L, X_R \in \mathtt{H}$ and $X_L \bigcup X_R = X_P$, we refer to X_L and X_R as children of X_P, and X_P the parent of X_L and X_R; if $X_L \subset X_P$ we refer to X_P as an ancestor of X_L and X_L a descendent of X_P.(We also denote the sibling of X_L, as $X_R = X_P \setminus X_L$.) For binary trees, the total number of possible pairs of siblings (X_L, X_R) for a parent with N elements is given by the Stirling number of the second kind $S(N, 2) = 2^{N-1} - 1$.

In our work, we consider an energy-based probabilistic model for hierarchical clustering. Our model is based on measuring the compatibility of all pairs of sibling nodes in a binary tree structure. Formally,

Definition 5.2.2. (*Energy-based Hierarchical Clustering*)

Let X be a dataset, $\mathtt{H}$ be a hierarchical clustering of X, let $\psi : 2^X \times 2^X \to \mathbb{R}^+$ be a potential function describing the compatibility of a pair of sibling nodes in $\mathtt{H}$, and let $\phi(X|\mathtt{H})$ be a potential function for the $\mathtt{H}$ structure. Then, the probability of $\mathtt{H}$ for the dataset X, $P(\mathtt{H}|X)$, is equal to the unnormalized potential of $\mathtt{H}$ normalized by the partition function, $Z(X)$:

$$P(\mathtt{H}|X) = \frac{\phi(X|\mathtt{H})}{Z(X)} \text{ with } \phi(X|\mathtt{H}) = \prod_{X_L, X_R \in \mathsf{sibs}(\mathtt{H})} \psi(X_L, X_R) \tag{5.1}$$

where $\mathsf{sibs}(\mathtt{H}) = \{(X_L, X_R) | X_L \in \mathtt{H}, X_R \in \mathtt{H}, X_L \cap X_R = \emptyset, X_L \cup X_R \in \mathtt{H}\}$. The partition function $Z(X)$ is given by:

$$Z(X) = \sum_{\mathtt{H} \in \mathcal{H}(X)} \phi(X|\mathtt{H}). \tag{5.2}$$

where $\mathcal{H}(X)$ represents all binary hierarchical clusterings of the elements X.

We refer to our model as an energy-based model given that $\psi(\cdot, \cdot)$ is often defined by the unnormalized Gibbs distribution, i.e., $\psi(X_L, X_R) = \exp(-\beta E(X_L, X_R))$, where β is the inverse temperature and $E(\cdot, \cdot)$ is the energy. This probabilistic model allows us to express many familiar distributions over tree structures. It also has a connection

to the classic algorithmic hierarchical clustering technique, agglomerative clustering, in that $\psi(\cdot, \cdot)$ has the same signature as a "linkage function" (i.e., single, average, complete linkage).

Often, probabilistic approaches, such as coalescent models (Teh et al., 2008; Boyles and Welling, 2012; Hu et al., 2013) and diffusion trees (Neal, 2003; Knowles and Ghahramani, 2011), model which tree structures are likely for a given dataset. For instance, in particle physics generative models of trees are used to model jets (Cacciari et al., 2008), and similarly coalescent models have been used in phylogenetics (Suchard et al., 2018). Inference in these approaches is done by approximate, rather than exact, methods that lead to local optima, such as greedy best-first, beam-search, sequential Monte Carlo (Wang et al., 2015), and MCMC (Neal, 2003). Also, these methods do not have efficient ways to compute an exact normalized distribution over all tree structures.

5.3 Background: Hierarchical Cluster Trellis

Exactly performing MAP inference and finding the partition function by enumerating all hierarchical clusterings over N elements is intractable since the number of hierarchies grows extremely rapidly, namely $(2N-3)!!$ (Callan, 2009; Dale and Moon, 1993), where !! is double factorial. To address this challenge, we introduce a cluster trellis data structure for hierarchical clustering. The cluster trellis, inspired by (Greenberg et al., 2018), enables us to use dynamic programming algorithms to exactly compute MAP structures and the partition function, as well as compute marginal distributions, including the probability of any sub-hierarchy or cluster.

5.3.1 Trellis Data Structure

The trellis data structure is a directed acyclic graph that encodes a set of hierarchical clusterings. Each vertex in the trellis corresponds to a node in a hierarchical clustering,

and edges between vertices in the trellis correspond to a parent/child relationship in a hierarchical clustering. As in a hierarchical clustering, the trellis has a root node, that corresponds to the entire dataset, and leaf nodes that correspond to the individual elements of the dataset. The dataset associated with a trellis vertex $\mathbb{V}$ is denoted $X(\mathbb{V})$ and the trellis vertex associated with a dataset X is denoted $\mathbb{V}(X)$. Each vertex in the trellis stores memoized values of $Z(\mathbb{V})$ for computing the partition function, as well as the value $\phi(\mathtt{H}^*[\mathbb{V}])$ and the backpointer $\Xi(\mathtt{H}^*[\mathbb{V}])$ for computing the MAP tree. We denote $\mathbb{C}(X)$ as the children of $\mathbb{V}(X)$. We refer to a full trellis as the data structure where every possible hierarchical clustering given a dataset X can be realised, i.e., there is a bijection between the set of trellis vertices and $\mathbb{P}(X)\backslash\varnothing$, where $\mathbb{P}$ indicates the power set, and there is an edge between $\mathbb{V}_i$ and $\mathbb{V}_j$ if $X(\mathbb{V}_i) \subset X(\mathbb{V}_j)$.

5.3.2 Computing the Partition Function

Given a dataset of elements, $X = \{x_i\}_{i=1}^N$, the partition function, $Z(X)$, for the set of hierarchical clusterings over X, $\mathcal{H}(X)$, is given by Equation 5.2. The trellis implements a memoized dynamic program to compute the partition function and the MAP. To achieve this, we need to re-write the partition function in the corresponding recursive way.

Proposition 1. For any $x \in X$, the hierarchical partition function can be written recursively, as $Z(X) = \sum_{\mathtt{H}\in\mathcal{H}(X)} \phi(X|\mathtt{H}) = \sum_{X_i\in\mathbb{C}(X)_x} \psi(X_i, X \setminus X_i) \cdot Z(X_i) \cdot Z(X \setminus X_i)$ where $\mathbb{C}(X)_x$ is the set of all children of X containing the element x, i.e,. $\mathbb{C}(X)_x = \{X_j : X_j \in \mathbb{C}(X) \wedge x \in X_j\}$. In the particular case of a full trellis, then $\mathbb{C}(X)_x = \{X_j : X_j \in 2^X \setminus X \wedge x \in X_j\}$.

The proof is given in § C.1 in the Appendix. Algorithm 2 describes in a recursive way how to efficiently compute the partition function using the trellis based on Proposition 1. We first set the partition function of the leaf nodes in the trellis to 1. Then, we start by selecting any element in the dataset, x_i, and consider all clusters

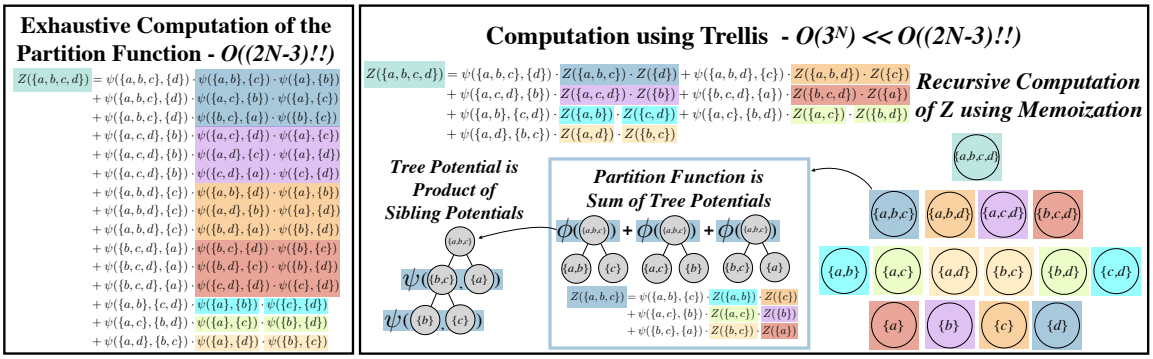

Figure 5.2: **Computing the partition function for the dataset** $\{a, b, c, d\}$. Left: exhaustive computation, consisting of the summation of $(2 \cdot 4 - 3)!! = 15$ energy equations. Right: computation using the trellis. The sum for the partition function is over $2^{4-1} - 1 = 7$ equations, each making use of a memoized Z value. Colors indicate corresponding computations over siblings in the trellis.

