ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

SCIENZE STATISTICHE

Ciclo 38

Settore Concorsuale: 13/STAT-01

Settore Scientifico Disciplinare: STAT-01/A

Advancements in structure learning of undirected graphical models: from novel specifications to Bayesian and variational inference

Presentata da: Pham Huu Thuan

Coordinatore Dottorato
Angela Montanari

Supervisore
Monica Chiogna

Co-supervisore
Davide Risso

Esame finale anno 2026

Dedicated to my family

Acknowledgements

I would like to extend my heartfelt gratitude to my supervisors, Professor Monica Chiogna, for her enthusiastic support throughout my PhD journey. Her unwavering support has been invaluable during my research and thesis writing process.

I would like to express my sincere gratitude to Professor Pier Giovanni Bissiri for his valuable support, insightful ideas, and constructive comments on Chapter 2 of this dissertation. My thanks also go to Dr. Nguyen Thi Kim Hue for providing the code for the LPGM algorithm and the real dataset used in the analysis. I am especially grateful to Dr. Luca Maestrini for his helpful suggestions and support, which enabled me to carry out Chapter 3 of this dissertation. I would also like to express my gratitude to Dr. Erika Banzato for her valuable suggestions, which enabled me to carry out Chapter 4 of this dissertation.

I am grateful to the University of Bologna for their financial support over the past three years. I would also like to thank my colleagues from the 38th PhD cycle in Statistical Sciences; they have been great friends and companions during my time in Italy.

Last but not least, I would like to thank my family for their support, encouragement, and unconditional love. They have been a great motivation for me to achieve my dream.

Abstract

This thesis introduces some advances in the field of structure learning of undirected graphical modeling, with a focus on Bayesian inference and variational approximations. First, in the context of count data, a novel model specification based on the Poisson distribution is proposed that demonstrates that, under certain conditions, the induced joint distribution satisfies the properties of a Markov graph. Second, two Bayesian structure learning algorithms are developed: Chain-LPGM, which leverages the proposed model specification, and Bayes-LPGM, which builds upon local Poisson graphical models as introduced by Allen and Liu (2013). Third, posterior inference is addressed within the framework of a general Bayes approach (Bissiri et al., 2016), employing Markov chain Monte Carlo sampling and mean field variational approximation methods. Lastly, a novel approach for structure learning of differential graphical models for exponential family distributions is proposed. This approach utilizes density ratio estimation results and integrates concepts from probabilistic classifiers to allow easy inference.

Collectively, these contributions advance theoretical foundations and practical methodologies for graphical modeling and Bayesian inference in complex data settings.

Contents

List of Figures

List of Tables

Chapter 0

Introduction

0.1 Overview

Graphical models provide a powerful framework for representing and analyzing complex dependency structures among random variables. In these models, nodes represent variables and edges encode conditional dependencies, making them particularly suitable for the analysis of high-dimensional data in fields such as biology, genetics, neuroscience, and social sciences. A substantial body of research has focused on Gaussian graphical models, where the inverse covariance matrix encodes the underlying graph structure. Beyond the Gaussian setting, Poisson graphical models have been proposed for analyzing count data, which are particularly relevant in genomics and other biological applications. More recently, differential graphical models have gained attention as they allow for the comparison of network structures across different experimental conditions.

A central task in graphical modeling is structure learning, namely the estimation of the underlying graph from observed data. Structure learning can be approached from several perspectives, including constraint-based methods, score-based approaches, and regularization techniques (Drton and Maathuis, 2017). In the Bayesian paradigm, prior distributions are imposed on graph structures and parameters, enabling coherent uncertainty quantification and

principled incorporation of prior knowledge (Vogels et al., 2024). However, Bayesian inference for structure learning is often computationally challenging due to the combinatorial nature of graph spaces. Recent developments, including variational inference and efficient sampling algorithms, have opened new possibilities for scalable Bayesian structure learning in both Gaussian and non-Gaussian graphical models. These advances are particularly relevant for modern applications, where the data are often high-dimensional, heterogeneous, and structured in complex ways.

The remainder of this dissertation is organized as follows.

- Chapter 1 introduces the fundamental concepts of graphical models and structure learning, providing the necessary background for the subsequent chapters.

- Chapter 2 proposes a novel model specification for undirected graphical models of count data based on the Poisson distribution and develops a general Bayesian framework for structure learning.

- Chapter 3 reviews the mean field variational Bayes methodology and applies it to the structure learning problem formulated in Chapter 2, with the goal of reducing computational costs while maintaining high predictive accuracy.

- Chapter 4 focuses on structure learning of differential graphical models and presents a new Bayesian approach to structure learning based on variable selection in density ratio estimation.

0.2 Main contribution of the thesis

Main contributions of the thesis can be summarized as follows.

1. A novel model specification, is proposed for undirected graphical modeling of count data based on the Poisson distribution. It is established

that, under certain conditions, the joint distribution induced by this specification constitutes a Markov graph.

2. Two Bayesian structure learning algorithms are proposed for making inference on the unknown structure, one, named Chain-LPGM, based on our model specification, and one, named Bayes-LPGM, exploiting the local Poisson graphical models introduced by Allen and Liu (2013).

3. Posterior inference is proposed within a general Bayes approach (Bissiri et al., 2016), using Markov chain Monte Carlo sampling and mean field variational approximation.

4. In the context of structure learning of differential graphical models, a novel approach is proposed for exponential family models that exploits results on density ratios of distributions and ideas from probabilistic classifiers.

Chapter 1

Background

In this chapter, we introduce the fundamental concepts of conditional independence and graphical modeling, concentrating on undirected graphical models. Furthermore, we provide an overview of the primary methods for structure learning in graphical models, ending with a succint explanation of the general Bayes approach.

1.1 Conditional independence

In this section, we define conditional independence for random variables. We begin with the concept of conditional independence of events.

Definition 1.1.1. We say that events α and β are conditionally independent given event γ in P, denoted by $\alpha \perp_P \beta \mid \gamma$ or $\alpha \perp \beta \mid \gamma$, if $P(\alpha \mid \beta \cap \gamma) = P(\alpha \mid \gamma)$ or $P(\beta \cap \gamma) = 0$.

From the property of unconditional independencies, we can provide an alternative definition.

Proposition 1.1.2. *$\alpha \perp_P \beta \mid \gamma$ if and only if $P(\alpha \cap \beta \mid \gamma) = P(\alpha \mid \gamma)P(\beta \mid \gamma)$.*

Now, we can define the conditional independence of random variables.

Definition 1.1.3. We say that random variables X and Y are conditionally independent given the random variable Z in a distribution P and write $X \perp Y \mid Z$ if for all $A \in \sigma(X),\ B \in \sigma(Y),\ C \in \sigma(Z)$ we have $X \in A \perp Y \in B \mid Z \in C$, where $\sigma(X), \sigma(Y), \sigma(Z)$ are the $\sigma-$algebras generated by the random variables X, Y and Z.

Before presenting some properties of conditional independence, we introduce an important subclass of distributions.

Definition 1.1.4. A distribution P is positive if for all events $\alpha \neq \emptyset$ in sample space, we have $P(\alpha) > 0$.

Proposition 1.1.5. *For the random variables X, Y, Z and W it holds:*

(C1) *if $X \perp Y \mid Z$ then $Y \perp X \mid Z$;*

(C2) *if $X \perp Y \mid Z$, then $X \perp g(Y) \mid Z$;*

(C3) *if $X \perp Y \mid Z$, then $X \perp Y \mid (Z, g(Y))$;*

(C4) *if $X \perp Y \mid Z$ and $X \perp W \mid (Y, Z)$, then $X \perp (Y, W) \mid Z$.*

If the joint distribution of the random variables has density with respect to (w.r.t) the product measure which is positive, we also have:

(C5) *if $X \perp Y \mid (Z, W)$ and $X \perp W \mid (Y, Z)$, then $X \perp (Y, W) \mid Z$.*

Conditional independence fundamentally encodes the concept of irrelevance. If we interpret $A \perp B \mid C$ as "Knowing C, A is irrelevant for learning B", the properties (C1)-(C4) translate to:

(I1) if, knowing C, learning A is irrelevant for learning B, then B is irrelevant for learning A;

(I2) if, knowing C, learning A is irrelevant for learning B, then A is irrelevant for learning any part D of B;

(I3) if, knowing C, learning A is irrelevant for learning B, it remains irrelevant even after learning any part D of B;

(I4) if, knowing C, learning A is irrelevant for learning B, and even after learning A, learning D; remains irrelevant for B, then both A and D are irrelevant for learning B.

The property (C5) is somewhat more subtle and not always immediately apparent. Additionally, the symmetry property (C1) is specific to probabilistic conditional independence, and does not generally apply to the broader concept of irrelevance.

The general interpretation of conditional independence highlights the value of an abstract study of algebraic structures that meet these properties. An independence model $\perp_\sigma$ is defined as a ternary relation over the subsets of a finite set V.

Definition 1.1.6. An independence model is *graphoid* if for all disjoint subsets A, B, C, D of V it satisfies following properties:

(S1) Symmetry: if $A \perp_\sigma B \mid C$, then $B \perp_\sigma A \mid C$;

(S2) Decomposition: if $A \perp_\sigma (B \cup D) \mid C$, then $A \perp_\sigma B \mid C$ and $A \perp_\sigma D \mid C$;

(S3) Weak Union: if $A \perp_\sigma (B \cup D) \mid C$, then $A \perp_\sigma B \mid (C \cup D)$;

(S4) Contraction: if $A \perp_\sigma B \mid C$ and $A \perp_\sigma D \mid (B \cup C)$, then $A \perp_\sigma (B \cup D) \mid C$;

(S5) Intersection: if $A \perp_\sigma B \mid (C \cup D)$ and $A \perp_\sigma C \mid (B \cup D)$, then $A \perp_\sigma (B \cup C) \mid D$.

The relation is *semi-graphoid* if (S1)-(S4) hold. It is *compositional graphoid* if (S1)-(S5) hold and:

(S6) Composition: If $A \perp_\sigma B \mid C$ and $A \perp_\sigma D \mid C$, then $A \perp_\sigma (B \cup D) \mid C$.

For a system V of labeled random variables $X_v, v \in V$, we use the shorthand

$$A \perp B \mid C \iff \mathbf{X}_A \perp \mathbf{X}_B \mid \mathbf{X}_C,$$

where $\mathbf{X}_A = (X_v, v \in A)$ denotes the variables with labels in A. The properties (C1)-(C4) imply that $\perp$ satisfies the semi-graphoid axioms for this system, and the graphoid axioms if the joint density of the variables is strictly positive.

1.2 Undirected Graphical Models

A graph is a pair $G = (V, E)$, where $V = \{1, \ldots, p\}$ represents the set of nodes with each node i corresponding to a random variable $X_i \in \mathbf{X}$. The set of edges is given by $E \subset V \times V$. A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. In probabilistic graphical network, $\partial i = \{j \in V : (i, j) \in E\}$ is the Markov blanket of node i (Koller and Friedman, 2009). For a vertex $i \in V$, the *neighbourhood* of i, denoted by $N(i)$, is defined as the set of all vertices that are adjacent to i, each vertex $j \in N(i)$ is called a *neighbour* of i. An *ordering* of V is a bijective function $\sigma : V \to \{1, 2, \ldots, p\}$, which assigns to each vertex $i \in V$ a unique position $\sigma(i)$ in the sequence.

1.2.1 Separation

Definition 1.2.1. (Separation) Let $G = (V, E)$ be a finite and simple undirected graph (no self-loops, no multiple edges). For subsets A, B, and S of V, we say that S separates A and B in G and write $A \perp_G B \mid S$ if every path between a node in A and a node in B contains a node in S.

It is straightforward to observe that the separation relation on subsets of V forms a compositional graphoid. The term "graphoid" for such separation relations derives from this property. This fact is why the name "graphoid" was chosen for such an independence model.

1.2.2 Decomposition

Here, we study decompositions and decomposable graphs.

Definition 1.2.2. A triple (A, B, C) of disjoint subsets of the vertex set V of an undirected graph G is said to form a decomposition of G if $V = A \cup B \cup C$, $A \perp_G B \mid C$, and C is a complete subset of V .

In this case we say that (A, B, C) decomposes G into the components $G_{A\cup C}$ and $G_{B\cup C}$. Note that we allow any of the sets in (A, B, C) to be empty. If the sets A and B in (A, B, C) are both non-empty, we say that the decomposition is proper.

Definition 1.2.3. An undirected graph G is said to be decomposable if it is complete, or if there exists a proper decomposition (A, B, C) into decomposable subgraphs $G_{A\cup C}$ and $G_{B\cup C}$.

1.2.3 Perfect sequence and perfect numbering

Here, we introduce the concepts of a perfect sequence of sets and a perfect numbering of the vertices, which are important notions that will be used in the subsequent parts of this dissertation.

Let now $B_1, \ldots, B_k$ be a sequence of subsets of the vertex set V of an undirected graph G. Let

$$H_j = B_1 \cup \ldots \cup B_j,\ R_j = B_j \setminus H_{j-1},\ S_j = H_{j-1} \cap B_j,$$

for $j = 1, \ldots, k$ and $H_0 = \emptyset$. The sequence is said to be perfect if the following conditions are fulfilled:

i) for all $i > 1$ there is a $j < i$ such that $S_i \subseteq B_j$;

ii) the set S_i are complete for all i.

The condition i) is known as the *running intersection property.*

Definition 1.2.4. A perfect numbering of the vertices V of G is a numbering $v_1, \ldots, v_k$ such that

$$B_j = (\{v_j\} \cup N(v_j)) \cap \{v_1, \ldots, v_j\}, \quad j \geq 1,$$

is a perfect sequence of sets.

1.2.4 Markov property

Recall that $\mathbf{X}_A = (X_i : i \in A \subseteq V)$ is the random vector of all variables in $A \subseteq V$. Variables in the random vector $\mathbf{X}$ satisfy the following properties

(P) Pairwise Markov property w.r.t G if:

$$(i,j) \notin E \implies X_i \perp X_j \mid X_{V\setminus\{i,j\}}.$$

(L) Local Markov property w.r.t G if:

$$X_i \perp X_{V\setminus\{\partial i \cup \{j\}\}} \mid X_{\partial i},$$

for every node $i \in V$.

(G) Global Markov property w.r.t G if:

$$\mathbf{X}_A \perp \mathbf{X}_B \mid \mathbf{X}_C,$$

for any triple of pairwise disjoint subsets $A, B, C \subset V$ such that C separates A and B.

It is easy to see that for any semi-graphoid (G) implies (L) implies (P). Although not true in general, the three Markov properties are equivalent when $\mathbf{X}$ satisfies the intersection axiom for conditional independence (Pearl and Paz, 1986).

The pairwise Markov property in undirected graphical models converts each missing edge into a full conditional independence. Consequently, conducting $\binom{|V|}{2}$ pairwise full conditional independence tests offers a method to estimate the graph. According to the local Markov property, each variable X_i can be optimally predicted from its Markov blanket $X_{\partial i}$. This concept is utilized in a method called neighborhood selection. The more extensive global Markov property is highly valuable for understanding conditional independences.

1.2.5 Factorization

The famous theorem of Hammersley and Clifford shows how to construct distributions that satisfy the Markov property.

Definition 1.2.5. (Factorization) Assume that the random vector $\mathbf{X} = (X_v : v \in V)$ has a density with respect to a product measure $\mu = \bigotimes_{v \in V} \mu_v$ defined on $\mathbb{R}^V$. Typically, each μ_v is either the Lebesgue measure or a counting measure, meaning that each variable X_v is either continuous or discrete. Let $\mathcal{C}(G)$ denote the collection of all cliques (i.e., fully connected subsets) in the graph G, so that $C \in \mathcal{C}(G)$ if $\{v, w\} \in E$ for every $v, w \in C$. The distribution of $\mathbf{X}$ is said to factor according to G if its density can be expressed as

$$\mathbb{P}(\mathbf{x}) = \prod_{C \in \mathcal{C}(G)} \phi_C(\mathbf{x}_C), \quad \mathbf{x} \in \mathbb{R}^V,$$

where each function ϕ_C is defined on $\mathbb{R}^C$, and $\mathbf{x}_C$ represents the subvector of $\mathbf{x}$ indexed by the clique C. (For a finite index set A, $\mathbb{R}^A$ refers to the space of real vectors of length $|A|$, with components labeled by elements of A.) These functions ϕ_C are not required to have a probabilistic interpretation, such as conditional densities.

Theorem 1.2.6. *(Hammersley-Clifford (Lauritzen, 1996)) Suppose* $\mathbf{X} = (X_v : v \in V)$ *has a positive density with respect to a product measure. Then the distribution of* $\mathbf{X}$ *factorizes with respect to* $G = (V, E)$ *if and only if* $\mathbf{X}$ *satisfies the pairwise Markov property with respect to* G.

1.3 Structure Learning

The process of fitting graphical models, sometimes called learning, usually consists of two steps. The first step, known as *structure learning*, involves identifying the graph structure that reflects the conditional independencies present in the data (Koller and Friedman, 2009). The second step is called *parameter learning*, involves estimating the parameters of the global distribution. Since the network structure has already been determined in the

previous step, this task can be simplified to estimating the parameters of the local distributions. In this context, we focus on the first step: structure learning.

Existing structure learning methods can be broadly categorized into three main approaches: constraint-based algorithm, score-based algorithm, and regression-based algorithm. Some algorithms approach structure learning in a hybrid manner, combining the strengths of previous classes.

1.3.1 Constraint-based Algorithms

Constraint-based approaches use conditional independence tests to initially determine a set of conditional independence relationships, and subsequently work to identify the network structure that most accurately aligns with these constraints. The two most widely used constraint-based algorithms are the SGS algorithm (Spirtes et al., 1990) and the PC algorithm (Spirtes and Glymour, 1991). Both aim to separate all variable pairs using all possible conditional sets, with sizes below a specified threshold.

1.3.2 Scoring-based Algorithms

Score-based approaches first establish a criterion for evaluating a network structure based on a given dataset. They then search through the space of all possible structures to identify the graph with the highest score. Score-based approaches are typically grounded in statistical principles, such as the Minimum Description Length (MDL), the Bayesian Information Criterion (BIC) or the Bayesian score. The Bayesian scoring approach was initially developed by Cooper and Herskovits (1992) and later refined by the Bayesian Dirichlet score (Heckerman et al., 1995), which is now one of the most widely used standards. The main problem with score based approaches is that their associated optimization problems are intractable. Studies have demonstrated that optimizing the structure of the networks is NP-hard for large samples, regardless of the scoring criterion used, such as MDL, BIC, and Bayesian

scores (Chickering et al., 2004).