$X_i \in \mathbb{C}(X)$ such that $x_i \in X_i$. Next, the partition function is computed (memoized, recursively) for X_i and its complement $X \setminus X_i$, thus enabling the application of Proposition 1 to get $Z(X)$. For a full trellis, the algorithm can straightforwardly be written in a bottom-up, non-recursive way. In this case, the partition function for every node in the trellis is computed in order (in a bottom-up approach), from the nodes with the smallest number of elements to the nodes with the largest number of elements, memoizing the partition function value at each node. By computing the partial partition functions in this order, whenever computing the partition function of a given node in the trellis, the corresponding ones of all of the descendent nodes will have already been computed and memoized. In Figure 5.2, we show a visualization comparing the computation of the partition function with the trellis to the brute force method for a dataset of four elements. Next, we present the complexity result for Algorithm

Theorem 7. For a given dataset X of N elements, Algorithm 2 computes $Z(X)$ in $\mathcal{O}(3^N)$ time.

Algorithm 2 `PartitionFunction`(X)

Pick $x_i \in X$ and set $Z(X) \leftarrow 0$
for X_i in $\mathbb{C}(X)_{x_i}$ **do**
  if $Z(X_i)$ not set **then**
  $Z(X_i) \leftarrow$ `PartitionFunction`(X_i)
  if $Z(X \setminus X_i)$ not set **then**
  $Z(X \setminus X_i) \leftarrow$ `PartitionFunction`$(X \setminus X_i)$
  $Z(X) \leftarrow Z(X) + \psi(X_i, X \setminus X_i) \cdot Z(X_i) \cdot Z(X \setminus X_i)$
return $Z(X)$

The time-complexity of the algorithm is $\mathcal{O}(3^N)$, which is is significantly smaller than the $(2N-3)!!$ possible hierarchies.

Corollary 3. For a given dataset X of N elements, Algorithm 2 is super-exponentially more efficient than brute force methods that consider every possible hierarchy. In particular the ratio is $\mathcal{O}((\frac{2}{3})^N\ \Gamma(N-1/2))$.

The proofs of Theorem 7 and Corollary 3 are given in § C.2 of the Appendix.

5.3.3 Computing the MAP Hierarchical Clustering

Similar to other dynamic programming algorithms, we can adapt Algorithm 2 in order to find the MAP hierarchical clustering.

The MAP clustering for dataset X, is $\mathtt{H}^\star(X) = \operatorname{argmax}_{\mathtt{H} \in \mathcal{H}(X)} \phi(\mathtt{H})$. Here we can also use a recursive memoized technique, where each node will store a value for the MAP, denoted by $\phi(\mathtt{H}^\star(X))$ and a backpointer $\Xi(\mathtt{H}^\star(X))$. Specifically,

Proposition 2. For any $x \in \mathbb{C}(X)$, let $\mathbb{C}(X)_x = \{X_j : X_j \in \mathbb{C}(X) \wedge x \in X_j\}$, then $\phi(\mathtt{H}^\star(X)) = \max_{X_i \in \mathbb{C}(X)_x} \psi(X_i, X \setminus X_i) \cdot \phi(\mathtt{H}^\star(X_i)) \cdot \phi(\mathtt{H}^\star(X \setminus X_i))$.

See §C.3 in the Appendix for the proof. As in the partition function algorithm described in Section 5.3.2, the time complexity for finding the MAP clustering is also $\mathcal{O}(3^N)$. The main difference is that to compute the maximal likelihood hierarchical clustering, the maximal energy of the sub-hierarchy rooted at each node is computed, instead of the partition function. Pointers to the children of the maximal sub-hierarchy

Algorithm 3 `MAP`(X)

if $\phi(X)$ set **then**
  return $\phi(X), \Xi(X)$
Pick $x_i \in X$
$\phi(X) \leftarrow -\infty$
$\Xi(X) \leftarrow$ `null` {Backpointer to give MAP tree structure.}
for X_i in $\mathbb{C}(X)_{x_i}$ **do**
  $t \leftarrow \psi(X_i, X \setminus X_i) \cdot \phi(\mathbb{V}(X_i)) \cdot \phi(\mathbb{V}(X \setminus X_i))$
  if $\phi(X) < t$ **then**
    $\phi(X) \leftarrow t$
    $\Xi(X) \leftarrow \{X_i, X \setminus X_i\} \cup \Xi(X_i) \cup \Xi(X \setminus X_i)$
return $\phi(X), \Xi(X)$

rooted at each node are stored at that node. A proof of the time complexity, analogous to the one for the partition function, can be found in §C.4 of the Appendix.

5.4 Related Work

Modeling distributions over tree structures has been the subject of a large body of work. Bayesian non-parametric models typically define a posterior distribution over tree structures given data such as diffusion trees coalescents, and others (Neal, 2003; Teh et al., 2008, inter alia). These methods, while providing a distribution over trees, only support using parametric distributions to define emission probabilities rather than the energy-based model used in this paper. The Bayesian hierarchical clustering (BHC) model (Heller and Ghahramani, 2005) is akin to the energy-based ones used in this paper. Inference includes greedy agglomerative (Heller and Ghahramani, 2005), randomized (Heller and Ghahramani, 2005), and tree re-arrangement approaches (Xu et al., 2009). Future work could consider how to use the trellis for BHC. Interestingly, the BHC likelihood is a mixture of tree consistent partitions, also related to using the trellis for flat clustering. Factor graph-based distributions over tree structures such as (Wick et al., 2012) on the other hand support a flexible class of distributions over tree structures as in our approach. However inference in factor graph models as well as many of the Bayesian non-parameteric models is typically approximate or performed

by sampling methods. This lends in practice to approximate MAP solutions and distributions over tree structures. Exact methods like the one proposed in this paper have not, to our knowledge, been proposed.

Dasgupta (2016) defines a cost function for hierarchical clustering. Much work has been done to develop approximate solution methods and related objectives (Moseley and Wang, 2017, inter alia).

Bootstrapping methods, such as (Suzuki and Shimodaira, 2006), represent uncertainty in hierarchical clustering. Unlike our approach, bootstrapping methods approximate statistics of interest through repeatedly (re-)sampling from the empirical distribution.

Work on exact inference and exact distributions over flat clusterings (Greenberg et al., 2018), provides the foundation of our dynamic programming approach. Other work on exact flat clustering uses fast convolutions via the Mobius transform and Mobius inversion (Kohonen and Corander, 2016). Kappes et al. (2015) produce approximate distributions over flat clusterings using Perturb and MAP (Papandreou and Yuille, 2011).

Orthogonal to our work on uncertainty in hierarchical clustering, recent work has proposed continuous representations of trees for hierarchical clustering (Monath et al., 2019; Chami et al., 2020). This work represents uncertainty of child-parent assignments by considering the distance between two nodes in embedding space. We note that the distribution over trees used in these papers does not directly correspond to the energy-based distribution proposed in our work.

5.5 My Contributions

I contribute to the development of algorithms and their applications in cancer genomics. By defining the energy of the algorithm to be the objective of correlation clustering, I run my exact MAP clustering tree on the Prediction Analysis of Microarray

50 (`pam50`) gene expression data set, and compare the result against that of a greedy agglomerative clustering.

5.5.1 Cancer Genomics

Hierarchical clustering is a common clustering approach for gene expression data (Sørlie et al., 2001). It is not uncommon to have a need for clustering a small number of samples in cancer genomics studies. An analysis of data available from `https://clinicaltrials.gov` shows that the median sample size for 7,412 completed phase I clinical trials involving cancer is only 30.

5.5.2 Probabilistic Model

In this case I am given a dataset of vectors indicating the level of gene expressions which are endowed with pairwise affinities that are both positive and negative. I define the energy of a pair of sibling nodes in the tree to be the sum of the across-cluster positive edges, minus the sum of negative within-cluster edge weights.

$$E(X_i, X_j) = \sum_{x_i, x_j \in X_i \times X_j} w_{ij} \mathbb{I}[w_{ij} > 0] - \sum_{\substack{x_i, x_j \in X_i \times X_i, \\ x_i < x_j}} w_{ij} \mathbb{I}[w_{ij} < 0] - \sum_{\substack{x_i, x_j \in X_j \times X_j, \\ x_i < x_j}} w_{ij} \mathbb{I}[w_{ij} < 0] \qquad (5.3)$$

where w_{ij} is the affinity between x_i and x_j.

The correlation clustering input can be represented as a complete weighted graph, $G = (V, E)$, where each edge has weight $w_{uv} \in [-1, 1], \forall (u, v) \in E$. The goal is to construct a clustering of the nodes that maximizes the sum of positive within-cluster edge weights minus the sum of all negative across-cluster edge weights (since I wish to minimize the energy function given by Equation 5.3). This energy is the correlation clustering objective (Bansal et al., 2004).

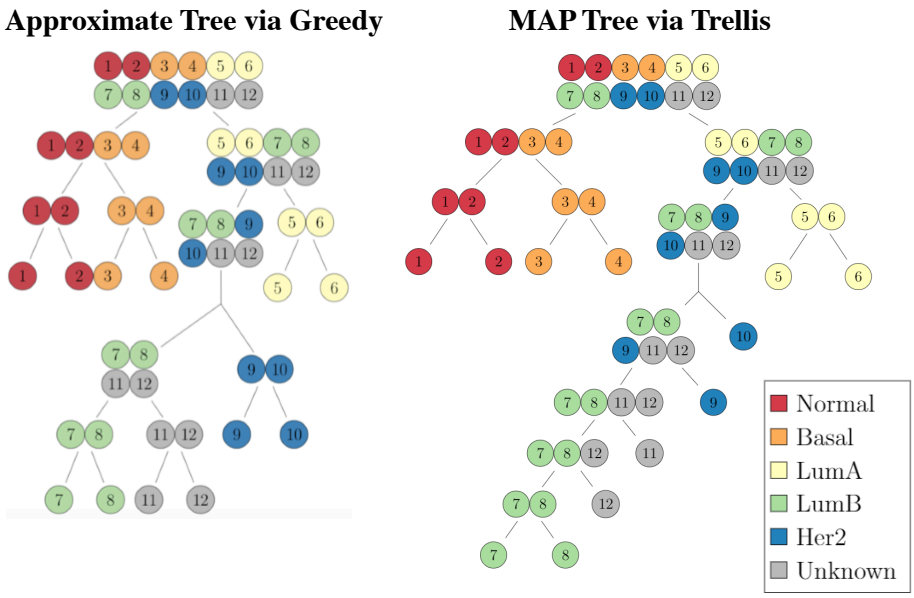

Figure 5.3: **Cancer Genomics**. Comparison of trees from greedy hierarchical clustering (left) and exact MAP clustering using the trellis (right) on the subsampled `pam50` data set. The colors indicate subtypes of breast cancer (grey if unknown). Though both appear to assign unknown samples to LumB, the right tree positions the unknown samples closer to the Her2 samples.

5.5.3 Data and Methods

I compare a greedy agglomerative clustering to my exact MAP clustering tree using the Prediction Analysis of Microarray 50 (`pam50`) gene expression data set. The `pam50` data set ($n = 232$, $d = 50$) is available from the UNC MicroArray Database (University of North Carolina, 2020). It has intrinsic subtype annotations for 139 of the 232 samples. Missing data values (2.65%) are filled in with zeros. I draw a stratified sample of the total data set with two samples from each known intrinsic subtype and two samples from the unknown group.

5.5.4 Results

Figure 5.3 displays the greedy hierarchical clustering tree and the MAP tree with transformed weights for the twelve samples selected from the pam50 dataset. (The correlations among subsampled `pam50` ($n = 12$) data set are all positive.) The main difference between these trees is in the split of the subtree including LumB, HER2,

and unknown samples. The greedy method splits HER2 from LumB and unknown, while the MAP tree shows a different topology for this subtree. For the MAP solution, I note that the subtree rooted at $\{7, 8, 9, 10, 11, 12\}$ is consistent. All of the correlation coefficients among this cluster are positive, so the optimal action is to split off the item with the smallest (positive) correlation coefficient.

5.6 Conclusion

This paper describes a trellis data structure and dynamic-programming algorithm to efficiently compute and sample from probability distributions over hierarchical clusterings. Our method improves upon the computation cost of brute-force methods from $(2N-3)!!$ to sub-quadratic in the substantially smaller powerset of N, which is super-exponentially more efficient. We demonstrate our method's utility on cancer genomics datasets and show its improvement over approximate methods.

APPENDIX A

SUPPLEMENTAL MATERIALS FOR CHAPTER 2

A.1 Proof of that ξ is concave in λ and ν

Lemma 4. ξ is concave in λ_{1k}.

Proof.

$$\begin{aligned}
\frac{\partial \xi}{\partial \lambda_{1k}} = \frac{\partial \xi}{\partial \hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} =& -\sum_{i=1}^{n} z_{ik}\frac{1}{\hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + \nu\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + (\hat{\pi}_k - 1) \\
&+ \lambda_{1k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} - \lambda_{2k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} \\
\frac{\partial^2 \xi}{\partial \lambda_{1k}^2} =& -\sum_{i=1}^{n} z_{ik}\left(\frac{1}{\hat{\pi}_k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} - \frac{1}{\sum_i z_{ik}}\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}}\right) + \nu\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} \\
&+ \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + \lambda_{1k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} + \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} - \lambda_{2k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} + \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} + \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} \\
=& 2\left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{1k}^2} + 3\frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^3} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \leq 0
\end{aligned}$$

□

Lemma 5. ξ is concave in λ_{2k}.

Proof.

$$\frac{\partial \xi}{\partial \lambda_{2k}} = \frac{\partial \xi}{\partial \hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} = -\sum_{i=1}^{n} z_{ik}\frac{1}{\hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} + \nu\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} + \lambda_{1k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} - \lambda_{2k}\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} - \hat{\pi}_k$$

$$\begin{aligned}
\frac{\partial^2 \xi}{\partial \lambda_{2k}^2} =& -\sum_{i=1}^{n} z_{ik}\left(\frac{1}{\hat{\pi}_k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} + \frac{1}{\sum_i z_{ik}}\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}}\right) + \nu\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} \\
&+ \lambda_{1k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} - \lambda_{2k}\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} - \frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} - \frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} - \frac{\partial \hat{\pi}_k}{\partial \lambda_{1k}} + \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} - \frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} - \frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} \\
=& 2\left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \lambda_{2k}^2} - 3\frac{\partial \hat{\pi}_k}{\partial \lambda_{2k}} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^3} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \leq 0
\end{aligned}$$

□

Lemma 6. ξ is concave in λ_{3k}.