In score-based approaches, the space of candidate structures is typically limited to directed models (Bayesian networks) because calculating common score metrics involves computing the normalization constant of the graphical model distribution, which is intractable for general undirected models. Consequently, estimation of graph structures in undirected models has largely been confined to simpler graph classes such as trees, polytrees (Chow and Liu, 1968), and hypertrees (Srebro, 2003).

1.3.3 Regression-based Algorithms

In recent years, learning a graphical structure through regression has gained popularity. The proposed algorithms mainly differ in their selection of objective loss functions. Algorithms in this category are fundamentally optimization problems that ensure a global optimum for the objective function and provide enhanced scalability.

Likelihood Objective

In this context, the objective loss function is typically the negative likelihood or log-likelihood. In Lee et al. (2006), the authors introduced a L_1-regularized structure learning algorithm for Markov Random Fields, particularly within the context of log-linear models.

Dependency Objective

The algorithms belonging to this class are employed for structure learning in Gaussian graphical models, utilizing linear regression to estimate the Markov blanket of each node. In this framework, each node is considered to depend on the nodes corresponding to nonzero regression coefficients. In Meinshausen and Bühlmann (2006), the authors employed linear regression with L_1 regularization to identify the neighbors of each node in a Gaussian graphical model. In Fan (2007), the authors suggested learning Gaussian graphical

models from directed graphical models by employing modified Lasso regression.

System-identification Objective

Algorithms in this category leverage ideas from network identification via multiple regression, a method originating from the engineering discipline of system identification, in which a model of the connections and functional relations between elements in a network is inferred from measurements of system dynamics (Gardner et al., 2003). The entire system is represented by a differential equation, which is then fitted using regression algorithms. This method has been employed to identify gene regulatory networks.

Minimum Description Length Objective

Methods in this category incorporate the parameters into the MDL criterion and seek to minimize the MDL with regularization or constraints. In Schmidt et al. (2007), the authors introduced a structure learning method that utilizes L_1-penalized regression with MDL as the loss function to determine the parents or neighbors of each node, followed by a score-based search.

1.3.4 Hybrid algorithms

Algorithms in this category combine the strengths of score-based, constraint-based and regression-based algorithms. In Tsamardinos et al. (2006), the authors proposed a Max-min Hill-climbing algorithm for structure learning of Bayesian networks. In Schmidt et al. (2007), the authors proposed a structure learning approach which uses the L_1 penalized regression to find the parents or neighbors for each node, and then apply the score-based search.

1.4 Bayesian approaches for structure learning

While frequentist methods aim to find and estimate the true graph, Bayesian methods attempt to discover the posterior probability distribution of the true graph given the data. It is given by Bayes' rule:

$$\mathbb{P}(G \mid \mathbf{X}) = \frac{\mathbb{P}(\mathbf{X} \mid G)\mathbb{P}(G)}{\mathbb{P}(\mathbf{X})},$$

A significant advantage of this approach is its ability to handle model uncertainty. Instead of choosing the best graph, Bayesian inference can average across many plausible graphs, yielding more robust conclusions.

1.4.1 Prior over Graph and Parameters

Prior over Graph

In the Bayesian paradigm of structure learning, the graph structure G itself is treated as a random variable, and a prior probability distribution, $p(G)$, is assigned to it. This prior serves as a formal mechanism to encode any initial beliefs, domain-specific knowledge, or assumptions about the underlying network topology before any observational data are incorporated (Kim et al., 2019). The specification of graph priors is not merely a formal requirement but serves several essential functions that directly affect both the outcome and the interpretability of structure learning. Here, we list some of these functions.

- **Encouraging sparsity:** in many high-dimensional applications, such as gene regulatory or protein interaction networks, the underlying graph is typically sparse. Carefully designed priors that encourage sparsity lead to parsimonious parameterizations of the underlying dependency structure, which are fundamental for interpretability, computational efficiency, and robustness against overfitting. In this sense, graph priors

act as regularizers that directly address challenges of identifiability and interpretability in complex networks (Talluri et al., 2014).

- **Incorporating prior knowledge:** Bayesian methods naturally allow the integration of external or domain-specific information. Informative graph priors can incorporate expert knowledge, results from previous studies, or theoretical considerations (e.g., known pathways or hub nodes), thereby improving both the accuracy and the scientific plausibility of the inferred structure (Yang et al., 2022).

- **Accounting for model uncertainty:** a key strength of the Bayesian framework is its ability to quantify uncertainty in the learned graph structure. This is particularly valuable in small-sample settings, where multiple competing graphs may explain the data equally well. Priors provide a principled way to average over plausible structures rather than committing to a single estimate.

A range of priors on graph edges has been proposed in the literature, reflecting different assumptions about sparsity, complexity, and connectivity.

- **Uniform prior:** the simplest choice assigns equal probability to each graph in the model space, $\pi(G) \propto 1$. While conceptually straightforward, this prior does not scale well with dimensionality, as it implicitly favors dense graphs due to the combinatorial growth in the number of possible edges.

- **Erdős–Rényi prior:** each edge is included independently with probability $\pi \in (0, 1)$, resulting in a binomial prior on the number of edges. By setting π small, sparsity is encouraged. A common extension introduces a Beta hyperprior on π, thereby adapting the degree of sparsity to the observed data (Erdös and Rényi, 1959).

- **Degree-based priors:** in many applications, such as biological or social networks, the degree distribution is informative. Degree-based pri-

ors assign probabilities based on node degrees, either penalizing high-degree nodes or encouraging hub structures. For instance, scale-free priors capture the heavy-tailed degree distributions observed in real-world networks (Mukherjee and Speed, 2008).

The choice of graph prior has significant implications: sparsity-inducing priors are often preferred in high-dimensional applications for interpretability and computational tractability, whereas informative priors are indispensable when substantial domain knowledge is available.

Priors over Parameters

Given a graph $G = (V, E)$, the parameters of interest are the elements of the matrix $\boldsymbol{\theta} = (\theta_{ij})_{p\times p}$, which encodes conditional independence relationships among the variables. The graph structure imposes constraints on $\boldsymbol{\theta}$, namely $\theta_{ij} = 0$ whenever $(i, j) \notin E$. In the Bayesian framework, prior distributions over $\boldsymbol{\theta}$ are required to respect these structural constraints. The specification of such priors is crucial, as it directly influences regularization, identifiability, and computational feasibility in posterior inference.

Early studies primarily focused on Gaussian graphical models, a specialized yet important subclass of undirected graphical models, in which the matrix $\boldsymbol{\theta}$ is defined as the inverse of the covariance matrix and is referred to as the precision matrix. Dawid and Lauritzen (1993) introduced the hyper inverse Wishart (HIW) prior, which enjoys conjugacy as well as weak and strong hyper Markov properties. This prior was subsequently employed by Giudici and Green (1999) and Carvalho et al. (2007) in algorithms for Bayesian structure learning. A notable limitation, however, is that the HIW prior is only defined for decomposable graphs. To address this, Roverato (2002) extended the HIW distribution to general graphs, leading to the formulation of the G-Wishart prior.

An alternative approach that can be applied to non-Gaussian undirected graphical models and is computationally attractive is the spike-and-slab prior

method, which models edge parameters separately using a mixture distribution. The *spike* component, typically concentrated near zero (e.g., a Dirac mass or a narrow Gaussian), represents irrelevant parameters, while the *slab* component, given by a broader distribution such as Normal or Cauchy, captures relevant parameters. This formulation is often implemented via latent binary indicators $\gamma \in \{0, 1\}$ for each candidate edge, with $\gamma = 0$ denoting exclusion and $\gamma = 1$ inclusion. Wang (2015) proposed a continuous variant where the spike is represented by a normal distribution with small variance, yielding both flexibility and computational efficiency. This approach has proven to be highly efficient and scalable to large graphs.

1.4.2 Markov chain Monte Carlo algorithms for structure learning

One major challenge in Bayesian structure learning is the computation of the marginal likelihood $\mathbb{P}(\mathbf{X} \mid G)$, which is intractable for most nondecomposable graphs due to the presence of the normalization constant. Several approximation and sampling methods have been developed to address this problem. Some of this work uses Markov chain Monte Carlo (MCMC) methods to sample from the parameter posterior. Specifically, when using MCMC sampling, one can approximate the expected value of $h(\hat{G})$ for any information function $h : \mathcal{G} \to \mathbb{R}$ defined on graph space $\mathcal{G}$, and $\hat{G}$ denotes the estimated graph. For example, if we choose:

$$h(G) = \begin{cases} 1 & \text{an edge } e \in E \text{ of } G, \\ 0 & \text{otherwise,} \end{cases}$$

this will give the posterior edge inclusion probabilities

$$p_e := P(e \in \hat{G} \mid \mathbf{x}) = \mathrm{E}(h(\hat{G}) \mid \mathbf{x}).$$

By using alternative information functions h, one can estimate various posterior probabilities. For example, the probability that the true graph contains

more than 10 edges, the probability that it is connected, or the probability that it contains a cycle, among others. However, this flexibility comes at a computational cost: MCMC algorithms are resource intensive as they iteratively explore high-dimensional parameter spaces.

Let $\boldsymbol{\theta}^\star$ and $\boldsymbol{\theta}$ be the true and unknown parameter vector, respectively. We want to make inference on the posterior distribution $\mathbb{P}(\boldsymbol{\theta} \mid \mathbf{x})$. MCMC starts at one initial value of the parameter vector $\boldsymbol{\theta}^{(0)}$ and jumps to $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(L)}$ for some $r \in \mathbb{N}$. The sequence $(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(L)})$ is a Markov chain with $\boldsymbol{\theta}^{(s)}$ called the state of the Markov chain with respect to $s = 1, \ldots, L$. All states are elements of the state space $\mathcal{S}$. The Markov chain is characterized by a transition kernel $P(\mathbf{x}, A)$, which represents the transition probability to a set $A \in \mathcal{S}$ from the current state $\mathbf{x} \in \mathcal{S}$. By designing this transition kernel appropriately, it is possible to ensure that the chain converges to a stationary distribution. Tierney (1994) identifies three sufficient conditions for this convergence, with the most important being the detailed balance condition. This condition states that for any subset $A \subset \mathcal{S}$, the probability of the chain entering A must equal the probability of the chain leaving A. Now, if the equilibrium condition holds and if the stationary distribution is equal to the posterior distribution, the distribution of our Markov chain will arbitrarily get closer to the posterior distribution as we increase the chain length L. This remains the same regardless of the initial value $x = \boldsymbol{\theta}^{(0)}$. We can then get arbitrarily close to our desired expected value using:

$$\mathrm{E}(h(\boldsymbol{\theta}^\star) \mid \mathbf{x}) = \lim_{L \to \infty} \frac{1}{L} \sum_{s=1}^{L} h(\boldsymbol{\theta}^{(s)}) \approx \frac{1}{L} \sum_{s=1}^{L} h(\boldsymbol{\theta}^{(s)}),$$

for sufficiently large L.

1.5 General Bayes

Here, we summarize a general Bayesian framework proposed by Bissiri et al. (2016), where the update of the prior distribution is not necessarily based on the traditional likelihood. Instead, the parameter of interest is linked to

the data through a loss function. Specifically, the parameter $\boldsymbol{\theta}_0$ is defined as the minimizer of the expected loss with respect to the data-generating distribution:

$$\boldsymbol{\theta}_0 = \operatorname{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \int l(\boldsymbol{\theta}; \mathbf{x})\, \mathrm{d}F_0(\mathbf{x}), \tag{1.1}$$

where $\boldsymbol{\Theta}$ is the parameter space and $l(\cdot, \mathbf{x})$ denotes a loss function for each observation $\mathbf{x}$.

For finite samples $\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)})^\top$, with $\mathbf{x}^{(i)} = (x_{i1}, \ldots, x_{ip})$, the empirical version of (1.1) becomes

$$\sum_{i=1}^{n} l(\boldsymbol{\theta}; \mathbf{x}^{(i)}), \qquad \boldsymbol{\theta} \in \boldsymbol{\Theta}. \tag{1.2}$$

If l is chosen as the negative log-likelihood, $\boldsymbol{\theta}_0$ corresponds to the maximum likelihood estimator. Alternative choices such as pseudo-likelihood or robust losses yield M-estimators.

Based on this foundation, Bissiri et al. (2016) propose updating the prior distribution π via the loss function l as

$$\pi_{\mathbf{x}}(\boldsymbol{\theta}) \propto \exp\{-\lambda\, l(\boldsymbol{\theta}; \mathbf{x})\}\, \pi(\boldsymbol{\theta}), \tag{1.3}$$

where $\lambda > 0$ is a calibration parameter. This generalized posterior, also studied in Zhang (2006a,b); Jiang and Tanner (2008); Bissiri and Walker (2010, 2012), allows Bayesian inference even when the likelihood is intractable or unavailable, and facilitates the construction of robust estimators in complex settings.

The results in Bissiri et al. (2016) are limited to the case where $\boldsymbol{\Theta}$ is a finite probability space. Bissiri and Walker (2019) extend these results to $\boldsymbol{\Theta} \subset \mathbb{R}$ and reorganize the axiomatic basis of the framework. Importantly, the update rule (1.3) satisfies a natural coherence property: the same posterior distribution is obtained whether data are processed jointly or sequentially. For n observations $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, this property implies

$$\mathrm{d}\pi_{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(n)}}/\mathrm{d}\pi(\boldsymbol{\theta}) = \frac{\exp\{-\lambda \sum_{i=1}^{n} l(\boldsymbol{\theta}; \mathbf{x}^{(i)})\}}{\int_{\boldsymbol{\Theta}} \exp\{-\lambda \sum_{i=1}^{n} l(\tau; \mathbf{x}^{(i)})\} \mathrm{d}\pi(\tau)}. \tag{1.4}$$

The choice of the calibration parameter λ is a subtle and important consideration. Readers interested in a detailed discussion of this topic are referred to Bissiri et al. (2016); Holmes and Walker (2017); Syring and Martin (2018).

Chapter 2

Undirected graphical models for Poisson data: from model specifications to Bayesian structure learning

In this chapter, we present a novel specification for Poisson undirected graphical Models that overcomes some limitation of existing specifications, and study its properties. Moreover, we present a general Bayesian framework for structure learning. Our approach is inspired by the works of Bissiri et al. (2016) and Bissiri and Walker (2019), which allow updating the posterior distribution from the prior through a loss function rather than using the traditional likelihood.

2.1 A novel Poisson-based model specification

Consider a graph $G = (V, E)$ where $V = \{1, \dots, p\}$ represents the set of nodes, with each node i corresponding to a random variable $X_i \in \mathbf{X}$. When the observations are count data, a Poisson distribution is typically considered

as starting point for the specification of the model. On noting that, for any random vector $\mathbf{X}$, a joint distribution can be obtained by following the chain rule:

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \mathbb{P}(X_1 = x_1) \prod_{s=2}^{p} \mathbb{P}(X_s = x_s \mid \mathbf{X}_{[s]} = \mathbf{x}_{[s]}), \tag{2.1}$$

where $\mathbf{X}_{[s]} = (X_1, \ldots, X_{s-1})$, we assume that each conditional distribution in (2.1) is Poisson, with probability mass function

$$\begin{aligned} \mathbb{P}_{(\eta_s, \boldsymbol{\theta}_{[s]})}(X_s = x_s \mid \mathbf{X}_{[s]} = \mathbf{x}_{[s]}) = \exp \big\{ & \eta_s x_s + x_s \langle \boldsymbol{\theta}_{[s]}, \mathbf{x}_{[s]} \rangle - \log x_s! \\ & - D(\eta_s, \boldsymbol{\theta}_{[s]}, \mathbf{x}_{[s]}) \big\}, \end{aligned} \tag{2.2}$$

where $D(\eta_s, \boldsymbol{\theta}_{[s]}, \mathbf{x}_{[s]}) = \exp\{\eta_s + \langle \boldsymbol{\theta}_{[s]}, \mathbf{x}_{[s]} \rangle\}$, $\boldsymbol{\theta}_{[s]} := (\theta_{s1}, \ldots, \theta_{s(s-1)})$ denotes the set of conditional dependence parameters and $\langle \cdot, \cdot \rangle$ denotes the inner product. The graph structure is encoded by the sparsity pattern of the parameters θ_{st}. A missing edge between node s and node t corresponds to the condition $\theta_{st} = 0$, for $t \leq s - 1$.

By imposing the Poisson assumption on the first variable as well, we obtain the following joint distribution:

$$\begin{aligned} \mathbb{P}_{(\boldsymbol{\eta}, \boldsymbol{\theta})}(\mathbf{X} = \mathbf{x}) = \exp \Bigg\{ & \sum_{s \in V} (\eta_s x_s - \log(x_s!)) + \frac{1}{2} \sum_{s=1}^{p} \sum_{t \neq s} \theta_{st} x_s x_t \\ & - A(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{x}) \Bigg\}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_p)$, $\boldsymbol{\theta} = \{\theta_{st} : \forall \ s \neq t \in V\}$, and

$$A(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{x}) = \sum_{1 \leq t < s \leq p} e^{\eta_s} e^{\theta_{st} x_t}.$$

We refer to this model as a Chain-local Poisson graphical model (Chain-LPGM).