Proof.

$$\frac{\partial \xi}{\partial \lambda_{3k}} = \left(\frac{1}{\sum_i z_{ik}}\right)\sum_i z_{ik} y_i - M_k^U - \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} + \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{4k}.$$

$$\frac{\partial^2 \xi}{\partial \lambda_{3k}^2} = -\left(\frac{1}{2\sum_i z_{ik}}\right) \leq 0$$

□

Lemma 7. ξ is concave in λ_{4k}.

Proof.

$$\frac{\partial \xi}{\partial \lambda_{4k}} = -\left(\frac{1}{\sum_i z_{ik}}\right)\sum_i z_{ik}y_i + M_k^L + \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{3k} - \left(\frac{1}{2\sum_i z_{ik}}\right)\lambda_{4k}.$$

$$\frac{\partial^2 \xi}{\partial \lambda_{4k}^2} = -\left(\frac{1}{2\sum_i z_{ik}}\right) \leq 0$$

□

Lemma 8. ξ is concave in ν.

Proof.

$$\frac{\partial \xi}{\partial \nu} = \frac{\partial \xi}{\partial \hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \nu} = -\sum_{i=1}^{n} z_{ik}\frac{1}{\hat{\pi}_k}\frac{\partial \hat{\pi}_k}{\partial \nu} + (\hat{\pi}_k - 1) + \nu\frac{\partial \hat{\pi}_k}{\partial \nu} + \lambda_{1k}\frac{\partial \hat{\pi}_k}{\partial \nu} - \lambda_{2k}\frac{\partial \hat{\pi}_k}{\partial \nu}$$

$$\begin{aligned}
\frac{\partial^2 \xi}{\partial \nu^2} =& -\sum_{i=1}^{n} z_{ik}\left(\frac{1}{\hat{\pi}_k}\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} - \frac{1}{\sum_i z_{ik}}\frac{\partial \hat{\pi}_k}{\partial \nu}\right) + \frac{\partial \hat{\pi}_k}{\partial \nu} \\
&+ \nu\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} + \frac{\partial \hat{\pi}_k}{\partial \nu} + \lambda_{1k}\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} - \lambda_{2k}\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} + \frac{\partial \hat{\pi}_k}{\partial \nu} + \frac{\partial \hat{\pi}_k}{\partial \nu} + \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} + \frac{\partial \hat{\pi}_k}{\partial \nu} \\
=& 2\left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{\partial^2 \hat{\pi}_k}{\partial \nu^2} + 3\frac{\partial \hat{\pi}_k}{\partial \nu} \\
=& \left(\lambda_{1k} - \lambda_{2k} + \nu\right)\frac{4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^3} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-4\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} + \frac{3\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \\
=& \frac{-\sum_{i=1}^{n} z_{ik}}{(\lambda_{2k} - \lambda_{1k} - \nu)^2} \leq 0
\end{aligned}$$

□

Theorem 8. ξ is concave in $\boldsymbol{\lambda}$ and ν.

Proof. By lemma A.1, 5, 6, 7, 8, ξ is concave in $\boldsymbol{\lambda}$ and ν. □

APPENDIX B

SUPPLEMENTAL MATERIALS FOR CHAPTER 3

B.1 BGMM posterior density in a factor graph form

Term 1 The density function of $p(y|z,\mu,\Lambda)$ is

$$p(y|z,\mu,\Lambda) = \prod_{i=1}^{N}\prod_{k=1}^{K}\left[\mathcal{N}(y_i;\mu_k,\Lambda_k^{-1})\right]^{z_{ik}}. \tag{B.1}$$

The log-density function is

$$\begin{aligned}
\log p(y|z,\mu,\Lambda) &= \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\left[-\frac{D}{2}\log(2\pi) - \frac{1}{2}\log\det\Lambda_k^{-1} - \frac{1}{2}(y_i-\mu_k)^T\Lambda(y_i-\mu_k)\right] \\
&+= \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\left[\frac{1}{2}\log\det\Lambda_k - \frac{1}{2}y_i^T\Lambda_k y_i + \mu_k^T\Lambda_k y_i - \frac{1}{2}\mu_k^T\Lambda_k\mu_k\right] \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\log\det\Lambda_k - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}y_i^T\Lambda_k y_i + \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\mu_k^T\Lambda_k y_i \\
&\quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\mu_k^T\Lambda_k\mu_k.
\end{aligned}$$

The first factor, $z_{ik}\log\det\Lambda_k$, only involves the latent variables $x_{A_{ik}} = \{z_{ik},\Lambda_k\}$. Grouping the terms from the density that only involve those terms gives

$$\theta_{A_{ik}}(x_{A_{ik}}) := \frac{1}{2}z_{ik}\left[\log\det\Lambda_k - y_i^T\Lambda_k y_i\right]. \tag{B.2}$$

Similarly, the term $z_{ik}\mu_k^T\Lambda_k y_i$ involves latent variables $x_{B_{ik}} = \{z_{ik},\mu_k,\Lambda_k\}$. Grouping all of the terms that involve $x_{B_{ik}}$ gives

$$\theta_{B_{ik}}(x_{B_{ik}}) := z_{ik}\left[\mu_k^T\Lambda_k y_i - \frac{1}{2}\mu_k^T\Lambda_k\mu_k\right]. \tag{B.3}$$

Now, the density function can be written in factor form as

$$\log p(y|z,\mu,\Lambda) = \sum_{f\in\mathcal{F}_1}\theta_f(x_f), \tag{B.4}$$

where $\mathcal{F}_1 = \{A_{ik},B_{ik}\}_{i=1,k=1}^{N,K}$.

Term 2. The log-density function of $p(z|\pi)$ is

$$\log p(z|\pi) = \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \log \pi_k. \tag{B.5}$$

The factors only involve latent variables $x_{C_{ik}} = \{z_{ik}, \pi_k\}$ and the factors are

$$\theta_{C_{ik}}(x_{C_{ik}}) := z_{ik} \log \pi_k. \tag{B.6}$$

The log-density function can then be written as

$$\log p(z|\pi) = \sum_{f \in \mathcal{F}_2} \theta_f(x_f), \tag{B.7}$$

where $\mathcal{F}_2 = \{C_{ik}\}_{i=1,k=1}^{N,K}$.

Term 3. The log-density function of $p(\pi|\alpha_0)$ is

$$\log p(\pi|\alpha_0) = \log \Gamma(\sum_{k=1}^{k} \alpha_{0k}) - \sum_{k=1}^{K} \log \Gamma(\alpha_{0k}) + \sum_{k=1}^{K} (\alpha_{0k} - 1) \log \pi_k. \tag{B.8}$$

The factors in this log-density only involve singleton latent variables, so they can be written as

$$\theta_{\pi_k} = (\alpha_{0k} - 1) \log \pi_k \tag{B.9}$$

and the log-density is

$$\log p(\pi|\alpha_0) = \sum_{v \in \mathcal{V}_1} \theta_v(v), \tag{B.10}$$

where $\mathcal{V}_1 = \{\pi_1, \ldots, \pi_K\}$.

Term 4. The log-density function is

$$\begin{aligned}
\log p(\mu|\Lambda; \mu_0, \beta_0) &= \sum_{k=1}^{K} \left[-\frac{D}{2} \log(2\pi) - \frac{1}{2} \log \det(\beta_0 \Lambda_k)^{-1} - \frac{\beta_0}{2} (\mu_k - \mu_0)^T \Lambda (\mu_k - \mu_0) \right] \\
&+= \sum_{k=1}^{K} \left[\frac{1}{2} \log \det(\beta_0 \Lambda_k) - \frac{\beta_0}{2} \mu_k^T \Lambda_k \mu_k + \beta_0 \mu_k^T \Lambda_k \mu_0 - \frac{\beta_0}{2} \mu_0^T \Lambda_k \mu_0 \right] \\
&= \frac{1}{2} \sum_{k=1}^{K} \log \det(\beta_0 \Lambda_k) - \frac{\beta_0}{2} \sum_{k=1}^{K} \mu_k^T \Lambda_k \mu_k + \beta_0 \sum_{k=1}^{K} \mu_k^T \Lambda_k \mu_0 \\
&\quad - \frac{\beta_0}{2} \sum_{k=1}^{K} \mu_0^T \Lambda_k \mu_0.
\end{aligned}$$

The first term only involves a single latent variable and the log-density can be reformulated by defining

$$\theta_{\Lambda_k} := \frac{1}{2} \log \det(\beta_0 \Lambda_k) - \frac{\beta_0}{2} \sum_{k=1}^{K} \mu_0^T \Lambda_k \mu_0 \tag{B.11}$$

The second term involves $x_{D_k} = \{\mu_k, \Lambda_k\}$. Grouping all the terms involving those latent variables gives

$$\theta_{D_k}(x_{D_k}) := -\frac{\beta_0}{2}\mu_k^T \Lambda_k \mu_k + \beta_0 \mu_k^T \Lambda_k \mu_0. \tag{B.12}$$

The log-density function can then be written as

$$\log p(\mu|\Lambda; \mu_0, \beta_0) = \sum_{v \in \mathcal{V}_2} \theta_v(x_v) + \sum_{f \in \mathcal{F}_3} \theta_f(x_f), \tag{B.13}$$

where $\mathcal{V}_2 = \{\Lambda_1, \ldots, \Lambda_K\}$ and $\mathcal{F}_3 = \{D_k\}_{k=1}^K$.

Term 5. The log-density function of $p(\Lambda)$ is

$$\log p(\Lambda; W_0, \nu_0) = \sum_{k=1}^K \log B(W_0, \nu_0) + \frac{(\nu_0 - D - 1)}{2} \log\det \Lambda_k - \frac{1}{2}(W_0^{-1}\Lambda_k),$$

where

$$B(W_0, \nu_0) = \left[\det W_0^{\nu_0/2} 2^{\nu_0 D/2} \pi^{D(D-1)/4} \prod_{d=1}^D \Gamma\left(\frac{\nu_0 + 1 - d}{2}\right)\right]^{-1}.$$

The factors in this log-density only involve singleton latent variables, so they can be written as

$$\theta_{\Lambda_k} += \frac{\nu_0 - D - 1}{2} \log\det \Lambda_k - \frac{1}{2}\left(W_0^{-1}\Lambda_k\right) \tag{B.14}$$

and the log-density is

$$\log p(\Lambda; W_0, \nu_0) = \sum_{v \in \mathcal{V}_2} \theta_v(v), \tag{B.15}$$

where $\mathcal{V}_2 = \{\Lambda_1, \ldots, \Lambda_K\}$.

Combining factors. Recall $\phi = (\alpha_0, W_0, \nu_0, \mu_0, \beta_0)$, $x = (z_{11}, \ldots, z_{NK}, \mu_1, \ldots, \mu_K, \Lambda_1, \ldots, \Lambda_K, \pi_1, \ldots, \pi_K)$, and $y = (y_1, \ldots, y_N)$. Combining the factorized representations of each of the terms in the joint log-density gives

$$p(x|y, \phi) = f(x, \theta) = \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \theta_f(x_f), \tag{B.16}$$

where $\mathcal{V} = \{z_{11}, \ldots, z_{NK}, \mu_1, \ldots, \mu_K, \Lambda_1, \ldots, \Lambda_K, \pi_1, \ldots, \pi_K\}$, and $\mathcal{F} = \{[A_{ik}]_{k=1,i=1}^{K,N}, [B_{ik}]_{k=1,i=1}^{K,N}, [C_{ik}]_{k=1,i=1}^{K,N}, [D_k]_{k=1}^K\}$. The factors are

$$\theta_{\pi_k}(\pi_k) = (\alpha_{0k} - 1)\log \pi_k, \tag{B.17}$$

$$\theta_{\Lambda_k}(\Lambda_k) = \frac{1}{2}\log\det(\beta_0\Lambda_k) - \frac{\beta_0}{2}\sum_{k=1}^{K}\mu_0^T\Lambda_k\mu_0 \tag{B.18}$$

$$+\frac{\nu_0 - D - 1}{2}\log\det\Lambda_k - \frac{1}{2}\left(W_0^{-1}\Lambda_k\right) \tag{B.19}$$

$$\theta_{A_{ik}}(x_{A_{ik}}) = \frac{1}{2}z_{ik}\left[\log\det\Lambda_k - y_i^T\Lambda_k y_i\right], \tag{B.20}$$

$$\theta_{B_{ik}}(x_{B_{ik}}) = z_{ik}\left[\mu_k^T\Lambda_k y_i - \frac{1}{2}\mu_k^T\Lambda_k\mu_k\right], \tag{B.21}$$

$$\theta_{C_{ik}}(x_{C_{ik}}) = z_{ik}\log\pi_k, \tag{B.22}$$

$$\theta_{D_k}(x_{D_k}) = -\frac{\beta_0}{2}\mu_k^T\Lambda_k\mu_k + \beta_0\mu_k^T\Lambda_k\mu_0. \tag{B.23}$$

The non-singleton factors are $\theta_{A_{ik}}(x_{A_{ik}}), \theta_{B_{ik}}(x_{B_{ik}}), \theta_{C_{ik}}(x_{C_{ik}})$ and $\theta_{D_k}(x_{D_k})$. There are $|\mathcal{F}| = 3NK + K = K(3N+1)$ non-singleton factors.

B.2 LDA posterior density in a factor graph form

Term 1 The log-density function of $p(w|z,\beta)$ is

$$\log p(w|z,\beta) = \sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{v=1}^{V} w_{dn}\log\beta_{z_{dn},v}. \tag{B.24}$$

The term involves latent variables $x_{A_{dnk}} = \{z_{dn}, \beta_{kv}\}$ and the factors are

$$\theta_{A_{dnk}}(x_{A_{dnk}}) := w_{dn}\log\beta_{z_{dn},v}. \tag{B.25}$$

The log-density can be written as

$$\log p(w|z,\beta) = \sum_{f\in\mathcal{F}_1}\theta_f(x_f), \tag{B.26}$$

where $\mathcal{F}_1 = \{A_{dnvk}\}_{d=1,n=1,v=1,k=1}^{M,N,V,K}$.

Term 2 The log-density function of $p(z|\Theta)$ is

$$\log p(z|\Theta) = \sum_{d=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{dn}\log\Theta_{dk}. \tag{B.27}$$

The term involves latent variables $x_{B_{dnk}} = \{\Theta_{dk}, z_{dn}\}$ and the factors are

$$\theta_{B_{dnk}}(x_{B_{dnk}}) := z_{dn}\log\Theta_{dk}. \tag{B.28}$$

The log-density function can then be written as

$$\log p(z|\Theta) = \sum_{f\in\mathcal{F}_2}\theta_f(x_f), \tag{B.29}$$

where $\mathcal{F}_2 = \{B_{dnk}\}_{d=1,n=1,k=1}^{M,N,K}$.

Term 3 The log-density function of $p(\Theta; \alpha_0)$ is

$$\log p(\Theta; \alpha_0) \propto \sum_{d=1}^{M} \sum_{k=1}^{K} (\alpha_{0k} - 1) \log \Theta_{dk}. \tag{B.30}$$

The factors in this log-density only involve singleton latent variables, so they can be written as

$$\theta_{\Theta_{dk}} = (\alpha_{0k} - 1) \log \Theta_{dk} \tag{B.31}$$

and the log-density is

$$\log p(\Theta; \alpha_0) = \sum_{v \in \mathcal{V}_1} \theta_v(v), \tag{B.32}$$

where $\mathcal{V}_1 = \{\Theta_{11}, \ldots, \Theta_{MK}\}$.