In the traditional model specification (see Allen and Liu, 2013), which we refer to as Bayes-LPGM in what follows, the conditional distribution of

each variable X_s given the remaining variables $\mathbf{X}_{V\setminus\{s\}}$ is assumed to follow a Poisson distribution, that is

$$\mathbb{P}_{(\eta_s,\boldsymbol{\theta}_s)}(X_s = x_s|\mathbf{X}_{V\setminus\{s\}} = \mathbf{x}_{V\setminus\{s\}}) = \exp\big\{\eta_s x_s + x_s\langle\boldsymbol{\theta}_s, \mathbf{x}_{V\setminus\{s\}}\rangle - \log x_s! - D(\eta_s, \boldsymbol{\theta}_s, \mathbf{x}_{V\setminus\{s\}})\big\}, \tag{2.4}$$

where $D(\eta_s, \boldsymbol{\theta}_s, \mathbf{x}_{V\setminus\{s\}}) = \exp\{\eta_s + \langle\boldsymbol{\theta}_s, \mathbf{x}_{V\setminus\{s\}}\rangle\}$ and $\boldsymbol{\theta}_s := (\theta_{st},\ t \in V\setminus\{s\})$ denotes the set of conditional dependence parameters. Here, $(s,t) \notin E$ corresponds to $\theta_{st} = \theta_{ts} = 0$. It is common, to further simplify the pairwise interactions, to assume that $\theta_{st} = \theta_{ts}$. When all node-conditional distributions follow univariate Poisson distributions as in equation (2.4), there exists a unique joint distribution that is consistent with these conditionals, given by:

$$\mathbb{P}_{(\boldsymbol{\eta},\boldsymbol{\theta})}(\mathbf{X} = \mathbf{x}) = \exp\left\{\sum_{s\in V}(\eta_s x_s - \log(x_s!)) + \frac{1}{2}\sum_{s=1}^{p}\sum_{t\neq s}\theta_{st}x_s x_t - A(\boldsymbol{\eta},\boldsymbol{\theta})\right\}, \tag{2.5}$$

where $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_p)$, $\boldsymbol{\theta} = \{\theta_{st} : \forall\ s \neq t \in V\}$ and $A(\boldsymbol{\eta},\boldsymbol{\theta})$ is the log-partition function that ensures normalization. Note that $A(\boldsymbol{\eta},\boldsymbol{\theta})$ must be finite to ensure that the distribution is properly normalized, i.e., it sums to one over the space $\mathbf{X} \in \{1, \ldots, \infty\}^p$. Specifically, the log-partition function is given by:

$$A(\boldsymbol{\eta},\boldsymbol{\theta}) = \log\left[\sum_{x_s,x_t=0}^{\infty}\exp\left(\sum_{s=1}^{p}(\eta_s x_s - \log(x_s!)) + \frac{1}{2}\sum_{s=1}^{p}\sum_{t\neq s}\theta_{st}x_s x_t\right)\right].$$

Notice that the term $\theta_{st}x_s x_t$ dominates the summation in the log-partition function. Therefore, to ensure that $A(\boldsymbol{\eta},\boldsymbol{\theta})$ is finite over the domain where $x_s, x_t \to \infty$, it is necessary that $\theta_{st} \leq 0, \forall s,t, \in V$ (Besag, 1974). This constraint imposes a limitation on the practical applicability of the model, particularly when modeling general multivariate count data that can exhibit positive and negative dependencies between variables.

While the joint distribution in (2.3) closely resembles that in (2.5), the principal distinction lies in the normalizing constant: $A(\boldsymbol{\eta}, \boldsymbol{\theta})$ in (2.5) is a normalizing constant, whereas $A(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{x})$ in (2.3) is data-dependent, varying with $\mathbf{x}$. It is worth noting that:

1. (2.3) does not belong to the exponential family, while (2.5) does;

2. (2.3) does not impose any restriction on the signs of the interaction parameters, a restriction which is imposed by (2.5);

3. the pattern of zeros in $\boldsymbol{\theta}$ appearing in (2.3) is independent of the ordering of variables in (2.1) and matches the pattern of zeros in (2.5);

4. unless $[s] = V \setminus \{p\}$, the conditional distributions in (2.2) differ from those implied by the joint distribution in (2.5);

5. The structure of the graph G ultimately leads to the idea of choosing an appropriate variable ordering when applying the chain rule.

We are now going to prove statement 5. Before presenting our theorem, we introduce several results useful to this aim. Proofs of these results can be found in Lauritzen (1996).

Proposition 2.1.1. *(Proposition 2.17 in Lauritzen (1996)) Let G be an undirected decomposable graph. Then, for any $i \in V$, the vertices of G admit a perfect numbering with $v_1 = i$.*

Theorem 2.1.2. *(Lemma 2.14 in Lauritzen (1996)) For an ordering $v_1, \ldots, v_p$ of the nodes of $G = (V, E)$, define*

$$D_j = \{v_1, \ldots, v_j\}, \quad B_j = \big(\{v_j\} \cup N(v_j)\big) \cap D_j,\ j = 1, \ldots, p,$$

where $N(v_j)$ denotes the set of neighbors of v_j in G. The sets B_j, as defined above, form a sequence that contains all cliques of G and satisfies the running intersection property.

Lemma 2.1.3. *(Lemma 2.13 in Lauritzen (1996)) Let $B_1, \ldots, B_k$ be a perfect sequence. Assume that $C_t \subseteq C_m$ for some $t \neq m$ and that m is minimal with this property for fixed t. Then:*

- *if $m < t$ then $B_1, \ldots, B_{t-1}, B_{t+1}, \ldots, B_k$ is a perfect sequence;*
- *if $m > t$ then $B_1, \ldots, B_{t-1}, B_m, B_{m+1}, \ldots, B_{m-1}, B_{m+1}, \ldots, B_k$ is a perfect sequence.*

We can now prove statement 5. The following theorem holds.

Theorem 2.1.4. *Let $G = (V, E)$ be a decomposable graph, and let $v_1, \ldots, v_p$ be a perfect numbering of the nodes. Form the joint distribution by the chain rule (2.1) along that ordering:*

$$\mathbb{P}_{(\boldsymbol{\eta},\boldsymbol{\theta})}(\mathbf{X} = \mathbf{x}) \;=\; \prod_{j=1}^{p} \phi_j\big(\mathbf{x}_{B_j}\big),$$

where terms

$$\phi_j\big(\mathbf{x}_{B_j}\big) = \mathbb{P}_{(\eta_{v_j},\boldsymbol{\theta}_{[v_j]})}(X_{v_j} = x_{v_j} \mid \mathbf{X}_{D_j\setminus\{v_j\}} = \mathbf{x}_{D_j\setminus\{v_j\}})$$

follow (2.2). Then $\mathbb{P}$ factorizes as a product of nonnegative clique potentials (with separator corrections) and is therefore Markov with respect to G.

For simplifying notation, in the following proof we remove subscripts denoting the parameters of the distributions.

Proof. Let $C_1, \ldots, C_k$ denote a perfect sequence of cliques obtained from the sequence $B_1, \ldots, B_p$ by removing redundant sets using the Lemma 2.1.3. We start from the chain-rule product

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \;=\; \prod_{j=1}^{p} \phi_j\big(\mathbf{x}_{B_j}\big).$$

For each clique C_ℓ define the combined (unnormalized) clique potential

$$\Psi_{C_\ell}(x_{C_\ell}) \;:=\; \prod_{j:\, B_j = C_\ell} \phi_j\big(x_{B_j}\big).$$

Then the product over all j can be rewritten as the product over clique-potentials (with multiplicities):

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \;=\; \prod_{\ell=1}^{k} \Psi_{C_\ell}(\mathbf{x}_{C_\ell}).$$

The running intersection property guarantees the following standard junction-tree identity for decomposable graphs: there exist separator sets $S_2, \ldots, S_k$ (with $S_\ell = C_\ell \cap (C_1 \cup \cdots \cup C_{\ell-1}) \subset C_j$ for some $j \leq \ell - 1$) such that each separator S appears with multiplicity $m(S)$, equal to the number of times it occurs among the S_ℓ.
By telescoping (or equivalently, by induction on the number of cliques), we can express the product of the Ψ_{C_ℓ} as

$$\prod_{\ell=1}^{k} \Psi_{C_\ell}(\mathbf{x}_{C_\ell}) \;=\; \frac{\prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)}{\prod_{S \in \mathcal{S}} \chi_S(\mathbf{x}_S)^{\,m(S)-1}},$$

for suitable nonnegative functions $\{\psi_C\}_{C \in \mathcal{C}}$ and $\{\chi_S\}_{S \in \mathcal{S}}$, which are defined as appropriate products and ratios of the constituent functions $\{\phi_j\}$. A direct constructive way is to set $\psi_{C_1} = \Psi_{C_1}$ and for $\ell \geq 2$ define $\psi_{C_\ell} = \Psi_{C_\ell}$ while dividing out by the appropriate separator factors already accounted for in earlier cliques; the running intersection property ensures this division is well-defined and depends only on the separator variables. Hence $\mathbb{P}$ factorizes into a product of clique potentials divided by separator potentials raised to known multiplicities.

Because this is exactly the (positive) clique–separator factorization for decomposable graphs, the joint distribution satisfies the factorization criterion of the Hammersley–Clifford theorem and therefore is Markov with respect to G.

□

It is worth noting that a theorem similar to Theorem 2.1.4 can be stated also outside the Poisson case and, more broadly, outside the discrete case, provided that

- for each $j = 1, \ldots, p$ the conditional density of $X_{v_j} \mid \mathbf{x}_{D_j \setminus \{v_j\}}$ has the form

$$p(x_{v_j} \mid \mathbf{x}_{D_j \setminus \{v_j\}}) = \phi_j\big(\mathbf{x}_{B_j}\big);$$

- for every $\mathbf{x}$ in the support we have $\phi_j(\mathbf{x}_{B_j}) > 0$ for all j.

2.2 Structure learning

A widely adopted approach to structure learning is known as neighborhood selection (see for example Meinshausen and Bühlmann, 2006). This strategy aims to recover the graph G by estimating the neighborhood $N(s)$ of each node s. In a frequentist setting, estimating $N(s)$ typically amounts to solving a (penalized) variable selection problem in a regression setting (Allen and Liu, 2013). Hue and Chiogna (2021) proposed a novel algorithm for learning the structure of undirected graphical models from count data (potentially right-truncated), replacing regression-based strategies with hypothesis testing, in line with the principles of the PC algorithm (Spirtes et al., 2000). To infer a graphical model from data, PC-like algorithms rely on sequences of conditional independence tests. Hue and Chiogna (2021) demonstrated that their method achieves high-dimensional consistency when conditional independencies are assessed using Wald-type tests. While this ensures asymptotic correctness, the algorithm's finite-sample performance may be highly sensitive to the significance level employed.

In this work, we adopt a Bayesian perspective on neighborhood selection, based on the general Bayes framework described in Section 1.5. This approach is particularly beneficial in the context of Bayes-LPGM, where the joint distribution of the Poisson graphical models may be unavailable, computationally intractable due to the present of partition function, or where concerns arise regarding the robustness of the employed statistical procedure, as it allows for the use of loss functions that are not necessarily derived from the likelihood. Readers interested in an alternative Bayesian approach may refer to Banerjee and Ghosal (2015).

To facilitate structure learning, we can, without loss of generality, set $\boldsymbol{\eta} = \mathbf{0}$ in both (2.3) and (2.5). We consider a spike and slab prior distribution for the parameters in the vector $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_{[s]}$, assuming that the elements θ_{st} are i.i.d. for every pair (s,t) with $s,t \in V$ and $s \neq t$. This prior is constructed following the Normal-mixture of inverse Gamma formulation introduced in Ishwaran and Rao (2005), and is defined as

$$\begin{aligned}
\theta_{st} \mid \gamma_{st}, \tau_{st}^2 &\overset{\text{prior}}{\sim} N\left(0, v^2\right) \text{ with } v^2 = \tau_{st}^2 \gamma_{st}, \\
\gamma_{st} \mid \omega &\overset{\text{prior}}{\sim} \omega\delta_1(\gamma_{st}) + (1-\omega)\delta_{v_0}(\gamma_{st}), \\
\tau_{st}^2 &\overset{\text{prior}}{\sim} \Gamma^{-1}\left(a_\tau, b_\tau\right), \\
\omega &\overset{\text{prior}}{\sim} \text{Beta}\left(a_\omega, b_\omega\right).
\end{aligned}$$

Here, $\delta_x(y)$ denotes a Dirac function that is 1 in x and 0 everywhere else and v_0 is some small positive constant. This means that the effective prior variance v^2 is very small if $\gamma_{st} = 0$ - this is the spike part of the prior. The variance τ_{st}^2 is sampled from an informative Inverse Gamma (Γ^{-1}) prior with density $p(x \mid a, b) = \frac{b^a}{\Gamma(a)} x^{-(a+1)} \exp\left(-\frac{b}{x}\right)$.

We consider the following loss function

$$l(\boldsymbol{\gamma}; \mathbf{x}) = -\sum_{s \in V} \log \prod_{t \in A} \zeta(\gamma_{st}; \mathbf{x}),$$

where $A = \{1, \ldots, s-1\}$ for Chain-LPGM and $A = V \setminus \{s\}$ for Bayes-LPGM, and $\zeta(\gamma_{st}; \mathbf{x})$ is computed by first evaluating the following quantity

$$\begin{aligned}
\zeta(\mathbf{x}; \gamma_{st}; \omega) = \prod_{i=1}^{n} \int \mathbb{P}_{\boldsymbol{\theta}_s}(X_s = x_{is} \mid \mathbf{X}_A = \mathbf{x}_A^{(i)}) \phi(\theta_{st} \mid 0, \tau_{st}^2 \gamma_{st}) p(\tau_{st}^2 \mid a_\tau, b_\tau) \\
\prod_{j \in A \setminus \{t\}} \phi(\theta_{sj} \mid 0, \tau_{sj}^2 \gamma_{sj}) p(\tau_{sj}^2 \mid a_\tau, b_\tau) g(\gamma_{sj} \mid \omega) \mathrm{d}\theta_{sj} \mathrm{d}\tau_{sj} \mathrm{d}\gamma_{sj} \mathrm{d}\theta_{st} \mathrm{d}\tau_{st}.
\end{aligned}$$

Here, $\phi(x \mid \mu, \sigma^2)$ denotes the density of the normal distribution with mean μ and variance σ^2 evaluated in x; $p(x \mid a, b)$ denotes the density of the inverse gamma distribution with shape parameter a and scale parameter b evaluated at x; and $g(x \mid \pi)$ denotes the density of the Bernoulli distribution with success probability π evaluated at x.

To obtain the conditional density of $\mathbf{x}$ given γ_{st}, we need to multiply it by the conditional density of ω given γ_{st} and integrate w.r.t ω. We have

$$\begin{aligned} P(\omega \mid \gamma_{st}) &= \frac{P(\gamma_{st} \mid \omega)P(\omega)}{\int_0^1 P(\gamma_{st} \mid \omega)P(\omega)\mathrm{d}\omega} \\ &= \frac{(\omega\delta_1(\gamma_{st}) + (1-\omega)\delta_{v_0}(\gamma_{st}))\dfrac{1}{B(a_\omega, b_\omega)}\omega^{a_\omega - 1}(1-\omega)^{b_\omega - 1}}{\int_0^1 (\omega\delta_1(\gamma_{st}) + (1-\omega)\delta_{v_0}(\gamma_{st}))\dfrac{1}{B(a_\omega, b_\omega)}\omega^{a_\omega - 1}(1-\omega)^{b_\omega - 1}\mathrm{d}\omega} \\ &= \frac{\omega^{a_\omega + \delta_1(\gamma_{st}) - 1}(1-\omega)^{b_\omega + \delta_{v_0}(\gamma_{st}) - 1}}{\int_0^1 \omega^{a_\omega + \delta_1(\gamma_{st}) - 1}(1-\omega)^{b_\omega + \delta_{v_0}(\gamma_{st}) - 1}\mathrm{d}\omega}. \end{aligned}$$

This corresponds to a Beta distribution with updated shape parameters $a_\omega + \delta_1(\gamma_{st})$ and $b_\omega + \delta_{v_0}(\gamma_{st})$. Here, for Bayes-LPGM, the loss function for γ is constructed based on the conditional distributions, or more precisely, the pseudo-likelihood of the parameter $\boldsymbol{\theta}$, which serves as an approximation to the traditional likelihood. In contrast, for Chain-LPGM, we work directly with the true likelihood. Therefore, it is reasonable to set $\lambda = 1$ in equation (1.3) for both model specifications. This choice can be seen as balancing the influence of the prior and the observed data in a way that is natural when the loss is fundamentally derived from the likelihood. Consequently, the negative logarithm defines the loss function l is given by:

$$\exp\{-l(\gamma_{st}; \mathbf{x})\} = \psi(\mathbf{x}; \gamma_{st})$$

where

$$\psi(\mathbf{x}; \gamma_{st}) = \int_0^1 \zeta(\mathbf{x}; \gamma_{st}; \omega) f_{\mathrm{Beta}}(\omega \mid a_\omega + \delta_1(\gamma_{st}), b_\omega + \delta_{v_0}(\gamma_{st}))\mathrm{d}\omega,$$

The corresponding general posterior of γ_{st} is given by:

$$\pi(\gamma_{st} = 1 \mid \mathbf{x}) = \frac{\psi(\mathbf{x}; 1)}{\psi(\mathbf{x}; \nu_0) + \psi(\mathbf{x}; 1)}.$$

2.3 Inference

We estimate the posterior distributions $\pi(\gamma_{st} = w_{st} \mid \mathbf{x})$ by performing independent Poisson regressions. For each $s \in V$, we fit a Poisson regression

model of X_s on $\mathbf{X}_A$:

$$\log \mathrm{E}(X_s = x_s | \mathbf{X}_A = \mathbf{x}_A) = \langle \boldsymbol{\theta}_A, \mathbf{x}_A \rangle, \tag{2.6}$$

where $A = \{1, \ldots, s-1\}$ or $A = V \setminus \{s\}$, depending on whether Chain-LPGM or Bayes-LPGM is applied, respectively, and $\boldsymbol{\theta}_A = (\theta_{si} : i \in A)$, as outlined in the following pseudo-code Algorithm 1

Algorithm 1:

Input: n independent realizations of the p-random vector $\mathbf{X} : \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$.
Output: An estimated undirected graph $\hat{G} = (V, \hat{E})$.
Choose a permutation σ_V of the index set V. **for** $s \in \sigma_V$ **do**
    Run the stochastic search variable selection for the Poisson regression of X_s against $\mathbf{X}_A$
end for
for $s \in \sigma_V$ **do**
    for $t \in \{1, \ldots, s-1\}$ **do**
        if $A = V \setminus s$ **then**
            (Bayes-LPGM)
            set $(s,t) \in \hat{E}$ if $\pi(\gamma_{st} = 1 | \mathbf{x}) \geq 0.5$ and $\pi(\gamma_{ts} = 1 | \mathbf{x}) \geq 0.5$
        end if
        else
            (Chain-LPGM)
            set $(s,t) \in \hat{E}$ if $\pi(\gamma_{st} = 1 | \mathbf{x}) \geq 0.5$
        end if
    end for
end for

The difference between the two specifications lies in the number of regressions performed and the set of conditioning variables, that is, the conditional set A. Under Chain-LPGM, the number of covariates in the model $p-1$ increases incrementally from 1 to $p-1$. In contrast, under Bayes-LPGM,

where $A = V \setminus \{s\}$, each of the p regressions involves $p-1$ covariates. In the Chain-LPGM, the dimensionality of the conditional models is lower than in Bayes-LPGM, resulting in reduced computational complexity.