Term 4 The log-density function of $p(\beta; \eta_0)$ is

$$\log p(\beta; \eta_0) \propto \sum_{k=1}^{K} \sum_{v=1}^{V} (\eta_{0v} - 1) \log \beta_{kv}. \tag{B.33}$$

The factors in this log-density only involve singleton latent variables, so they can be written as

$$\theta_{\beta_{kv}} = (\eta_{0v} - 1) \log \beta_{kv} \tag{B.34}$$

and the log-density is

$$\log p(\beta; \eta_0) = \sum_{v \in \mathcal{V}_2} \theta_v(v), \tag{B.35}$$

where $\mathcal{V}_2 = \{\beta_{11}, \ldots, \beta_{KV}\}$.

Combining factors Recall $\phi = (\alpha_0, \eta_0)$, $x = (z_{11}, \ldots, z_{MN}, \Theta_1, \ldots, \Theta_M, \beta_1, \ldots, \beta_K)$, and $w = (w_{11}, \ldots, w_{MN})$. Combining the factorized representations of each of the terms in the joint log-density gives

$$p(x|w, \phi) = f(x, \theta) = \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{f \in \mathcal{F}} \theta_f(x_f), \tag{B.36}$$

where $\mathcal{V} = \{\Theta_{11}, \ldots, \Theta_{MK}, \beta_{11}, \ldots, \beta_{KV}\}$, and $\mathcal{F} = \{[A_{dnvk}]_{d=1,n=1,v=1,k=1}^{M,N,V,K}, [B_{dnk}]_{d=1,n=1,k=1}^{M,N,K}\}$. The factors are

$$\begin{aligned}
\theta_{\beta_{kv}} &= (\eta_{0v} - 1) \log \beta_{kv}, && \text{(B.37)} \\
\theta_{\Theta_{dk}}(\Theta_{dk}) &= (\alpha_{0k} - 1) \log \Theta_{dk}, && \text{(B.38)} \\
\theta_{A_{dnvk}}(x_{A_{dnvk}}) &= w_{dn} \log \beta_{z_{dn}, v}, && \text{(B.39)} \\
\theta_{B_{dnk}}(x_{B_{dnk}}) &= z_{dn} \log \Theta_{dk}. && \text{(B.40)}
\end{aligned}$$

The non-singleton factors are $\theta_{A_{dnvk}}(x_{A_{dnvk}})$ and $\theta_{B_{dnk}}(x_{B_{dnk}})$. There are $|\mathcal{F}| = MNVK + MNK = MNK(V + 1)$ non-singleton factors.

APPENDIX C

SUPPLEMENTAL MATERIALS FOR CHAPTER 4

C.1 Proof of Proposition 1

Proof. Given a dataset X, pick an element $x \in X$. We consider all possible Ω clusters X_L^ω in $\mathbb{C}(X)_x$. Given X_L^ω, then X_R^ω is fixed so as to satisfy $X_L^\omega \bigcup X_R^\omega = X$ and $X_L^\omega \bigcap X_R^\omega = \emptyset$. We want to show that the partition function $Z(X)$ can be written recursively in terms of $Z(X_L^\omega)$ and $Z(X_R^\omega)$.

The partition function is defined as the sum of the energies of all possible hierarchical clusterings $\mathcal{H}_X = \{\mathtt{H}^m\}_{m=1}^M$,

$$Z(X) = \sum_{m=1}^{M} \phi(\mathtt{H}^m(X)) = \sum_{m=1}^{M} \psi(X_L^m, X_R^m)\ \phi(\mathtt{H}^m(X_L^m))\ \phi(\mathtt{H}^m(X_R^m)) \tag{C.1}$$

where $X_L^m \bigcup X_R^m = X$, $X_L^m \bigcap X_R^m = \emptyset$. Also, $\mathtt{H}^m(X_L^m)$ and $\mathtt{H}^m(X_R^m)$ are the sub-hierarchies in $\mathtt{H}^m$ that are rooted at X_L^m and X_R^m, respectively. Next, we rewrite Eq. C.1 grouping together all the hierarchies $\mathtt{H}^i$ that have the same clusters $\{X_L^m, X_R^m\}$ [1],

$$Z(X) = \sum_{\omega=1}^{\Omega} \psi(X_L^\omega, X_R^\omega) \sum_{j=1}^{J} \phi(\mathtt{H}^j(X_L^\omega)) \sum_{k=1}^{K} \phi(\mathtt{H}^k(X_R^\omega)) = \sum_{\omega=1}^{\Omega} \psi(X_L^\omega, X_R^\omega)\ Z(X_L^\omega)\ Z(X_R^\omega) \tag{C.2}$$

with $M = \Omega \cdot J \cdot K$, $J = (2|X_L^\omega| - 3)!!$, and $K = (2|X_R^\omega| - 3)!!$ for a full trellis. Thus, $Z(X)$ of a cluster X can be written recursively in terms of the partition function of the sub-clusters of X [2].

□

C.2 Proofs of Theorem 7 and Corollary 3

The partition function is computed for each node in the trellis, and due to the order of computation, at the time of computation for node i, the partition functions for all nodes in the subtrellis rooted at node i have already been computed. Therefore, the

[1] The cluster trellis provides an exact solution conditioned on the fact that the domain of the linkage function is the set of pairs of clusters, and not pairs of trees.

[2] Note that for each singleton x_i, we have $Z(x_i) = 1$.

partition function for a node with i elements can be computed in 2^i steps (given the pre-computed partition functions for each of the node's descendants), since the number of nodes for the trellis rooted at node i (with i elements) corresponds to the powerset of i. There are $\binom{N}{i}$ nodes of size i, making the total computation $\sum_{i=1}^{N} 2^i \binom{N}{i} = 3^N - 1$.

In Corollary 3 we state that Algorithm 2 is super-exponentially more efficient than brute force methods that consider every possible hierarchy. Their ratio is

$$r = \frac{(2N-3)!!}{3^N} = \frac{1}{2\sqrt{\pi}} \left(\frac{2}{3}\right)^N \Gamma(N - 1/2) \tag{C.3}$$

with Γ the gamma function. Thus, r presents a super-exponential growth in terms of N.

C.3 Proof of Proposition 2

Proof. We proceed in a similar way as detailed in Appendix § C.1 , as follows. Given a dataset X, pick an element $x \in X$. We consider all possible Ω clusters X_L^ω in $\mathbb{C}(X)_x$. Given X_L^ω, then X_R^ω is fixed so as to satisfy $X_L^\omega \bigcup X_R^\omega = X$ and $X_L^\omega \bigcap X_R^\omega = \emptyset$. We want to show that the MAP clustering $\phi(\mathtt{H}^*(X))$ can be computed recursively in terms of $\phi(\mathtt{H}^*(X_L^\omega))$ and $\phi(\mathtt{H}^*(X_R^\omega))$.