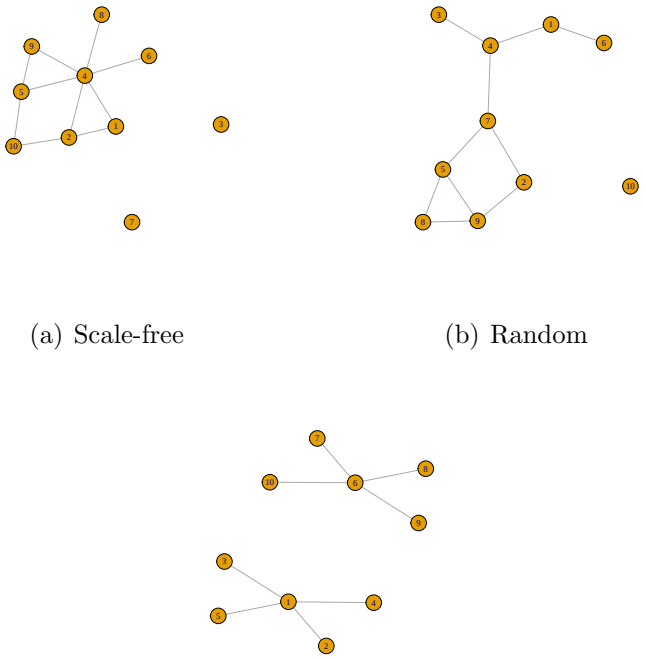
The sampling algorithm adherees to the procedure outlined in Fabian Scheipl and Kneib (2012), and is presented in detail in Appendix A.1 for the reader's convenience, with the necessary notational modifications. Implementation of this procedure is found in the **spikeSlabGAM** package (see Scheipl, 2011, for more information) .

2.4 Simulation studies

2.4.1 Data generation

In this study, we consider three distinct graph structures for two different dimensional settings, specifically $p = 10$ and $p = 100$. The considered structures are: (i) a scale-free graph characterized by a power-law degree distribution, (ii) a random graph with edges drawn independently and identically as Bernoulli random variables, and (iii) a hub graph in which each node is connected to one of several central nodes. For the scale-free network, a power-law distribution with parameter 0.01 is assumed for the node degrees. In the case of hub graphs, two hub nodes are considered for $p = 10$, and fifteen hub nodes for $p = 100$. The construction of random networks is performed using the `igraph` package in `R`, with edge probabilities set to 0.2 for $p = 10$ and 0.02 for $p = 100$. The hub networks are constructed using the `XMRF` package in `R`. Detailed visual representations of the selected graphs for $p = 10$ and $p = 100$ are provided in Figures 2.1 and 2.2, respectively.

For each graph structure, we generate 500 datasets for each combination of sample size and dimension, considering $n = 100$, 200, 500 for $p = 10$, and $n = 500$, 1000, 1500 for $p = 100$. We assume $\theta_{st} = -1$ for each edge between nodes s and t to ensure that the joint distribution also exists under the classical model specification. To explore the impact of value for dependence parameters, we also consider scenarios in which approximately half of the



(a) Scale-free (b) Random

(c) Hub

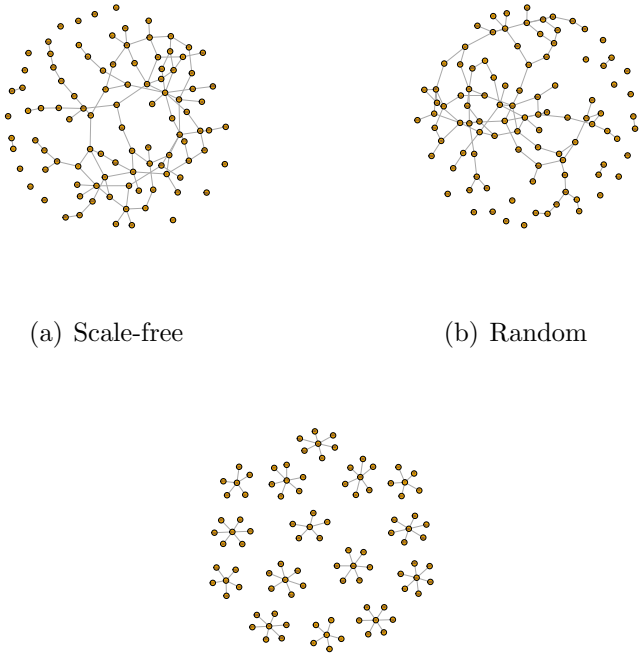
Figure 2.1: The graph structures for $p = 10$ employed in the simulation studies.

parameters θ_{st} are set to -1 and the other half to -0.5.

Data generation is based on the approach proposed in Zhang et al. (2021), implemented via the `ModPGMInference` package in `R`. It is worth noting that this package generates data based on the classical model specification.

2.4.2 Results

Our objective is to evaluate the effectiveness of the algorithm in accurately recovering the underlying graph structure. To this end, we consider two evaluation metrics: the Positive Predictive Value (PPV), which represents the proportion of correctly identified edges among all edges selected by the algorithm, and the Sensitivity (Se), which denotes the proportion of true



(a) Scale-free (b) Random

(c) Hub

Figure 2.2: The graph structures for $p = 100$ employed in the simulation studies.

edges that are correctly identified by the algorithm. The definitions of the evaluation metrics True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) are provided in Table 2.4.2. Then, the Positive Predictive Value (PPV) and Sensitivity (Se) are defined as follows

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \;\; \text{Se} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Another important aspect to consider is that the choice of hyperparameter in the Bayesian framework can significantly influence the results, and improved performance may be achieved through appropriate selection. To assess this aspect, we examine two different configurations for the hyperparameter pair (a_τ, b_τ). Specifically, we consider $(a_\tau, b_\tau) = (5, 25)$ and

		True graph	
		Edge	No edge
Estimated graph	Edge	True Positive (TP)	False Positive (FP)
	No edge	False Negative (FN)	True Negative (TN)

Table 2.1: Definition of performance metrics.

$(a_\tau, b_\tau) = (3, 50)$. In all simulations, the parameter v_0 is fixed at 0.00025. We also compare our Bayesian approach with the frequentist method, PC-LPGM developed by Hue and Chiogna (2021).

Tables 2.2, 2.3, 2.4, and 2.5 report the Monte Carlo means of TP, PPV, Se, and computational time for the algorithm in two different cardinality settings. The reported computational time corresponds to the runtime for a single dataset. Additionally, Table 2.6 presents results for the case $p = 10$ with parameter values set to either -1 or -0.5, aiming to evaluate the influence of parameter magnitude on the algorithm's performance. The runtime measurements were conducted on an AMD Epyc 7763 2.45GHz processor running R 4.2.1 utilizing 16 cores.

type	n	PC-LPGM				Bayesian approaches							
						Chain-LPGM				Bayes-LPGM			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	100	6.598	0.928	0.660	0.035	5.122	0.899	0.512	0.176	2.903	0.976	0.290	0.231
	200	9.114	0.971	0.911	0.040	9.208	0.891	0.920	0.267	8.438	0.986	0.844	0.396
	500	9.922	0.990	0.992	0.051	10.000	0.800	1.000	0.530	10.000	0.982	1.000	0.782
Scale-free	100	3.232	0.903	0.323	0.032	2.866	0.894	0.287	0.180	1.840	0.958	0.184	0.294
	200	4.270	0.943	0.427	0.039	5.724	0.920	0.572	0.269	4.562	0.972	0.456	0.411
	500	7.004	0.984	0.700	0.048	8.654	0.902	0.866	0.542	7.330	0.979	0.733	0.800
Hub	100	1.935	0.733	0.242	0.031	1.855	0.859	0.232	0.184	1.121	0.874	0.140	0.230
	200	5.286	0.924	0.661	0.040	5.726	0.930	0.716	0.284	2.911	0.927	0.364	0.362
	500	7.884	0.974	0.986	0.047	7.994	0.952	0.999	0.574	7.598	0.974	0.950	0.775

Table 2.2: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 100,\ 200,\ 500$ from graph with $p = 10$ variables with Poisson node conditional distribution with $(a_\tau, b_\tau) = (5, 25),\ v_0 = 0.00025$ and $\theta_{st} = -1,\ \forall\ (s,t) \in E$.

For case $p = 10$, the results reported in Table 2.2 demonstrate that the

type	n	PC-LPGM				Bayesian approaches							
						Chain-LPGM				Bayes-LPGM			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	100	6.598	0.928	0.660	0.035	7.620	0.894	0.762	0.159	6.630	0.978	0.663	0.234
	200	9.114	0.971	0.911	0.040	9.840	0.890	0.984	0.244	9.690	0.986	0.970	0.365
	500	9.922	0.990	0.992	0.051	10.000	0.868	1.000	0.502	10.000	0.996	1.000	0.774
Scale-free	100	3.232	0.903	0.323	0.032	4.306	0.898	0.431	0.160	3.552	0.970	0.355	0.243
	200	4.270	0.943	0.427	0.039	6.420	0.925	0.642	0.248	5.382	0.984	0.538	0.379
	500	7.004	0.984	0.700	0.048	9.074	0.995	0.907	0.537	8.730	0.999	0.873	0.800
Hub	100	1.935	0.733	0.242	0.031	3.147	0.888	0.393	0.178	1.892	0.886	0.236	0.241
	200	5.286	0.924	0.661	0.040	6.539	0.953	0.817	0.282	5.205	0.978	0.651	0.370
	500	7.884	0.974	0.986	0.047	7.916	0.981	0.990	0.578	7.852	0.998	0.982	0.797

Table 2.3: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 100,\ 200,\ 500$ from graph with $p = 10$ variables with Poisson node conditional distribution with $(a_\tau, b_\tau) = (3, 50),\ v_0 = 0.00025$ and $\theta_{st} = -1,\ \forall\ (s,t) \in E$.

type	n	PC-LPGM				Bayesian approaches							
						Chain-LPGM				Bayes-LPGM			
		TP	PPV	Se	Time(mins)	TP	PPV	Se	Time(mins)	TP	PPV	Se	Time(mins)
Random	500	96.582	0.883	0.976	0.044	96.480	0.776	0.975	5.772	93.088	0.973	0.940	12.172
	1000	98.504	0.950	0.995	0.059	98.990	0.774	0.999	11.848	98.902	0.989	0.999	25.670
	1500	98.878	0.974	0.986	0.072	99.000	0.785	1.000	17.312	99.000	0.996	1.000	44.770
Scale-free	500	91.076	0.872	0.911	0.042	91.198	0.762	0.912	5.769	85.55	0.969	0.856	12.299
	1000	95.988	0.946	0.960	0.060	98.550	0.764	0.986	11.827	97.010	0.989	0.970	25.706
	1500	97.530	0.972	0.975	0.073	99.702	0.771	0.997	17.277	99.234	0.996	0.992	44.851
Hub	500	24.674	0.498	0.290	0.057	45.756	0.704	0.538	6.039	18.278	0.788	0.215	13.434
	1000	60.040	0.836	0.706	0.062	72.808	0.892	0.857	12.292	46.308	0.983	0.545	26.383
	1500	76.652	0.915	0.902	0.087	79.918	0.932	0.940	18.086	65.896	0.998	0.775	46.048

Table 2.4: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 500,\ 1000,\ 1500$ from graph with $p = 100$ variables with Poisson node conditional distribution with hyperparameter $(a_\tau, b_\tau) = (5, 25),\ v_0 = 0.00025$ and $\theta_{st} = -1,\ \forall\ (s,t) \in E$.

type	n	PC-LPGM				Bayesian approaches							
						Chain-LPGM				Bayes-LPGM			
		TP	PPV	Se	Time(mins)	TP	PPV	Se	Time(mins)	TP	PPV	Se	Time(mins)
Random	500	96.582	0.883	0.976	0.044	97.572	0.939	0.986	5.724	95.142	0.999	0.961	13.610
	1000	98.504	0.950	0.995	0.059	98.916	0.977	0.999	11.318	98.718	0.999	0.997	27.883
	1500	98.878	0.974	0.986	0.072	98.982	0.987	0.999	17.264	98.952	0.999	0.999	45.920
Scale-free	500	91.076	0.872	0.911	0.042	93.874	0.931	0.939	5.659	87.818	0.999	0.878	13.655
	1000	95.988	0.946	0.960	0.060	97.638	0.966	0.976	11.383	94.742	0.999	0.947	27.928
	1500	97.530	0.972	0.975	0.073	98.656	0.976	0.987	17.089	96.926	0.999	0.969	45.782
Hub	500	24.674	0.498	0.290	0.057	26.678	0.799	0.314	5.966	14.010	0.990	0.165	14.122
	1000	60.040	0.836	0.706	0.062	30.940	0.859	0.364	11.787	18.686	0.999	0.220	29.005
	1500	76.652	0.915	0.902	0.087	31.860	0.856	0.375	18.053	19.220	0.999	0.226	47.064

Table 2.5: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 500,\ 1000,\ 1500$ from graph with $p = 100$ variables with Poisson node conditional distribution with hyperparameter $(a_\tau, b_\tau) = (3, 50),\ \ v_0 = 0.00025$ and $\theta_{st} = -1,\ \forall\ (s,t) \in E$.

Type	n	Chain-LPGM				Bayes-LPGM			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	100	5.530	0.914	0.553	0.171	4.454	0.966	0.445	0.251
	200	8.632	0.933	0.863	0.263	8.11	0.987	0.811	0.382
	500	9.930	0.953	0.993	0.537	9.920	0.995	0.992	0.805
Scale-free	100	3.146	0.893	0.315	0.177	2.370	0.946	0.237	0.249
	200	6.036	0.946	0.604	0.266	5.240	0.981	0.524	0.389
	500	8.656	0.983	0.866	0.546	8.194	0.999	0.819	0.819
Hub	100	3.418	0.910	0.427	0.174	2.128	0.925	0.266	0.253
	200	5.826	0.957	0.728	0.272	4.008	0.952	0.501	0.393
	500	7.028	0.985	0.879	0.566	6.890	0.998	0.861	0.830

Table 2.6: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 100,\ 200,\ 500$ from graph with $p = 10$ variables with Poisson node conditional distribution with $(a_\tau, b_\tau) = (3, 50),\ \ v_0 = 0.00025$ for parameters θ_{st} equal to -1 or -0.5.

PC-LPGM algorithm outperforms the Bayes-LPGM, particularly in settings with a limited sample size, namely $n = 100$. However, as the sample size increases, the performance gap between the two methods narrows. For example, in the case of a random graph with a sample size of 500, Bayes-LPGM achieves a sensitivity (Se) of 1, slightly exceeding the 0.992 obtained by PC-LPGM, while maintaining a high positive predictive value (PPV) of 0.982, which remains close to the 0.990 recorded for PC-LPGM. In contrast, Chain-LPGM consistently yields a higher sensitivity than Bayes-LPGM and achieves a performance comparable to PC-LPGM even for smaller sample sizes. However, this improvement in sensitivity comes at the cost of a lower PPV compared to Bayes-LPGM. This trade-off may be attributed to the fact that the data generation process does not strictly follow Chain-LPGM, and the corresponding joint distribution may not factorize with respect to the graph G.

Table 2.3 demonstrates that tuning the hyperparameter leads to significant improvements in the performance of the Bayesian approaches, particularly under small sample settings ($n = 100$). For example, for Bayes-LPGM, the number of true positives (TP) increases considerably across all network types: from 2.903 to 6.630 in the random graph, from 1.840 to 3.552 in the scale-free graph, and from 1.121 to 1.892 in the hub graph. A similar trend is evident for Chain-LPGM, where both TP and sensitivity (Se) values improve markedly with the modified hyperparameter configuration. In particular, in the scale-free graph scenario with a larger sample size ($n = 500$), the Bayesian approaches outperform the PC-LPGM method. Specifically, Chain-LPGM and Bayes-LPGM achieve Se values of 0.907 and 0.873, respectively, which are substantially higher than the 0.700 reported for PC-LPGM. At the same time, both algorithms maintain high positive predictive values (PPV), with values of 0.995 and 0.999, respectively, compared to 0.984 for PC-LPGM.

Overall, the Bayesian methods demonstrate performance comparable to the frequentist PC-LPGM approach in terms of true positives (TP), positive predictive value (PPV), and sensitivity (Se), although they generally require

longer computational time. The sensitivity of the Bayesian approaches to the choice of hyperparameter is particularly evident in small-sample settings. By selecting appropriate hyperparameter values, the Bayesian methods can outperform the PC-LPGM algorithm in terms of estimation accuracy.

For the case $p = 100$, the results presented in Tables 2.4 and 2.5 for the random and scale-free graphs indicate comparable performance between the PC-LPGM algorithm and the Bayesian approaches. Bayes-LPGM consistently achieves high PPV values across all settings. In contrast, the PPV of Chain-LPGM shows a marked improvement under the hyperparameter configuration $(a_\tau, b_\tau) = (3, 50)$, as reported in Table 2.5. The impact of hyperparameter choices on the performance of Bayes-LPGM, however, appears less pronounced for these two types of graph structures.

The differences between the approaches become particularly evident in the case of the hub structure. The results reported in Table 2.4 demonstrate the superiority of the Bayesian methods over the PC-LPGM algorithm, especially in settings with limited sample sizes. Specifically, for a sample size of 500, Bayes-LPGM achieves a PPV of 0.788, significantly higher than the 0.498 obtained by PC-LPGM, while its sensitivity (Se) is slightly lower at 0.215 compared to 0.290. In contrast, Chain-LPGM shows a clear advantage, with a PPV of 0.704 and a substantially higher Se of 0.538. The influence of the hyperparameter values is also evident in the case of the hub graph. While the PPV improves when the hyperparameter setting for the variance changes from $(a_\tau, b_\tau) = (5, 25)$ to $(a_\tau, b_\tau) = (3, 50)$, this adjustment results in a substantial reduction in the number of true positives (TP) and sensitivity (Se).

Regarding computational efficiency, as anticipated, the PC-LPGM algorithm exhibits substantially lower running times in comparison to the Bayesian approaches. Within the Bayesian framework, Chain-LPGM proves to be more efficient than Bayes-LPGM. This improvement is primarily attributed to the reduced number of dependent variables considered in each regression, as outlined in Section 2.3, thereby lowering the overall computa-

tional burden.

To examine the impact of parameter values, we compare the results in Table 2.3, where all parameters are fixed at $\theta_{st} = -1$, with those in Table 2.6, where the parameters are set to either -1 or -0.5, using the same hyperparameter values in both cases. The results suggest that variations in parameter values do not substantially affect the performance of the algorithms. This underscores the robustness of the proposed approaches under different parameterizations, particularly for larger sample sizes, where the performance metrics converge closely across all settings.

The performance of Chain-LPGM may be influenced by the order in which variables are selected for applying the chain rule in (2.1). To investigate this, we consider different permutations of variable orderings for Chain-LPGM in the case $p = 100$, with sample sizes of 500, 1000, and 1500. Table 2.7 reports the Monte Carlo means and standard deviations of the chosen metrics based on 50 different permutations. While some variation is observed, the results demonstrate the algorithm's ability to maintain stable performance across permutations, highlighting its reliability in handling variable ordering under varying conditions.

Type	n	TP	PPV	Se
Random	500	98.080(0.752)	0.937(0.021)	0.991(0.008)
	1000	99.000(0.000)	0.983(0.013)	1.000(0.000)
	1500	99.000(0.000)	0.977(0.016)	1.000(0.000)
Scale-free	500	90.920(1.412)	0.930(0.026)	0.909(0.014)
	1000	96.980(1.363)	0.974(0.015)	0.970(0.014)
	1500	99.560(0.541)	0.976(0.016)	0.995(0.005)
Hub	500	23.880(2.677)	0.797(0.067)	0.281(0.031)
	1000	43.020(3.857)	0.911(0.039)	0.506(0.045)
	1500	53.440(5.697)	0.916(0.035)	0.629(0.067)

Table 2.7: Monte Carlo means (standard deviations) of TP, PPV, and Se for 50 permutations of the order variable with $\theta_{st} = -1$, $\forall\ (s,t) \in E$ using Chain-LPGM with $(a_\tau, b_\tau) = (3, 50)$, $v_0 = 0.00025$.

2.4.3 Sensitivity to hyperparameter selection

To evaluate the sensitivity of the reconstruction results to the choice of hyperparameter, we investigated the number of edges identified by Chain-LPGM and Bayes-LPGM across different values of (a_τ, b_τ), where $a_\tau \in \{1, \ldots, 10\}$ and $b_\tau \in \{40, \ldots, 60\}$. Specifically, we considered the task of recovering the structure of random graphs with $p = 10$.

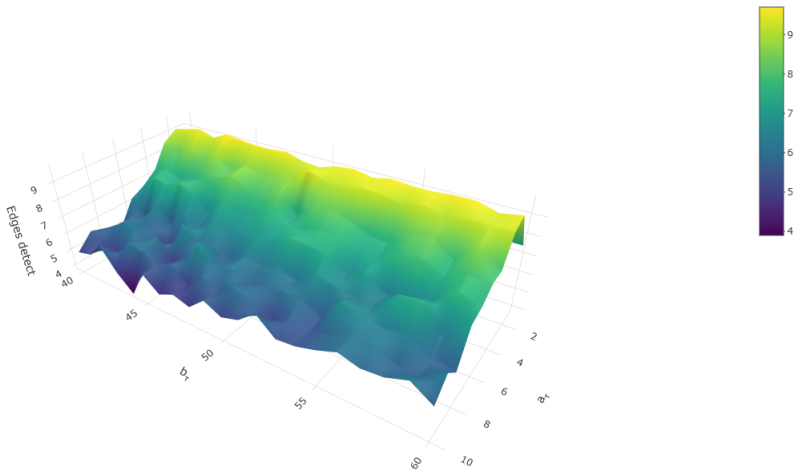


Figure 2.3: Average number of edges estimated by Chain-LPGM across 10 simulated datasets under different hyperparameter settings, for a random graph with $p = 10$ and a sample size of 100.

The results illustrated in Figure 2.3 demonstrate the sensitivity of Chain-LPGM to the choice of hyperparameter. For instance, when $(a_\tau, b_\tau) = (2, 58)$, the average number of detected edges is 9.6, whereas for $(a_\tau, b_\tau) = (10, 58)$, this number drops to only 5.9. It is also noteworthy that, for a fixed value of a_τ, the number of detected edges remains relatively stable across different values of b_τ.

A similar phenomenon is observed for the Bayes-LPGM algorithm, as shown in Figure 2.4. For example, when $(a_\tau, b_\tau) = (2, 53)$, the average number of detected edges is 7.6, whereas with $(a_\tau, b_\tau) = (10, 53)$, it decreases to only 2. Again, for a fixed value of a_τ, the number of detected edges remains

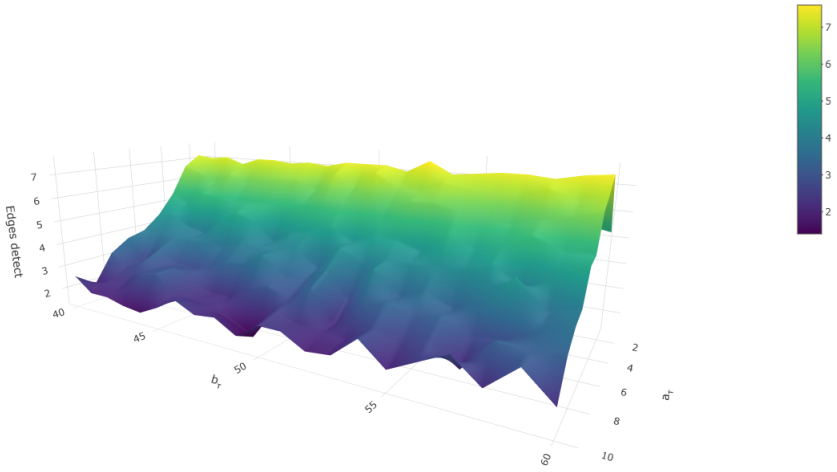


Figure 2.4: Average number of edges estimated by Bayes-LPGM across 10 simulated datasets under different hyperparameter settings, for a random graph with $p = 10$ and a sample size of 100.

largely unaffected by changes in b_τ.

It is also worth emphasizing that, under the same hyperparameter settings, the Chain-LPGM algorithm consistently demonstrates superior edge detection ability compared to Bayes-LPGM. For instance, with $(a_\tau, b_\tau) = (2, 53)$, the average number of edges detected by Chain-LPGM is 9.4, whereas Bayes-LPGM identifies only 7.6.

2.5 Real data analysis

In this section, we reanalyze a subset of the data from the study by Gadye et al. (2017), focusing on the olfactory sensory neuron lineage. This lineage originates from horizontal basal stem cells (HBCs) and progresses through several intermediate stages to produce mature olfactory sensory neurons. Wild-type HBCs were isolated using fluorescence-activated cell sorting (FACS), and their transcriptomic profiles were obtained through single-cell RNA sequencing (scRNA-seq). Our algorithm was employed to reconstruct the gene

interaction network with the aim of identifying central genes within this network. The identification of these hub genes is expected to reveal key transcription factors that regulate the expression of numerous downstream genes and could be prioritized for further experimental investigation. More specifically, we anticipate that several well-known transcription factors involved in this developmental process will emerge as hubs within the inferred network structure.

The gene expression data used in this study were obtained via high-throughput sequencing and downloaded from `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE99251`. The raw dataset consisted of transcriptomic profiles from 542 cells. We identified 850 transcription factor genes for analysis using a curated reference list (`https://github.com/diyadas/HBC-regen/tree/master/ref`).

To ensure robust analysis, we selected the top 20% of genes based on mean expression levels. Data preprocessing followed established methodologies (Allen and Liu, 2013), including:

- normalization by 75% quantile matching;
- selection of the top 50% most variable microRNAs;
- power transformation of the data using X^α ($\alpha \in [0, 1]$).

The optimal transformation parameter $\alpha = 0.219$ was determined by minimization of the Kolmogorov-Smirnov statistic (Li et al., 2012). Following quality control measures to remove genes with low mean expression and limited variability across samples, the final processed dataset contained expression profiles for 542 cells ($n = 542$) and 85 genes ($p = 85$).

The structure of the gene network was estimated by applying Chain-LPGM and Bayes-LPGM to the normalized dataset. The resulting reconstructions are visualized in Figures 2.5 and 2.6, respectively. Based on the simulation results for the case $p = 100$, we selected the hyperparameter for optimal performance as $(a_\tau, b_\tau) = (5, 25)$ to apply the algorithm to the real dataset. For completeness, we also report the results obtained under the hyperparameter setting $(a_\tau, b_\tau) = (3, 50)$ in the Appendix A.2. By ap-

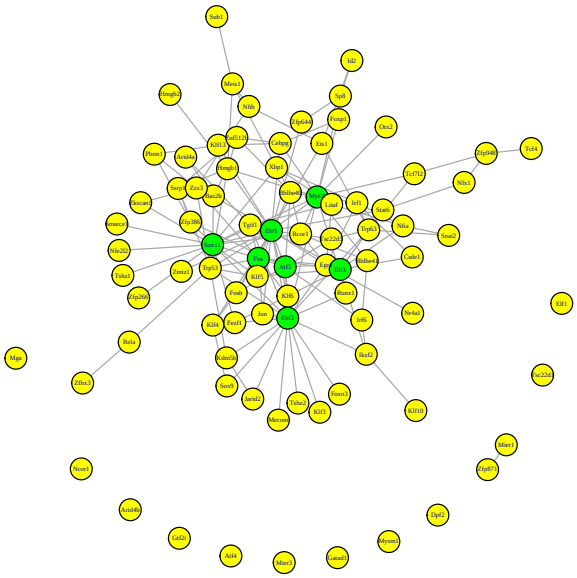


Figure 2.5: Olfactory Neuron gene network estimated by the algorithm using Chain-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$ (hub nodes coloured green).

plying Bayes-LPGM, we obtained a notably sparse graph containing only 18 edges. In contrast, the network reconstructed using Chain-LPGM shows greater biological relevance: we identified seven hub nodes defined as nodes connected by at least 10 edges namely *Sox11, Ebf1, Ebf2, Elf3, Atf3, Fos,* and *Myt1l*. It is noteworthy that *Sox11* has been highlighted for its essential role in neurogenesis (Ninkovic et al., 2013). Moreover, *Ebf1* functions as a transcription factor implicated in the later stages of neuronal lineage commitment, particularly during terminal neuronal maturation (Wang et al., 1997; Garel et al., 1997). Likewise, *Ebf2* contributes to early cortical neurogenesis and regulates the development of Cajal-Retzius neurons in the cerebral cortex (Chuang et al., 2012).

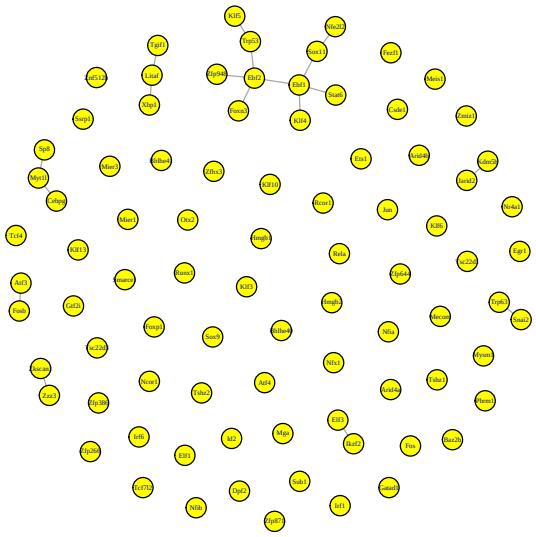


Figure 2.6: Olfactory Neuron gene network estimated by the algorithm using Bayes-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$.

For Bayes-LPGM, we also considered a modified version, referred to as the ORBayes-LPGM, which adopts a different rule for determining the existence of an edge between two nodes s and t, as proposed by Allen and Liu (2013). Specifically, we defined an edge between s and t if the posterior inclusion probability satisfies $\pi(\gamma_{st} = 1 \mid \mathbf{x}) \geq 0.5$ OR $\pi(\gamma_{ts} = 1 \mid \mathbf{x}) \geq 0.5$, rather than requiring both $\pi(\gamma_{st} = 1 \mid \mathbf{x}) \geq 0.5$ AND $\pi(\gamma_{ts} = 1 \mid \mathbf{x}) \geq 0.5$ as in the original Bayes-LPGM. The result obtained by applying the modified algorithm is visualized in Figure 2.7. We identified four hub nodes: *Myt1l, Sox11, Ebf1,* and *Ebf2*. This result is biologically more plausible compared to the network inferred by the original Bayes-LPGM.

One reason why the modified algorithm performs better than Bayes-LPGM is that the variable selection procedure can have difficulty identifying

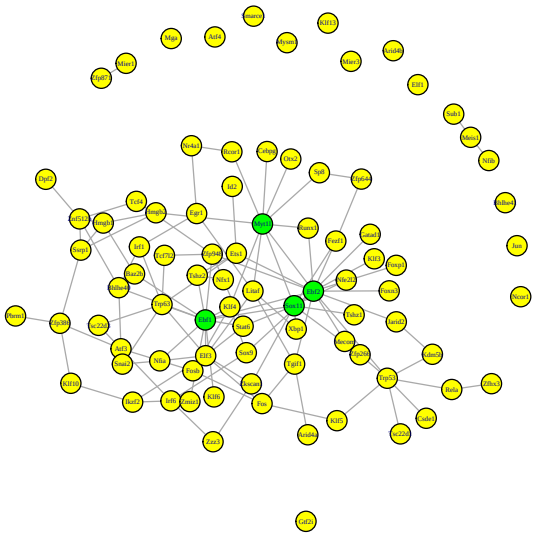


Figure 2.7: Olfactory Neuron gene network estimated by the algorithm using ORBayes-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$ (hub nodes coloured green).

parameters θ_{st} that are nonzero but very close to zero, as we discuss in Appendix A.3. Consequently, the modified algorithm helps mitigate cases where there is indeed an edge between two nodes s and t meaning that both θ_{st} and θ_{ts} are non-zero but one of them is very close to zero and cannot be reliably detected by the variable selection process.

We also speculated that the posterior inclusion probability could be affected by the number of candidate predictor variables considered. To test this hypothesis, we focused on the four hub nodes identified by the modified Bayes-LPGM algorithm, namely *Sox11, Myt1l, Ebf1,* and *Ebf2*, as well as a non-hub node, *Trp63*. For each of these nodes, we then separately performed the stochastic search variable selection of the line 4 in Algorithm 1 using as

predictors:

i) all other genes (full model);

ii) a subset of genes, denoted as $\tilde{N}(s)$ chosen to be meaningfull for the response, statistically or biologically (reduced model).

The results in Table 2.8 compare the posterior inclusion probabilities of the neighboring variables under two settings.

The results show that when we restrict the set of predictor variables, the posterior inclusion probabilities tend to increase in most cases, except for a few instances where they remain unchanged or decrease slightly. Notably, some genes that were not significant in the full model become significant in the reduced models, as highlighted in bold in the table. For example, for the gene *Sox11*, the posterior inclusion probability of *Xbp1* increases from 0.035 to 0.964, and that of *Zfp266* increases from 0.022 to 0.98. A similar pattern is observed for the non-hub gene *Trp63*, where the posterior inclusion probability of *Irf1* increases from 0.047 to 0.978, and that of *Elf3* increases from 0.16 to 0.983.

These findings indicate that the number of predictors included in the regression models significantly affects the algorithm's ability to identify key variables. By concentrating on a narrower set of candidate predictors, reduced models can reveal important regressors that might be overlooked in the more complex, high-dimensional full models. This insight could partly account for the superior performance of Chain-LPGM relative to Bayes-LPGM, since Chain-LPGM leverages multiple reduced models.

When comparing our results with those reported in Hue and Chiogna (2021), where a similar analysis was conducted on the same dataset using the PC-LPGM algorithm, they identified four hub nodes: *Sox11, Ebf1, Elf3,* and *Trp63*. Among these, two genes—*Sox11* and *Ebf1*—are consistent with the hubs identified by our ORBayes-LPGM, and three nodes—*Sox11, Ebf1,* and *Elf3*—overlap with those identified by our Chain-LPGM. In this case study, the true underlying gene network is unknown, and there is also limited understanding of the underlying biological processes. As a result, it is

challenging to interpret the differences in the results produced by different algorithms. Taking into account the broader comparison presented in Hue and Chiogna (2021), we can nonetheless reinforce the notion that methods which appropriately account for the discrete nature of the data tend to extract more coherent information from the data.

2.6 Conclusion

In this chapter, we presented a Bayesian approach to learning graph structures from count data. We proposed a novel model specification, referred to as Chain-LPGM, in which we assumed that the distribution of node s, conditional on nodes 1 through $s-1$, followed a Poisson distribution. We showed that, under certain conditions, the joint distribution implied by this specification was Markov with respect to the underlying graph. In addition, we considered the classical specification of local Poisson graphical models, referred to as Bayes-LPGM, where the conditional distribution of each node given all other nodes was assumed to follow a Poisson distribution. For this specification, the general Bayesian framework proposed by Bissiri et al. (2016) enabled the use of a pseudo-likelihood in place of the full likelihood in the Bayesian inference, thereby overcoming both the constraint requiring non-positive parameter dependencies and the computational intractability of the full likelihood.

The results of our simulation studies demonstrated the effectiveness of Bayesian methods in recovering the underlying graph structure. They also underscored the importance of carefully selecting prior hyperparameter for the overall performance of the algorithms. In particular, Chain-LPGM significantly improved computational efficiency compared to Bayes-LPGM while maintaining high accuracy. This algorithm also showed remarkable performance on hub graphs, which presented particular challenges for structure learning due to their highly concentrated connectivity, often leading to reduced sensitivity for many methods. As expected, the frequentist algorithm

PC-LPGM consistently outperformed others in terms of runtime across all scenarios; however, its performance decreased for smaller sample sizes, although it improved as the sample size increased.

The application of our algorithms to reconstruct the gene regulatory network from a real dataset of the olfactory sensory neuron lineage enabled us to identify key transcription factors that played central roles in neuronal differentiation. We also discussed several possible reasons for false negative edges in Bayes-LPGM, which could be partially mitigated by using the modified algorithm or by restricting the number of candidate predictors in the regression models. This helped explain why Chain-LPGM performed particularly well in this context: by leveraging a sequential modeling approach and focusing on more compact regression models, it enhanced the ability to detect biologically meaningful connections.