The MAP value is defined as the energy of the clustering with maximal energy ϕ among all possible hierarchical clusterings $\mathcal{H}_X = \{\mathtt{H}^m\}_{m=1}^M$,

$$\begin{aligned} \phi(\mathtt{H}^*(X)) &= \max_{m \in M} \phi(\mathtt{H}^m(X)) \\ &= \max_{m \in M} \psi(X_L^m, X_R^m)\ \phi(\mathtt{H}^m(X_L^m))\ \phi(\mathtt{H}^m(X_R^m)) \end{aligned} \tag{C.4}$$

where $X_L^m \bigcup X_R^m = X$, $X_L^m \bigcap X_R^m = \emptyset$. Also, $\mathtt{H}^m(X_L^m)$ and $\mathtt{H}^m(X_R^m)$ are the sub-hierarchies in $\mathtt{H}^m$ that are rooted at X_L^m and X_R^m, respectively. As mentioned earlier, the cluster trellis provides an exact MAP solution conditioned on the fact that the domain of the linkage function is the set of pairs of clusters, and not pairs of trees. Thus, we can rewrite Eq. C.4 grouping together all the hierarchies $\mathtt{H}^i$ that have the same clusters $\{X_L^m, X_R^m\}$, as follows

$$\begin{aligned} \phi(\mathtt{H}^*(X)) &= \max_{\omega \in \Omega} \left(\psi(X_L^\omega, X_R^\omega)\ \max_{j \in J} \phi(\mathtt{H}^j(X_L^\omega))\ \max_{k \in K} \phi(\mathtt{H}^k(X_R^\omega)) \right) \\ &= \max_{\omega \in \Omega} \psi(X_L^\omega, X_R^\omega)\ \phi(\mathtt{H}^*(X_L^\omega))\ \phi(\mathtt{H}^*(X_R^\omega)) \end{aligned} \tag{C.5}$$

with $M = \Omega \cdot J \cdot K$. Thus, $\phi(\mathtt{H}^*(X))$ of a cluster X can be written recursively in terms of the MAP values of the sub-clusters of X [3].

□

[3] Note that for each singleton x_i, we have $\phi(\mathtt{H}^*(x_i)) = 1$.

C.4 Proof of MAP Time Complexity

The MAP tree is computed for each node in the trellis, and due to the order of computation, at the time of computation for node i, the MAP trees for all nodes in the subtrellis rooted at node i have already been computed. Therefore, the MAP tree for a node with i elements can be computed in 2^i steps (given the pre-computed partition functions for each of the node's descendants), since the number of nodes for the trellis rooted at node i (with i elements) corresponds to the powerset of i. There are $\binom{n}{i}$ nodes of size i, making the total computation $\sum_{i=1}^{N} 2^i \binom{N}{i} = 3^N - 1$.

BIBLIOGRAPHY

[1] A. Ahmed, M. Aly, J. E. Gonzalez, S. M. Narayanamurthy, and A. Smola. Scalable inference in latent variable models. In *Web Search and Data Mining*, 2012.

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003. ISSN 1573-0565. doi: 10.1023/A:1020281327116. URL `https://doi.org/10.1023/A:1020281327116`.

[3] S. Balakrishnan, M. J. Wainwright, and B. Yu. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *Ann. Statist.*, 45(1): 77–120, 02 2017. doi: 10.1214/16-AOS1435. URL `https://doi.org/10.1214/16-AOS1435`.

[4] H. Bandi, D. Bertsimas, and R. Mazumder. Learning a mixture of gaussians via mixed-integer optimization. *Informs Journal on Optimization*, 1(3):221–240, 2019.

[5] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 2004.

[6] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[7] M. S. Bazaraa and H. D. Sherali. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. March 1980. doi: 10.1002/nav.3800270104.

[8] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4, 1962.

[9] D. Bertsekas, G. Lauer, N. Sandell, and T. Posbergh. Optimal short-term scheduling of large-scale power systems. *IEEE Transactions on Automatic Control*, 28(1):1–11, 1983. doi: 10.1109/TAC.1983.1103136.

[10] D. Bertsimas and A. King. OR forum–An algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2016. doi: 10.1287/opre.2015.1436. URL `https://doi.org/10.1287/opre.2015.1436`.

[11] D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016. doi: 10.1214/15-aos1388. URL https://doi.org/10.1214/15-aos1388.

[12] J. Bisschop and A. Meeraus. On the Development of a General Algebraic Modeling System in a Strategic Planning Environment. In *Applications*, volume 20 of *Mathematical Programming Studies*, pages 1–29. Springer Berlin Heidelberg, 1982.

[13] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518): 859–877, 2017. doi: 10.1080/01621459.2017.1285773.

[14] C. Blundell, Y. Teh, and K. Heller. Bayesian rose trees. *Uncertainty in Artificial Intelligence, (UAI)*, 2010.

[15] L. Boyles and M. Welling. The time-marginalized coalescent prior for hierarchical clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[16] M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. *Journal of the American Statistical Association*, 105(489):324–335, 2010. doi: 10.1198/jasa.2009.tm08030. URL https://doi.org/10.1198/jasa.2009.tm08030.

[17] M. Cacciari, G. P. Salam, and G. Soyez. The anti-k_t jet clustering algorithm. *JHEP*, 2008.

[18] D. Callan. A combinatorial survey of identities for the double factorial. *arXiv*, 2009.

[19] I. Chami, A. Gu, V. Chatziafratis, and C. Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] P. Cimiano and S. Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005.

[21] J. J. Cochran. *INFORMS Analytics Body of Knowledge*. John Wiley & Sons, 2018.

[22] E. Dale and J. Moon. The permuted analogues of three Catalan sets. *Journal of Statistical Planning and Inference*, 1993.

[23] S. Dasgupta. A cost function for similarity-based hierarchical clustering. *Symposium on Theory of Computing (STOC)*, 2016.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL http://www.jstor.org/stable/2984875.

[25] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[26] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

[27] P. Flaherty, P. Wiratchotisatian, J. A. Lee, and A. C. Trapp. Maximum a-posteriori estimation for the gaussian mixture model via mixed integer nonlinear programming. *arXiv preprint arXiv:1911.04285*, 2019.

[28] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications (Topics in Chemical Engineering)*. Oxford University Press, 1995.

[29] C. A. Floudas and A. R. Ciric. Strategies for overcoming uncertainties in heat exchanger network synthesis. *Computers & Chemical Engineering*, 13:1133–1152, 1989.

[30] GAMS Development Corporation. *GAMS — A User's Guide*. GAMS Development Corporation, 2017.

[31] A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410): 398–409, 1990. ISSN 01621459. URL http://www.jstor.org/stable/2289776.

[32] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. doi: 10.1109/TPAMI.1984.4767596.

[33] A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10, 1972.

[34] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[35] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/8d6dc35e506fc23349dd10ee68dabb64-Paper.pdf.

[36] C. Greenberg, N. Monath, A. Kobren, P. Flaherty, A. McGregor, and A. McCallum. Compact representation of uncertainty in clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[37] T. Griffiths and M. Steyvers. Prediction and semantic association. *Advances in Neural Information Processing Systems (NeurIPS)*, 2002.

[38] M. Grötschel. *Optimization stories*. Dt. Mathematiker-Vereinigung, 2012.

[39] M. Guignard. Lagrangean relaxation. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 11(2):151–200, 2003.

[40] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL `http://www.jstor.org/stable/2334940`.

[41] K. Heller and Z. Ghahramani. Randomized algorithms for fast bayesian hierarchical clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005.

[42] K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. *Proceedings of the 22nd international conference on Machine learning*, 2005.

[43] S. Hettich and S. D. Bay. UCI kdd archive, 1999. URL `http://kdd.ics.uci.edu`.

[44] Y. Hu, J. L. Ying, H. Daume III, and Z. I. Ying. Binary to bushy: Bayesian hierarchical clustering with the beta coalescent. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[45] C. Jin, Y. Zhang, S. Balakrishnan, M. J. Wainwright, and M. I. Jordan. Local maxima in the likelihood of Gaussian mixture models: Structural results and algorithmic consequences. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4123–4131, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL `http://dl.acm.org/citation.cfm?id=3157382.3157558`.

[46] J. Johnson, D. Malioutov, and A. Willsky. Lagrangian relaxation for map estimation in graphical models. *45th Annual Allerton Conference on Communication, Control, and Computing 2007*, 1, 10 2007.