Gene s	$\tilde{N}(s)$	$\pi(\gamma_{st}\|\mathbf{x})$	$\pi(\gamma_{st}\|\mathbf{x}_{\tilde{N}(s)})$
Sox11	Ebf1*	0.835	0.999
	*Ebf2**	0.034	0.082
	Xbp1*	0.035	0.964
	*Tshz1**	0.031	0.392
	Myt1l*	0.069	0.571
	Zfp266*	0.022	0.98
	Nfe212*	0.634	0.999
	*Zkscan1**	0.001	0.376
	*Fezf1**	0.015	0.108
	Stat6*	0.007	0.689
Myt1l	Sox11*	1	1
	Ebf2*	0.793	1
	Cebpg*	0.999	1
	Rcor1	0.037	0.032
	Sp8*	0.967	0.999
	Klf4	0.999	1
	Litaf	0.402	0.585
	Hmgb2	0.783	1
	Otx2	0.002	0.039
	Runx1	0.013	0.031
Ebf1	Sox11*	0.993	1
	Baz2b*	0.504	0.969
	Klf6*	0.982	0.999
	Ebf2*	1	1
	Bhlhe40*	0.158	0.660
	Nfia	0.889	0.997
	*Elf3**	0.255	0.295
	Zmiz1*	0.109	0.985
	Klf4*	0.938	1
	*Tshz2**	0.003	0.055
	Stat6	0.681	0.986
	Zfp948	0.071	0.633
Ebf2	Sox11*	0.959	1
	Tgif1*	0.473	0.85
	Trp53*	0.919	1
	Ebf1*	0.997	1
	Foxp1	0.901	0.999
	Klf3*	0.0434	0.793
	Foxn3*	0.662	1
	*Myt1l**	0.122	0.024
	Gatad1	0.587	0.956
	Jarid2*	0.208	0.975
	*Sox9**	0.016	0.039
	Mecom	0.005	0.028
	*Ets1**	0.025	0.046
	*Fezf1**	0.013	0.026
	*Runx1**	0.01	0.024
	Zfp948	0.537	0.994
Trp63	Ebf1*	0.987	1
	Irf1*	0.047	0.978
	Elf3*	0.16	0.983
	Tsc22d3*	0.034	0.771
	Tcf712*	0.952	0.999
	Ets1*	0.013	0.599
	Snai2*	0.965	0.999

Table 2.8: Posterior inclusion probabilities of of regressors in full (third column) and reduced (fourth column) local regression models for nodes Sox11, Myt1l, Trp63, Ebf1 and Ebf2. Regressors that are not selected in the full model but are selected in the reduced model are marked in bold; regressors that are not selected in both the full and reduced model are marked in italic, regressors that are selected by Chain-LPGM are marked *.

Chapter 3

Mean field variational Bayes

As discussed in the previous chapter, a major limitation of using MCMC methods for making inference about the posterior distribution of parameters is their high computational cost. In this chapter, we explore a deterministic approximation approach that relies on optimization rather than sampling, which can substantially reduce the computational burden. This method is known as *variational approximation*.

3.1 Variational inference

For convenience, we denote the observed data by $\mathbf{x}$ and use $\boldsymbol{\xi} \in \boldsymbol{\Xi}$ to represent the vector containing all the model's parameters. Variational inference (VI) is based on the principle of density transformation. The approach involves approximating the posterior density $p(\boldsymbol{\xi} \mid \mathbf{x})$ with another distribution, called variational distribution, here denoted by $\mathfrak{q}(\boldsymbol{\xi})$ that is more tractable and easier to work with. Let $\mathcal{Q}$ be a family of densities. The goal of variational inference is to find a distribution $\mathfrak{q}^\star(\boldsymbol{\xi})$ that minimizes the divergence measure $\mathcal{D}$ between the variational distribution $\mathfrak{q}(\boldsymbol{\xi})$ and the true posterior distribution $p(\boldsymbol{\xi} \mid \mathbf{x})$. It is worth noting that $\mathcal{D}$ does not need to satisfy symmetry.

It is straightforward to observe that different choices of the pair $(\mathcal{Q}, \mathcal{D})$

lead to different approximation schemes. A commonly used divergence measure is the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951).

$$KL(\mathfrak{q}(\boldsymbol{\xi}) \,||\, p(\boldsymbol{\xi} \mid \mathbf{x})) = \int_{\boldsymbol{\Xi}} \mathfrak{q}(\boldsymbol{\xi}) \log \left\{ \frac{\mathfrak{q}(\boldsymbol{\xi})}{p(\boldsymbol{\xi} \mid \mathbf{x})} \right\} \mathrm{d}\boldsymbol{\xi},$$

leading to

$$\mathfrak{q}^{\star}(\boldsymbol{\xi}) = \operatorname{argmin}_{\mathfrak{q}(\boldsymbol{\xi}) \in \mathcal{Q}} KL(\mathfrak{q}(\boldsymbol{\xi}) \,||\, p(\boldsymbol{\xi} \mid \mathbf{x})). \tag{3.1}$$

This setting is referred to as Variational Bayes (VB). The KL divergence possesses the following key properties:

- **non-negativity**: for any two probability distributions $p(\cdot)$ and $\mathfrak{q}(\cdot)$, the KL divergence satisfies $KL(p \,\|\, \mathfrak{q}) \geq 0$.
- **asymmetry**: the KL divergence is not symmetric, i.e., $KL(p \,\|\, \mathfrak{q}) \neq KL(\mathfrak{q} \,\|\, p)$.

One issue that renders the minimization problem in (3.1) intractable is that we do not have access to the true posterior distribution $p(\boldsymbol{\xi} \mid \mathbf{x})$. To address this, we reformulate the problem into an equivalent optimization task by leveraging the following transformation. Let $p(\mathbf{x})$ denote the marginal likelihood, i.e., the likelihood function integrated over the parameter space. Consider

$$\begin{aligned}
\log p(\mathbf{x}) &= \log p(\mathbf{x}) \int \mathfrak{q}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \\
&= \int \mathfrak{q}(\boldsymbol{\xi}) \log \frac{p(\boldsymbol{\xi}, \mathbf{x}) / \mathfrak{q}(\boldsymbol{\xi})}{p(\boldsymbol{\xi} \mid \mathbf{x}) / \mathfrak{q}(\boldsymbol{\xi})} \mathrm{d}\boldsymbol{\xi} \\
&= \int \mathfrak{q}(\boldsymbol{\xi}) \log \frac{p(\boldsymbol{\xi}, \mathbf{x})}{\mathfrak{q}(\boldsymbol{\xi})} \mathrm{d}\boldsymbol{\xi} \\
&+ \int \mathfrak{q}(\boldsymbol{\xi}) \log \frac{\mathfrak{q}(\boldsymbol{\xi})}{p(\boldsymbol{\xi} \mid \mathbf{x})} \mathrm{d}\boldsymbol{\xi} \\
&= \int \mathfrak{q}(\boldsymbol{\xi}) \log \frac{p(\boldsymbol{\xi}, \mathbf{x})}{\mathfrak{q}(\boldsymbol{\xi})} \mathrm{d}\boldsymbol{\xi} + KL(\mathfrak{q}(\boldsymbol{\xi}) \,||\, p(\boldsymbol{\xi} \mid \mathbf{x})) \\
&\geq \int \mathfrak{q}(\boldsymbol{\xi}) \log \frac{p(\boldsymbol{\xi}, \mathbf{x})}{\mathfrak{q}(\boldsymbol{\xi})} \mathrm{d}\boldsymbol{\xi} = \log \underline{p}(\mathbf{x}; \mathfrak{q}),
\end{aligned} \tag{3.2}$$

where $p(\boldsymbol{\xi}, \mathbf{x})$ denote the joint distribution of $\boldsymbol{\xi}$ and $\mathbf{X}$. The logarithm of the marginal likelihood $p(\mathbf{x})$ can be decomposed into two components:

- the lower bound on the log-marginal likelihood $\log \underline{p}(\mathbf{x}; \mathfrak{q})$, also called the evidence lower bound (ELBO) in the machine learning literature;
- the Kullback-Leibler divergence between the variational density $\mathfrak{q}(\boldsymbol{\xi})$ and the true posterior distribution $p(\boldsymbol{\xi} \mid \mathbf{x})$.

From Equation (3.2), it is easy to see that minimizing $KL(\mathfrak{q}(\boldsymbol{\xi}) \mid\mid p(\boldsymbol{\xi} \mid \mathbf{x}))$ is equivalent to maximizing the lower bound $\underline{p}(\mathbf{x}, \mathfrak{q})$. Notably, all the quantities required to solve this new optimization problem are known.

In summary, variational inference (Ormerod and Wand, 2010) approximates the posterior density $p(\boldsymbol{\xi}|\mathbf{x})$ by a variational density $\mathfrak{q}(\boldsymbol{\xi})$, for which the lower bound $\underline{p}(\mathbf{x}; \mathfrak{q})$ is more tractable than the marginal likelihood $p(\mathbf{x})$. This is achieved by imposing certain manageable assumptions on the variational family $\mathcal{Q}$. Two common assumptions are:

1. **Mean field (MF)**: $\mathfrak{q}(\boldsymbol{\xi}) = \prod_{i=1}^{N} \mathfrak{q}_i(\boldsymbol{\xi}_i)$, for some partition $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N\}$ of $\boldsymbol{\xi}$;
2. $\mathfrak{q}(\boldsymbol{\xi})$ belongs to a parametric family of density functions.

Depending on the Bayesian model under consideration, these assumptions can have varying impacts on the inference process.

3.1.1 Non-parametric mean field variational Bayes

In this case, we only assume that the family of distributions $\mathcal{Q}$ satisfies the mean-field (MF) assumption. Under this assumption, the ELBO can be expressed as follows.

$$\begin{aligned}
\log p(\boldsymbol{\xi}, \mathfrak{q}) &= \int \prod_{i=1}^{N} \mathfrak{q}_i(\boldsymbol{\xi}_i) \{\log p(\boldsymbol{\xi}; \mathbf{x}) - \sum_{i=1}^{N} \log \mathfrak{q}_i(\boldsymbol{\xi}_i)\} \mathrm{d}\boldsymbol{\xi}_1 \ldots \mathrm{d}\boldsymbol{\xi}_N \\
&= \int \mathfrak{q}_1(\boldsymbol{\xi}_1) \left\{ \int (\log p(\boldsymbol{\xi}; \mathbf{x}) \mathfrak{q}_2(\boldsymbol{\xi}_2) \ldots \mathfrak{q}_N(\boldsymbol{\xi}_N)) \mathrm{d}\boldsymbol{\xi}_2 \ldots \mathrm{d}\boldsymbol{\xi}_N \right\} \mathrm{d}\boldsymbol{\xi}_1 \\
&\quad - \int \mathfrak{q}_1(\boldsymbol{\xi}_1) \log \mathfrak{q}_1(\boldsymbol{\xi}_1) \mathrm{d}\boldsymbol{\xi}_1 - \sum_{i=2}^{N} \mathrm{E}_{\mathfrak{q}}[\log \mathfrak{q}_i(\boldsymbol{\xi}_i)],
\end{aligned}$$

where $\mathrm{E}_{\mathfrak{q}}(\cdot)$ denotes the expectation of the argument w.r.t the distribution of $\mathfrak{q}$. We define a new joint density $\tilde{p}(\boldsymbol{\xi}_1, \mathbf{x})$

$$\tilde{p}(\boldsymbol{\xi}_1, \mathbf{x}) = \frac{\exp \int \log p(\boldsymbol{\xi}; \mathbf{x}) \mathfrak{q}_2(\boldsymbol{\xi}_2) \dots \mathfrak{q}_N(\boldsymbol{\xi}_N)) \mathrm{d}\boldsymbol{\xi}_2 \dots \mathrm{d}\boldsymbol{\xi}_N}{\int \int \{\exp \int \log p(\boldsymbol{\xi}; \mathbf{x}) \mathfrak{q}_2(\boldsymbol{\xi}_2) \dots \mathfrak{q}_N(\boldsymbol{\xi}_N)) \mathrm{d}\boldsymbol{\xi}_2 \dots \mathrm{d}\boldsymbol{\xi}_N\} \mathrm{d}\boldsymbol{\xi}_1 \mathrm{d}\mathbf{x}},$$

and the lower bound is equal to

$$\log \underline{p}(\boldsymbol{\xi}, \mathfrak{q}) = \int \mathfrak{q}_1(\boldsymbol{\xi}_1) \log \left\{ \frac{\tilde{p}(\boldsymbol{\xi}_1; \mathbf{x})}{\mathfrak{q}_1(\boldsymbol{\xi}_1)} \right\} \mathrm{d}\boldsymbol{\xi}_1 + \text{term not depend to } \mathfrak{q}_1(\boldsymbol{\xi}_1).$$

Then, we have

$$\begin{aligned} \mathfrak{q}_1^\star(\boldsymbol{\xi}_1) &= \mathrm{argmax}_{\mathfrak{q}_1(\boldsymbol{\xi}_1) \in \mathcal{Q}} \log \underline{p}(\mathbf{x}; \mathfrak{q}) = \tilde{p}(\boldsymbol{\xi}_1 \mid \mathbf{x}) = \frac{\tilde{p}(\boldsymbol{\xi}_1; \mathbf{x})}{\int \tilde{p}(\boldsymbol{\xi}_1; \mathbf{x}) \mathrm{d}\boldsymbol{\xi}_1} \\ &\propto \exp \left\{ \int \log p(\boldsymbol{\xi}; \mathbf{x}) \mathfrak{q}_2(\boldsymbol{\xi}_2) \dots \mathfrak{q}_N(\boldsymbol{\xi}_N)) \mathrm{d}\boldsymbol{\xi}_2 \dots \mathrm{d}\boldsymbol{\xi}_N \right\} \\ &= \exp\{\mathrm{E}_{-\boldsymbol{\xi}_1}[\log p(\boldsymbol{\xi}; \mathbf{x})]\}, \end{aligned} \tag{3.3}$$

where $\mathrm{E}_{-\boldsymbol{\xi}_1}$ is the expected value over $\prod_{i \neq 1} \mathfrak{q}_i(\boldsymbol{\xi}_i)$. By systematically applying the same procedure to $\mathfrak{q}_2(\boldsymbol{\xi}_2), \dots, \mathfrak{q}_N(\boldsymbol{\xi}_N)$, we derive a general formulation analogous to that presented in equation (3.3).

$$\begin{aligned} \mathfrak{q}_i^\star(\boldsymbol{\xi}_i) &\propto \exp\{\mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}; \mathbf{x})]\} \\ &\propto \exp\{\mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})]\}. \end{aligned} \tag{3.4}$$

The resulting formulation leads to an iterative algorithm for finding the optimal variational densities $\mathfrak{q}_i^\star(\boldsymbol{\xi}_i)$, as presented in Algorithm 2. This procedure is commonly referred to as Coordinate Ascent Variational Inference (CAVI) in the machine learning literature (Blei et al., 2017).

Equation (3.4) illustrates the connection between the nonparametric variational approximation and the Gibbs sampling, which can be viewed as a special case of Markov chain Monte Carlo (MCMC) methods. While Gibbs sampling requires computing the full conditional distribution $p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})$, nonparametric mean field variational Bayes (MFVB) instead involves evaluating the expectation of the logarithm of this full conditional distribution. The way in which $\boldsymbol{\xi}$ is partitioned into components $\{\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_N\}$ can affect

Algorithm 2: CAVI for non-parametric MFVB.

Initialize: $\mathfrak{q}_1^\star(\boldsymbol{\xi}_1), \ldots, \mathfrak{q}_N^\star(\boldsymbol{\xi}_N)$
while *increase in* $\log \underline{p}(\mathbf{x}; \mathfrak{q})$ *is greater than* ϵ **do**
  for $i = 1, \ldots, N$ **do**
    $\mathfrak{q}_i^\star(\boldsymbol{\xi}_i) \leftarrow \dfrac{\exp\{\mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})]\}}{\int \mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})]\}\mathrm{d}\boldsymbol{\xi}_i}$
  end for
  compute $\log \underline{p}(\mathbf{x}; \mathfrak{q})$
end while

the results of variational inference. Therefore, it is important to strike a balance between the accuracy of the approximation and the computational tractability (Wand et al., 2011).

3.1.2 Semi-parametric mean field variational Bayes

Within this model, in addition to the analytically tractable densities, certain density functions arising in the factorization are assumed to belong to a family of parametric distributions selected for their computational convenience (Rohde and Wand, 2016). For convenience, in addition to the observed data $\mathbf{x}$ and the parameter vector $\boldsymbol{\xi}$, we assume the presence of an additional parameter vector $\boldsymbol{\phi} \in \boldsymbol{\Phi} \subset \mathbb{R}^d$. Semi-parametric mean field variational Bayes (MFVB) is carried out in two steps. In the first step, we posit a mean field factorization of the variational density $\mathfrak{q}(\boldsymbol{\xi}, \boldsymbol{\phi})$, for example,

$$\mathfrak{q}(\boldsymbol{\xi}, \boldsymbol{\phi}) = \mathfrak{q}(\boldsymbol{\phi}) \prod_{i=1}^{N} \mathfrak{q}_i(\boldsymbol{\xi}_i). \tag{3.5}$$

The second step focuses on approximating the variational density $\mathfrak{q}(\boldsymbol{\phi})$ by assuming that it belongs to a parametric family of densities characterized by a parameter vector $\boldsymbol{\psi} \in \boldsymbol{\Psi} \subset \mathbb{R}^z$. As a result, the factorization of the

variational density becomes:

$$\mathfrak{q}(\boldsymbol{\xi}, \boldsymbol{\phi}) = \mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi}) \prod_{i=1}^{N} \mathfrak{q}_i(\boldsymbol{\xi}_i).$$

Consider the joint distribution $p(\boldsymbol{\xi}, \boldsymbol{\phi}; \mathbf{x})$ with M factors and the semi parametric mean field assumption (3.5). Under this setting, the ELBO can be rewritten as follows.

$$\begin{aligned}
\log \underline{p}(\mathbf{x}; \mathfrak{q}) &= \int \mathfrak{q}(\boldsymbol{\xi}, \boldsymbol{\phi}) \Bigg\{ \log p(\boldsymbol{\xi}, \boldsymbol{\phi}; \mathbf{x}) - \sum_{i=1}^{N} \log \mathfrak{q}_i(\boldsymbol{\xi}_i) \\
&\quad - \log \mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi}) \Bigg\} \mathrm{d}\boldsymbol{\xi}_1 \ldots \mathrm{d}\boldsymbol{\xi}_N \mathrm{d}\boldsymbol{\phi} \\
&= \sum_{j=1}^{M} \mathrm{E}_{\mathfrak{q}}(\log p_j) + \sum_{i=1}^{N} \mathrm{E}_{\mathfrak{q}} \left[- \log \mathfrak{q}_i(\boldsymbol{\xi}_i) \right] + \mathrm{E}_{\mathfrak{q}} \left[- \log \mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi}) \right] \\
&= \sum_{j=1}^{M} \mathrm{E}_{\mathfrak{q}}(\log p_j) + \sum_{i=1}^{N} \mathrm{H}\{\mathfrak{q}_i(\boldsymbol{\xi}_i)\} + \mathrm{H}\{\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})\},
\end{aligned} \tag{3.6}$$

where H is denoted for entropy of a random vector $\mathbf{X}$ with density function $p(\mathbf{x})$, i.e,

$$\mathrm{H}\{p(\mathbf{x})\} = \mathrm{E}_p[-\log p(\mathbf{x})].$$

To maximize the lower bound in (3.6) with respect to $\mathfrak{q}_1(\boldsymbol{\xi}_1), \ldots, \mathfrak{q}_N(\boldsymbol{\xi}_N)$ and $\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})$, the optimal variational densities $\mathfrak{q}_1^\star(\boldsymbol{\xi}_1), \ldots, \mathfrak{q}_N^\star(\boldsymbol{\xi}_N)$ are given by the general form in (3.4). Meanwhile, the optimal form of $\mathfrak{q}^\star(\boldsymbol{\phi}, \boldsymbol{\psi})$ is obtained by maximizing the $\boldsymbol{\phi}$-localized component of the lower bound $\log p(\mathbf{x}; \mathfrak{q})$, denoted by $\log p(\mathbf{x}; \mathfrak{q})^{[\boldsymbol{\phi}]}$, with respect to $\boldsymbol{\psi} \in \boldsymbol{\Psi}$, as presented in Rohde and Wand (2016). This quantity is defined as,

$$\begin{aligned}
\log \underline{p}(\mathbf{x}; \mathfrak{q})^{[\boldsymbol{\phi}]} &= \mathrm{H}\{\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})\} + \mathrm{NonEntropy}\{\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi}\} \\
&= \mathrm{H}\{\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})\} + \overline{\mathrm{H}}\{\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})\} \\
&= \mathrm{H}(\mathfrak{q}(\boldsymbol{\phi}, \boldsymbol{\psi})) + \sum_{i \in \mathrm{neighbors}(\boldsymbol{\phi})} \mathrm{E}_{\mathfrak{q}}[\log p_i],
\end{aligned} \tag{3.7}$$

where

$$\text{neighbors}(\boldsymbol{\phi}) = \{1 \leq i \leq M : \ p_i \text{ involves } \boldsymbol{\phi}\}.$$

The iterative procedure that extends the coordinate ascent algorithm (Algorithm 2) to the case of semi-parametric mean field approximation is presented in Algorithm 3.

Algorithm 3: CAVI for semi-parametric MFVB.

Initialize: $\mathfrak{q}_1^\star(\boldsymbol{\xi}_1), \ldots, \mathfrak{q}_N^\star(\boldsymbol{\xi}_N), \boldsymbol{\psi}^\star$
while *increase in* $\log \underline{p}(\mathbf{x}, \mathfrak{q})$ *is greater than* ϵ **do**
  for $i = 1, \ldots, N$ **do**
    $\mathfrak{q}_i^\star(\boldsymbol{\xi}_i) \leftarrow \dfrac{\exp\{\mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})]\}}{\int \mathrm{E}_{-\boldsymbol{\xi}_i}[\log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{-i}; \mathbf{x})]\}\mathrm{d}\boldsymbol{\xi}_i}$
  end for
  while *convergence not reached* **do**
    $\boldsymbol{\psi}^\star \leftarrow \text{argmax}_{\boldsymbol{\psi}} \log \underline{p}(\mathbf{x}; \mathfrak{q})^{[\boldsymbol{\phi}]}$
  end while
  compute $\log \underline{p}(\mathbf{x}, \mathfrak{q})$
end while

3.2 Semi-parametric mean field variational Bayes for structure learning

As discussed in the previous chapter, the main idea underlying the proposed algorithms is to perform stochastic search variable selection for regression models. In this section, we present a MFVB approach to the variable selection problem in regression models, as an alternative to MCMC sampling. Specifically, we apply the semi-parametric MFVB method to the problem of variable selection in Poisson regression models. The goal is to reduce computational time while maintaining a high level of accuracy.

3.2.1 Models

We begin this section by presenting a general semiparametric MFVB method for Bayesian variable selection in Poisson regression models. Later on we will apply this approach to our structure learning problem. Let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ be a vector of responses containing count data and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^\top$ a vector of regression coefficients, and consider the following Poisson response regression model with a spike-and-slab prior on each regression coefficient, for $i = 1, \ldots, n$ and $s = 1, \ldots, p$:

$$
\begin{aligned}
& y_i \,|\, \boldsymbol{\theta} \sim \text{Poisson}(\exp(\mathbf{x}_i^\top \boldsymbol{\theta})), \\
& \theta_s \,|\, \gamma_s, \tau_s^2, v_1 \overset{\text{prior}}{\sim} N\left(0, \tau_s^2 \left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}\right)^{-1}\right), \\
& \gamma_s \,|\, \omega \overset{\text{prior}}{\sim} \text{Bernoulli}(\omega), \\
& \tau_s^2 \overset{\text{prior}}{\sim} \Gamma^{-1}(a_\tau, b_\tau), \\
& v_1 \overset{\text{prior}}{\sim} \Gamma^{-1}(a_{v_1}, b_{v_1}), \\
& \omega \overset{\text{prior}}{\sim} \text{Beta}(a_\omega, b_\omega).
\end{aligned}
\tag{3.8}
$$

In the model above, v_0 is a small positive constant value, and a_τ, b_τ, a_{v_1}, b_{v_1}, a_ω and b_ω are user-specified hyperparameter. Here, we introduce a slight modification to the NMIG prior employed in the previous chapter by treating v_1 as a random variable following an inverse-gamma distribution, rather than fixing $v_1 = 1$ as in the original NMIG prior. This modification is inspired by the work of Ročková and George (2014) to make the model more flexible while also offering computational advantages.

To carry out the computations for MFVB, we need to evaluate the full joint log-likelihood induced by the model in equation (3.8), which corresponds to the logarithm of the product of the following density functions.

$$
p(y_i \,|\, \boldsymbol{\theta}; \mathbf{x}_i) = \frac{\exp(y_i \mathbf{x}_i^\top \boldsymbol{\theta}) \exp\{-\exp(\mathbf{x}_i^\top \boldsymbol{\theta})\}}{y_i!}, \quad i = 1, \ldots, n;
$$

$$
p(\theta_s \,|\, \gamma_s, \tau_s^2, v_1; v_0) = \sqrt{\frac{\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}}{2\pi\tau_s^2}} \exp\left(-\frac{\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}}{2\tau_s^2}\theta_s^2\right), \quad s = 1, \ldots, p;
$$

$$
\begin{aligned}
p(\gamma_s \,|\, \omega) &= \omega\gamma_s + (1-\omega)(1-\gamma_s), \quad s = 1, \ldots, p; \\
p(\tau_s^2 \,|\, a_\tau, b_\tau) &= \frac{b_{\tau_s}^{a_\tau}}{\Gamma(a_\tau)} (\tau_s^2)^{-(a_\tau+1)} \exp\left(-\frac{b_\tau}{\tau_s^2}\right), \quad s = 1, \ldots, p; \\
p(v_1 \,|\, a_{v_1}, b_{v_1}) &= \frac{b_{v_1}^{a_{v_1}}}{\Gamma(a_{v_1})} v_1^{-(a_{v_1}+1)} \exp\left(-\frac{b_{v_1}}{v_1}\right); \\
p(\omega \,|\, a_\omega, b_\omega) &= \frac{\omega^{a_\omega - 1}(1-\omega)^{b_\omega - 1}}{\mathrm{B}(a_\omega, b_\omega)},
\end{aligned}
$$

where $\Gamma(\cdot)$ and $\mathrm{B}(\cdot,\cdot)$ denote the gamma and beta functions, respectively. The joint log-likelihood function is then

$$
\begin{aligned}
&\log p(\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1, \omega; \boldsymbol{y}) \\
&= \log\left(\prod_{i=1}^{n} p(y_i \,|\, \boldsymbol{\theta}) \prod_{s=1}^{p} \left\{p(\theta_s \,|\, \gamma_s, \tau_s^2, v_1) p(\gamma_s \,|\, \omega) p(\tau_s^2)\right\} p(v_1) p(\omega)\right) \\
&= \sum_{i=1}^{n} y_i \mathbf{x}_i^\top \boldsymbol{\theta} - \sum_{i=1}^{n} \exp(\mathbf{x}_i^\top \boldsymbol{\theta}) - \sum_{i=1}^{n} \log(y_i!) \\
&\quad - \frac{p}{2}\log(2\pi) - \frac{1}{2}\sum_{s=1}^{p}\log(\tau_s^2) - \frac{p}{2}\log(v_0) - \frac{1}{2}\log\left(\frac{v_1}{v_0}\right)\sum_{s=1}^{p}\gamma_s \\
&\quad - \frac{1}{2}\sum_{s=1}^{p}\left\{\left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}\right)\frac{\theta_s^2}{\tau_s^2}\right\} + \log\left(\frac{\omega}{1-\omega}\right)\sum_{s=1}^{p}\gamma_s + p\log(1-\omega) \\
&\quad + p(a_\tau \log(b_\tau) - \log\{\Gamma(a_\tau)\}) - (a_\tau + 1)\sum_{s=1}^{p}\log(\tau_s^2) - \sum_{s=1}^{p}\frac{b_\tau}{\tau_s^2} \\
&\quad + a_{v_1}\log(b_{v_1}) - \log\{\Gamma(a_{v_1})\} - (a_{v_1}+1)\log(v_1) - \frac{b_{v_1}}{v_1} \\
&\quad + (a_\omega - 1)\log(\omega) + (b_\omega - 1)\log(1-\omega) - \log\{\mathrm{B}(a_\omega, b_\omega)\},
\end{aligned}
\tag{3.9}
$$

where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$, $\boldsymbol{\tau}^2 = (\tau_1^2, \ldots, \tau_p^2)$, with $\gamma_s \in \{0, 1\}$, we used the relationship

$$
\sum_{s=1}^{p}\log\left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}\right) = -p\log(v_0) - \log\left(\frac{v_1}{v_0}\right)\sum_{s=1}^{p}\gamma_s.
$$

3.2.2 Mean field variational Bayes

We aim to find an approximating density $\mathfrak{q}$ factorized as

$$\mathfrak{q}(\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1, \omega) = \mathfrak{q}(\boldsymbol{\theta}) \prod_{s=1}^{p} \{\mathfrak{q}_s(\gamma_s)\mathfrak{q}_s(\tau_s^2)\}\mathfrak{q}(v_1)\mathfrak{q}(\omega). \tag{3.10}$$

From equation (3.4), the optimal density functions are given by

$$\begin{aligned}
\mathfrak{q}^\star(\boldsymbol{\theta}) &\propto \exp\left\{ \mathrm{E}_{-\boldsymbol{\theta}}[\log p(\boldsymbol{\theta} \mid \text{rest})] \right\}, \\
\mathfrak{q}_s^\star(\gamma_s) &\propto \exp\left\{ \mathrm{E}_{-\gamma_s}[\log p(\gamma_s \mid \text{rest})] \right\}, \\
\mathfrak{q}_s^\star(\tau_s^2) &\propto \exp\left\{ \mathrm{E}_{-\tau_s^2}[\log p(\tau_s^2 \mid \text{rest})] \right\}, \\
\mathfrak{q}^\star(v_1) &\propto \exp\left\{ \mathrm{E}_{-v_1}[\log p(v_1 \mid \text{rest})] \right\}, \\
\mathfrak{q}^\star(\omega) &\propto \exp\left\{ \mathrm{E}_{-\omega}[\log p(\omega \mid \text{rest})] \right\}.
\end{aligned}$$

We can obtain the full conditional densities from the joint log-likelihood function (3.9). These are

$$p(\boldsymbol{\theta} \,|\, \text{rest}) \propto \exp\left[\sum_{i=1}^{n} y_i \mathbf{x}_i^\top \boldsymbol{\theta} - \sum_{i=1}^{n} \exp(\mathbf{x}_i^\top \boldsymbol{\theta}) - \frac{1}{2} \sum_{s=1}^{p} \left\{ \left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0} \right) \frac{\theta_s^2}{\tau_s^2} \right\} \right],$$

$$p(\gamma_s \,|\, \text{rest}) \propto \exp\left[\gamma_s \left\{ \log\left(\frac{\omega}{1-\omega} \right) + \frac{\theta_s^2}{2\tau_s^2} \left(\frac{1}{v_0} - \frac{1}{v_1} \right) - \frac{1}{2} \log\left(\frac{v_1}{v_0} \right) \right\} \right],$$

$$p(\tau_s^2 \,|\, \text{rest}) \propto (\tau_s^2)^{-\left(\frac{1}{2} + a_\tau + 1\right)} \exp\left[-\frac{1}{\tau_s^2} \left\{ \frac{1}{2} \left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0} \right) \theta_s^2 + b_\tau \right\} \right],$$

$$p(v_1 \,|\, \text{rest}) \propto v_1^{-\left(\frac{1}{2}\sum_{s=1}^{p} \gamma_s + a_{v_1} + 1\right)} \exp\left\{ -\frac{1}{v_1} \left(\sum_{s=1}^{p} \frac{\gamma_s \theta_s^2}{2\tau_s^2} + b_{v_1} \right) \right\},$$

$$p(\omega \,|\, \text{rest}) \propto \exp\left\{ \left(a_\omega + \sum_{s=1}^{p} \gamma_s - 1 \right) \log(\omega) + \left(p + b_\omega - \sum_{s=1}^{p} \gamma_s - 1 \right) \log(1-\omega) \right\}.$$

Before explicitly computing the optimal density functions, we introduce some useful notation. Let $\boldsymbol{\xi}$ be a general parameter vector and let $f(\boldsymbol{\xi})$ denote an element-wise function of $\boldsymbol{\xi}$. We define $\boldsymbol{\mu}_{\mathfrak{q}(f(\boldsymbol{\xi}))}$ as the vector of expectations of the elements of $f(\boldsymbol{\xi})$ with respect to the variational density $\mathfrak{q}(\boldsymbol{\xi})$. For example, if $\boldsymbol{\xi}$ is a scalar parameter $\xi = \sigma^2$ and $f(\sigma^2) = 1/\sigma^2$, then the quantity $\mu_{\mathfrak{q}(1/\sigma^2)}$ is given by

$$\mu_{\mathfrak{q}(1/\sigma^2)} = \int_0^\infty \frac{1}{\sigma^2} \mathfrak{q}(\sigma^2)\, d\sigma^2.$$

The corresponding optimal density functions are then provided in the following propositions.

Proposition 3.2.1. *The optimal density $\mathfrak{q}^\star(\boldsymbol{\theta})$ is not a standard form.*

Proof.

$$\mathfrak{q}^\star(\boldsymbol{\theta}) \propto \exp\{\mathrm{E}_{-\boldsymbol{\theta}}[\log p(\boldsymbol{\theta} \mid \text{rest})]\}$$

$$= \exp\left\{ \mathrm{E}_{-\boldsymbol{\theta}} \left[\sum_{i=1}^{n} \{ y_i \mathbf{x}_i^\top \boldsymbol{\theta} - \exp(\mathbf{x}_i^\top \boldsymbol{\theta}) \} - \frac{1}{2} \sum_{s=1}^{p} \left\{ \left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0} \right) \frac{\theta_s^2}{\tau_s^2} \right\} \right] \right\}$$

$$= \exp\left\{ \sum_{i=1}^{n} \{ y_i \mathbf{x}_i^\top \boldsymbol{\theta} - \exp(\mathbf{x}_i^\top \boldsymbol{\theta}) \} - \frac{\theta_s^2}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \left(\mu_{\mathfrak{q}(1/v_1)} \mu_{\mathfrak{q}_s(\gamma_s)} + \frac{\mu_{\mathfrak{q}_s(1-\gamma_s)}}{v_0} \right) \right\}.$$

In this case, the form of the resulting kernel does not match that of any tractable or well-known distribution. □

Proposition 3.2.2. *The optimal density $\mathfrak{q}_s^\star(\gamma_s)$ is Bernoulli($\mu_{\mathfrak{q}_s(\gamma_s)}$), with:*

$$\log\left(\frac{\mu_{\mathfrak{q}_s(\gamma_s)}}{1 - \mu_{\mathfrak{q}_s(\gamma_s)}} \right) = \frac{1}{2} \left\{ \mu_{\mathfrak{q}(\theta_s^2)} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \left(\frac{1}{v_0} - \mu_{\mathfrak{q}(1/v_1)} \right) + \log(v_0) - \mu_{\mathfrak{q}(\log(v_1))} \right\}$$
$$+ \mu_{\mathfrak{q}(\log(\omega))} - \mu_{\mathfrak{q}(\log(1-\omega))}.$$

Proof.

$$\mathfrak{q}_s^\star(\gamma_s) \propto \exp\{\mathrm{E}_{-\gamma_s}[\log p(\gamma_s \mid \text{rest})]\}$$

$$= \exp\left\{ \mathrm{E}_{-\gamma_s}\left[\gamma_s\left\{\log\left(\frac{\omega}{1-\omega}\right) + \frac{\theta_s^2}{2\tau_s^2}\left(\frac{1}{v_0}-\frac{1}{v_1}\right) - \frac{1}{2}\log\left(\frac{v_1}{v_0}\right)\right\}\right]\right\}$$

$$= \exp\left\{\gamma_s\left[\frac{1}{2}\left\{\mu_{\mathfrak{q}(\theta_s^2)}\mu_{\mathfrak{q}_s(1/\tau_s^2)}\left(\frac{1}{v_0} - \mu_{\mathfrak{q}(1/v_1)}\right) + \log(v_0) - \mu_{\mathfrak{q}(\log(v_1))}\right\}\right.\right.$$

$$\left.\left. + \mu_{\mathfrak{q}(\log(\omega))} - \mu_{\mathfrak{q}(\log(1-\omega))}\right]\right\}.$$

This expression represents the kernel of a Bernoulli distribution, where the parameter is given by the formulation in Proposition 3.2.2. □

Proposition 3.2.3. *The optimal density* $\mathfrak{q}_s^\star(\tau_s^2)$ *is* $\Gamma^{-1}(\alpha_{\mathfrak{q}_s(\tau_s^2)}, \beta_{\mathfrak{q}_s(\tau_s^2)})$*, with*

$$\alpha_{\mathfrak{q}_s(\tau_s^2)} = \frac{1}{2} + a_\tau \quad \textit{and} \quad \beta_{\mathfrak{q}_s(\tau_s^2)} = \frac{1}{2}\mu_{\mathfrak{q}(\theta_s^2)}\left\{\mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1-\mu_{\mathfrak{q}_s(\gamma_s)})\right\} + b_\tau.$$