[47] J. K. Johnson. Convex relaxation methods for graphical models: Lagrangian and maximum entropy approaches. EECS, MIT, 2008.

[48] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(37):183–233, 1999.

[49] K. P. Kalinin and N. G. Berloff. Computational complexity continuum within ising formulation of np problems. *Communications Physics*, 5, 2022.

[50] J. H. Kappes, P. Swoboda, B. Savchynskyy, T. Hazan, and C. Schnörr. Probabilistic correlation clustering and image partitioning using perturbed multicuts. *International Conference on Scale Space and Variational Methods in Computer Vision*, 2015.

[51] D. A. Knowles and Z. Ghahramani. Pitman-yor diffusion trees. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[52] J. Kohonen and J. Corander. Computing exact clustering posteriors with subset convolution. *Communications in Statistics-Theory and Methods*, 2016.

[53] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[54] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011. doi: 10.1109/TPAMI.2010.108.

[55] A. Kraskov, H. Stögbauer, R. G. Andrzejak, and P. Grassberger. Hierarchical clustering using mutual information. *EPL (Europhysics Letters)*, 2005.

[56] H.-C. Kuo and C.-H. Lin. A directed genetic algorithm for global optimization. *Applied Mathematics and Computation*, 219(14):7348–7364, 2013. doi: 10.1016/j.amc.2012.12.046. URL `https://doi.org/10.1016/j.amc.2012.12.046`.

[57] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer New York, NY, 1997.

[58] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497, 1960. doi: 10.2307/1910129. URL `https://doi.org/10.2307/1910129`.

[59] C. Lemarechal. *In Computational Combinatorial Optimization*. 2001.

[60] S. Macaluso, C. Greenberg, N. Monath, J. A. Lee, P. Flaherty, K. Cranmer, A. McGregor, and A. McCallum. Cluster trellis: Data structures amp; algorithms for exact inference in hierarchical clustering. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2467–2475. PMLR, 13–15 Apr 2021. URL `https://proceedings.mlr.press/v130/macaluso21a.html`.

[61] H. Mawengkang and B. Murtagh. Solving nonlinear integer programs with large-scale optimization software. *Annals of Operations Research*, 5:425–437, 06 1986. doi: 10.1007/BF02022084.

[62] M. Meila. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007. ISSN 0047-259X. doi: https://doi.org/10.1016/j.jmva.2006.11.013. URL https://www.sciencedirect.com/science/article/pii/S0047259X06002016.

[63] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, (21):1087–1092, 1953. URL https://doi.org/10.1063/1.1699114.

[64] N. Monath, M. Zaheer, D. Silva, A. McCallum, and A. Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. *Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[65] B. Moseley and J. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[66] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[67] R. M. Neal. Density modeling and clustering using dirichlet diffusion trees. *Bayesian statistics*, 2003.

[68] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988.

[69] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, New York, 2006. ISBN 9780387303031.

[70] A. Oniśko, M. J. Druzdzel, and H. Wasyluk. Learning bayesian network parameters from small data sets: application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001. ISSN 0888-613X. doi: https://doi.org/10.1016/S0888-613X(01)00039-1. URL https://www.sciencedirect.com/science/article/pii/S0888613X01000391.

[71] S. Papageorgaki and G. V. Reklaitis. Optimal design of multipurpose batch plants 1 & 2. *Industrial & Engineering Chemistry Research*, 29:2054, 1990.

[72] G. Papandreou and A. L. Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. *2011 International Conference on Computer Vision*, 2011.

[73] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. 1988.

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[75] G. Rebane and J. Pearl. The recovery of causal poly-trees from statistical data. In *3rd Workshop on Uncertainty in AI*, pages 222–228, 2008.

[76] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

[77] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer New York, NY, 2004. doi: 10.1007/978-1-4757-4145-2. URL https://doi.org/10.1007/978-1-4757-4145-2.

[78] P. Rossi. *bayesm: Bayesian Inference for Marketing/Micro-Econometrics*, 2022. URL https://cran.r-project.org/web/packages/bayesm. R package version 3.1-5.

[79] N. Sahinidis, I. Grossmann, R. Fornari, and M. Chathrathi. Optimization model for long range planning in the chemical industry. *Computers Chemical Engineering*, 13(9):1049–1063, 1989. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(89)87046-2. URL https://www.sciencedirect.com/science/article/pii/0098135489870462.

[80] N. V. Sahinidis. *BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017.

[81] N. V. Sahinidis, I. E. Grossmann, and C. M. U. D. R. Center. Reformation of the multiperiod MILP model for capacity expansion of chemical process. 1 1989. doi: 10.1184/R1/6467378.v1. URL https://kilthub.cmu.edu/articles/journal_contribution/Reformation_of_the_multiperiod_MILP_model_for_capacity_expansion_of_chemical_process/6467378.

[82] M. I. Shlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976.

[83] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message-passing. In *24th Conference in Uncertainty in Artificial Intelligence*, pages 503–510. AUAI Press, 2008.

[84] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 219–254. MIT Press, 2012.

[85] T. Sørlie, C. M. Perou, R. Tibshirani, T. Aas, S. Geisler, H. Johnsen, T. Hastie, M. B. Eisen, M. Van De Rijn, S. S. Jeffrey, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 2001.

[86] M. A. Suchard, P. Lemey, G. Baele, D. L. Ayres, A. J. Drummond, and A. Rambaut. Bayesian phylogenetic and phylodynamic data integration using beast 1.10. *Virus evolution*, 2018.

[87] R. Suzuki and H. Shimodaira. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, 2006.

[88] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005. doi: 10.1007/s10107-005-0581-8. URL `https://doi.org/10.1007/s10107-005-0581-8`.

[89] Y. W. Teh, H. Daume III, and D. M. Roy. Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems (NeurIPS)*, 2008.

[90] University of North Carolina. UNC microarray database. *https://genome.unc.edu/*, 2020.

[91] L. Uusitalo. Advantages and challenges of bayesian networks in environmental modelling. *Ecological Modelling*, 203(3):312–318, 2007. ISSN 0304-3800. doi: https://doi.org/10.1016/j.ecolmodel.2006.11.033. URL `https://www.sciencedirect.com/science/article/pii/S0304380006006089`.

[92] J. Vaselenak, I. E. Grossmann, and A. W. Westerberg. Optimal retrofit design of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 26: 718, 1987.

[93] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. ISSN 1935-8237. doi: 10.1561/2200000001. URL `http://dx.doi.org/10.1561/2200000001`.

[94] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51:3697–3717, 2005.

[95] D. J. Wales and J. P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997. doi: 10.1021/jp970984n. URL `https://doi.org/10.1021/jp970984n`.

[96] C. Wang and D. M. Blei. Variational inference in nonconjugate models. *J. Mach. Learn. Res.*, 14(1):1005–1031, Apr. 2013. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2502581.2502613`.

[97] L. Wang, A. Bouchard-Côté, and A. Doucet. Bayesian phylogenetic inference using a combinatorial sequential monte carlo method. *Journal of the American Statistical Association*, 2015.

[98] Y. Weiss, C. Yanover, and T. Meltzer. Map estimation, linear programming and belief propagation with convex free energies. *Uncertainty in Artificial Intelligence, (UAI)*, 2007.

[99] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, 2011.

[100] M. Wick, S. Singh, and A. McCallum. A discriminative hierarchical model for fast coreference at large scale. *Association for Computational Linguistics (ACL)*, 2012.

[101] Y. Xu, K. Heller, and Z. Ghahramani. Tree-based inference for dirichlet process mixtures. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[102] T. Yee and I. Grossmann. Simultaneous optimization models for heat integration—ii. heat exchanger network synthesis. *Computers Chemical Engineering*, 14(10):1165–1184, 1990. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(90)85010-8. URL `https://www.sciencedirect.com/science/article/pii/0098135490850108`.