Proof.

$$\mathfrak{q}_s^\star(\tau_s^2) \propto \exp\{\mathrm{E}_{-\tau_s^2}[p(\tau_s^2 \mid \text{rest})]\}$$

$$= \exp\left\{ \mathrm{E}_{-\tau_s^2}\left[\left(\frac{1}{2} + a_\tau + 1\right)\log(\tau_s^2) - \frac{1}{\tau_s^2}\left\{\frac{1}{2}\left(\frac{\gamma_s}{v_1} + \frac{1-\gamma_s}{v_0}\right)\theta_s^2 + b_\tau\right\}\right]\right\}$$

$$= (\tau_s^2)^{-\left(\frac{1}{2}+a_\tau+1\right)} \exp\left\{ -\frac{1}{\tau_s^2}\left[\frac{1}{2}\mu_{\mathfrak{q}(\theta_s^2)}\left\{\mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1-\mu_{\mathfrak{q}_s(\gamma_s)})\right\} + b_\tau\right]\right\}.$$

This expression represents the kernel of a Γ^{-1} distribution, where the parameter is given in Proposition 3.2.3. □

Proposition 3.2.4. *The optimal density* $\mathfrak{q}^\star(v_1)$ *is* $\Gamma^{-1}(\alpha_{\mathfrak{q}(v_1)}, \beta_{\mathfrak{q}(v_1)})$*,with*

$$\alpha_{q(v_1)} = \frac{1}{2}\sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)} + a_{v_1} \quad \textit{and} \quad \beta_{q(v_1)} = \frac{1}{2}\sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(\theta_s^2)}\mu_{\mathfrak{q}_s(1/\tau_s^2)} + b_{v_1}.$$

Proof.

$$\mathfrak{q}^\star(v_1) \propto \exp\{\mathrm{E}_{v_1}[\log p(v_1 \mid \text{rest}]\}$$

$$= \exp\left\{ \mathrm{E}_{-v_1}\left[-\left(\frac{1}{2}\sum_{s=1}^{p}\gamma_s + a_{v_1} + 1\right)\log(v_1) - \frac{1}{v_1}\left(\sum_{s=1}^{p}\frac{\gamma_s\theta_s^2}{2\tau_s^2} + b_{v_1}\right)\right]\right\}$$

$$= (v_1)^{-(\frac{1}{2}\sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)}+a_{v_1}+1)} \exp\left\{ -\frac{1}{v_1}\left\{\frac{1}{2}\sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(\theta_s^2)}\mu_{\mathfrak{q}_s(1/\tau_s^2)} + b_{v_1}\right\}\right\}.$$

This expression represents the kernel of a Γ^{-1} distribution, where the parameter is given in Proposition 3.2.4. □

Proposition 3.2.5. *The optimal density* $\mathfrak{q}^\star(\omega)$ *is Beta* $\big(\alpha_{q(\omega)}, \beta_{q(\omega)}\big)$*, with*

$$\alpha_{q(\omega)} = a_\omega + \sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)} \quad \textit{and} \quad \beta_{q(\omega)} = p + b_\omega - \sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)}.$$

Proof.

$\mathfrak{q}^\star(\omega) \propto \exp\{\mathrm{E}_{-\omega}[\log p(\omega \mid \text{rest})]\}$

$$= \exp\left\{ \mathrm{E}_{-\omega}\left[\left(a_\omega + \sum_{s=1}^{p}\gamma_s - 1\right)\log(\omega) + \left(p + b_\omega - \sum_{s=1}^{p}\gamma_s - 1\right)\log(1-\omega)\right]\right\}$$

$$= \omega^{a_\omega + \sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)} - 1}(1-\omega)^{p + b_\omega - \sum_{s=1}^{p}\mu_{\mathfrak{q}_s(\gamma_s)} - 1}.$$

This expression represents the kernel of a Beta distribution, where the parameter is given in Proposition 3.2.5. □

As a result, we obtain closed-form expressions for all variational densities in equation (3.10), except for $\mathfrak{q}^\star(\boldsymbol{\theta})$, which does not correspond to a known distribution. To address this issue, we employ the semi-parametric MFVB approach, in which we assume that $\mathfrak{q}(\boldsymbol{\theta})$ belongs to a tractable parametric family. As is common practice, we choose the multivariate normal distribution, i.e., $\boldsymbol{\theta} \sim N(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})$, meaning that

$$\mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})^\top \Sigma_{\mathfrak{q}(\boldsymbol{\theta})}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})\right).$$

Under this assumption, the MF factorization in equation (3.10) becomes:

$$\mathfrak{q}(\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1, \omega) = \mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) \prod_{s=1}^{p}\{\mathfrak{q}_s(\gamma_s)\mathfrak{q}_s(\tau_s^2)\}\mathfrak{q}(v_1)\mathfrak{q}(\omega). \tag{3.11}$$

This change in the factorization does not affect the algorithm updates for the parameters other than $\boldsymbol{\theta}$. Let $\mathbf{X}$ be the matrix obtained by stacking the $p \times 1$ vectors of covariates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as row vectors.

Proposition 3.2.6. *The lower bound for the models* (3.8) *associated to the variational density factorized as in* (3.11)*, can be expressed in closed form*

$$
\begin{aligned}
\log \underline{p}(\boldsymbol{y}, \mathfrak{q}) =& \frac{1}{2} \log |\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}| + \frac{p}{2} - \sum_{s=1}^{p} \Bigg\{ \gamma_s \log \mu_{\mathfrak{q}_s(\gamma_s)} + (1-\gamma_s) \log(1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \\
&+ \alpha_{\mathfrak{q}_s(\tau_s^2)} \log \beta_{\mathfrak{q}_s(\tau_s^2)} + \log \Gamma(\alpha_{\mathfrak{q}_s(\tau_s^2)}) + \beta_{\mathfrak{q}_s(\tau_s^2)} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \\
&+ (\alpha_{\mathfrak{q}_s(\tau_s^2)} + 1) \mu_{\mathfrak{q}_s(\log \tau_s^2)} \Bigg\} + \{ \alpha_{\mathfrak{q}(v_1)} \log \beta_{\mathfrak{q}(v_1)} + \log \Gamma(\alpha_{\mathfrak{q}(v_1)}) \\
&+ \beta_{\mathfrak{q}(v_1)} \mu_{\mathfrak{q}(1/v_1)} + (\alpha_{\mathfrak{q}(v_1)} + 1) \mu_{\mathfrak{q}(\log v_1)} \} - (\alpha_{\mathfrak{q}(\omega)} - 1) \mu_{\mathfrak{q}(\log \omega)}) \\
&- (\beta_{\mathfrak{q}(\omega)} - 1) \mu_{\mathfrak{q}(\log(1-\omega))} + \log \mathrm{B}(\alpha_{\mathfrak{q}(\omega)}, \beta_{\mathfrak{q}(\omega)}) \\
&+ \sum_{i=1}^{n} \left\{ y_i \mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} - \exp\left(\mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2} \mathbf{x}_i^\top \boldsymbol{\Sigma}_{\mu_{\mathfrak{q}(\boldsymbol{\theta})}} \mathbf{x}_i \right) - \log(y_i!) \right\} \\
&- \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\log \tau_s^2)} - \frac{p}{2} \log v_0 - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\gamma_s)} (\log v_0 - \mu_{\mathfrak{q}(\log(1/v_1))}) \\
&- \sum_{s=1}^{p} \frac{(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})_s^2 + (\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})_{ss}}{2} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \Bigg(\mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(1/v_1)} \\
&+ \frac{1}{v_0} (1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \Bigg) + \sum_{s=1}^{p} [(1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \mu_{\mathfrak{q}(\log(1-\omega))} + \mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(\log \omega)}] \\
&+ p(a_\tau \log b_\tau - \log \Gamma(a_\tau)) - \sum_{s=1}^{p} \{ (a_\tau + 1) \mu_{\mathfrak{q}_s(\log \tau_s^2)} - b_\tau \mu_{\mathfrak{q}_s(1/\tau_s^2)} \} \\
&+ a_{v_1} \log b_{v_1} - \log \Gamma(a_{v_1}) - (a_{v_1} + 1) \mu_{\mathfrak{q}(\log v_1)} - b_{v_1} \mu_{\mathfrak{q}(1/v_1)} \\
&+ (a_\omega - 1) \mu_{\mathfrak{q}(\log \omega)} + (b_\omega - 1) \mu_{\mathfrak{q}(\log(1-\omega))} - \mathrm{B}(a_\omega, b_\omega).
\end{aligned}
$$

Proof. The lower bound can be expressed as follow

$$\begin{aligned}\log \underline{p}(\boldsymbol{y}, \mathfrak{q}) = &\mathrm{H}\{\mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\} + \sum_{s=1}^{p}(\mathrm{H}\{\mathfrak{q}_s(\gamma_s)\} + \mathrm{H}\{\mathfrak{q}_s(\tau_s^2)\}) \\ &+ \mathrm{H}\{\mathfrak{q}(v_1)\} + \mathrm{H}\{\mathfrak{q}(\omega)\} + \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{y} \mid \boldsymbol{\theta})] \\ &+ \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{\theta} \mid \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1)] + \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[\log p(\gamma_s \mid \omega)] \\ &+ \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[\log p(\tau_s^2)] + \mathrm{E}_{\mathfrak{q}}[\log p(v_1)] + \mathrm{E}_{\mathfrak{q}}[\log p(\omega)].\end{aligned} \tag{3.12}$$

We now proceed to compute each component of the above summation individually.

$$\begin{aligned}\mathrm{H}\{\mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\} &= \mathrm{E}_{\mathfrak{q}}[-\log \mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})] \\ &= \mathrm{E}_{\mathfrak{q}}\Bigg[- \Bigg[- \frac{p}{2}\log 2\pi - \frac{1}{2}|\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}| \\ &\quad - \frac{1}{2}(\boldsymbol{\theta} - \mu_{\mathfrak{q}(\boldsymbol{\theta})})^\top \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}^{-1}(\boldsymbol{\theta} - \mu_{\mathfrak{q}(\boldsymbol{\theta})}) \Bigg] \Bigg] \\ &= \frac{p}{2}\log 2\pi + \frac{1}{2}|\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}| + \frac{p}{2}.\end{aligned}$$

$$\begin{aligned}\sum_{s=1}^{p} \mathrm{H}_{\mathfrak{q}}\{\mathfrak{q}_s(\gamma_s)\} &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[-\log p(\gamma_s)] \\ &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[-(\gamma_s \log \mu_{\mathfrak{q}_s(\gamma_s)} + (1-\gamma_s)\log(1-\mu_{\mathfrak{q}_s(\gamma_s)}))] \\ &= -\sum_{s=1}^{p} \left\{\gamma_s \log \mu_{\mathfrak{q}_s(\gamma_s)} + (1-\gamma_s)\log(1-\mu_{\mathfrak{q}_s(\gamma_s)})\right\}.\end{aligned}$$

$$\begin{aligned}\sum_{s=1}^{p} \mathrm{H}\{\mathfrak{q}_s(\tau_s^2)\} &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[-\log \mathfrak{q}_s(\tau_s^2)] \\ &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[-(\alpha_{\mathfrak{q}_s(\tau_s^2)} \log \beta_{\mathfrak{q}_s(\tau_s^2)} - \log \Gamma(\alpha_{\mathfrak{q}_s(\tau_s^2)}) - \frac{\beta_{\mathfrak{q}_s(\tau_s^2)}}{\tau_s^2} \\ &\quad - (\alpha_{\mathfrak{q}_s(\tau_s^2)} + 1)\log(\tau_s^2)]\end{aligned}$$

$$= \sum_{s=1}^{p} [\alpha_{\mathfrak{q}_s(\tau_s^2)} \log \beta_{\mathfrak{q}_s(\tau_s^2)} + \log \Gamma(\alpha_{\mathfrak{q}_s(\tau_s^2)}) + \beta_{\mathfrak{q}_s(\tau_s^2)} \mu_{\mathfrak{q}_s(1/\tau_s^2)} + (\alpha_{\mathfrak{q}_s(\tau_s^2)} + 1) \mu_{\mathfrak{q}_s(\log \tau_s^2)}].$$

$$\begin{aligned} \mathrm{H}_{\mathfrak{q}}\{\mathfrak{q}(v_1)\} &= = \mathrm{E}_{\mathfrak{q}}[-\log \mathfrak{q}(v_1)] \\ &= \mathrm{E}_{\mathfrak{q}}[-(\alpha_{\mathfrak{q}(v_1)} \log \beta_{\mathfrak{q}(v_1)} - \log \Gamma(\alpha_{\mathfrak{q}(v_1)}) - \frac{\beta_{\mathfrak{q}(v_1)}}{v_1} \\ &\quad - (\alpha_{\mathfrak{q}(v_1)} + 1) \log(v_1)] \\ &= \alpha_{\mathfrak{q}(v_1)} \log \beta_{\mathfrak{q}(v_1)} + \log \Gamma(\alpha_{\mathfrak{q}(v_1)}) + \beta_{\mathfrak{q}(v_1)} \mu_{\mathfrak{q}(1/v_1)} \\ &\quad + (\alpha_{\mathfrak{q}(v_1)} + 1) \mu_{\mathfrak{q}(\log v_1)}. \end{aligned}$$

$$\begin{aligned} \mathrm{H}_{\mathfrak{q}}\{\mathfrak{q}(\omega)\} &= \mathrm{E}_{\mathfrak{q}}[-\log \mathfrak{q}(\omega)] \\ &= \mathrm{E}_{\mathfrak{q}}[-((\alpha_{\mathfrak{q}(\omega)} - 1) \log \omega + (\beta_{\mathfrak{q}(\omega)} - 1) \log(1 - \omega) \\ &\quad - \log \mathrm{B}(\alpha_{\mathfrak{q}(\omega)}, \beta_{\mathfrak{q}(\omega)})] \\ &= -(\alpha_{\mathfrak{q}(\omega)} - 1) \mu_{\mathfrak{q}(\log \omega)}) - (\beta_{\mathfrak{q}(\omega)} - 1) \mu_{\mathfrak{q}(\log(1-\omega))} \\ &\quad + \log \mathrm{B}(\alpha_{\mathfrak{q}(\omega)}, \beta_{\mathfrak{q}(\omega)}). \end{aligned}$$

$$\begin{aligned} \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{y} \mid \boldsymbol{\theta})] &= \mathrm{E}_{\mathfrak{q}} \left[\sum_{i=1}^{n} [y_i \mathbf{x}_i^\top \boldsymbol{\theta} - \exp(\mathbf{x}_i^\top \boldsymbol{\theta}) - \log(y_i!)] \right] \\ &= \sum_{i=1}^{n} \left\{ y_i \mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} - \exp\left(\mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2} \mathbf{x}_i^\top \boldsymbol{\Sigma}_{\mu_{\mathfrak{q}(\boldsymbol{\theta})}} \mathbf{x}_i \right) - \log(y_i!) \right\}. \end{aligned}$$

$$\begin{aligned} \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{\theta} \mid \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1)] &= \mathrm{E}_{\mathfrak{q}} \left[-\frac{p}{2} \log 2\pi - \frac{1}{2} \sum_{s=1}^{p} \log \tau_s^2 - \frac{p}{2} \log v_0 \right. \\ &\quad \left. - \frac{1}{2} \log \frac{v_0}{v_1} \sum_{s=1}^{p} \gamma_s - \frac{1}{2} \sum_{s=1}^{p} \left\{ \left(\frac{\gamma_s}{v_1} + \frac{1 - \gamma_s}{v_0} \right) \frac{\theta_s^2}{\tau_s^2} \right\} \right] \\ &= -\frac{p}{2} \log 2\pi - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\log \tau_s^2)} - \frac{p}{2} \log v_0 \\ &\quad - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\gamma_s)} \log v_0 - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(\log(1/v_1))} \end{aligned}$$

$$- \sum_{s=1}^{p} \frac{(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})_s^2 + (\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})_{ss}}{2} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \Bigg(\mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \Bigg).$$

$$\begin{aligned} \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[\log p(\gamma_s \mid \omega)] &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[(1-\gamma_s)\log(1-\omega) + \gamma_s \log \omega] \\ &= \sum_{s=1}^{p} [(1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \mu_{\mathfrak{q}(\log(1-\omega))} + \mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(\log \omega)}]. \end{aligned}$$

$$\begin{aligned} \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}}[\log p(\tau_s^2)] &= \sum_{s=1}^{p} \mathrm{E}_{\mathfrak{q}} \left[a_\tau \log b_\tau - \log \Gamma(a_\tau) - (a_\tau + 1) \log \tau_s^2 - \frac{b_\tau}{\tau_s^2} \right] \\ &= a_\tau \log b_\tau - \log \Gamma(a_\tau) - \sum_{s=1}^{p} \{(a_\tau + 1) \mu_{\mathfrak{q}_s(\log \tau_s^2)} - b_\tau \mu_{\mathfrak{q}_s(1/\tau_s^2)}\}. \end{aligned}$$

$$\begin{aligned} \mathrm{E}_{\mathfrak{q}}[\log p(v_1)] &= \mathrm{E}_{\mathfrak{q}}[a_{v_1} \log b_{v_1}] - \log \Gamma(a_{v_1}) - (a_\tau + 1) \log v_1 - \frac{b_{v_1}}{v_1} \\ &= a_{v_1} \log b_{v_1} - \log \Gamma(a_{v_1}) - (a_{v_1} + 1) \mu_{\mathfrak{q}(\log v_1)} - b_{v_1} \mu_{\mathfrak{q}(1/v_1)} \end{aligned}$$

$$\begin{aligned} \mathrm{E}_{\mathfrak{q}}[\log p(\omega)] &= \mathrm{E}_{\mathfrak{q}}[(a_\omega - 1)\log \omega + (b_\omega - 1)\log(1-\omega) - \mathrm{B}(a_\omega, b_\omega)] \\ &= (a_\omega - 1)\mu_{\mathfrak{q}(\log \omega)} + (b_\omega - 1)\mu_{\mathfrak{q}(\log(1-\omega))} - \mathrm{B}(a_\omega, b_\omega). \end{aligned}$$

By substituting these quantities into equation (3.12), and simplifying the common term $\frac{p}{2} \log 2\pi$ that appears in both the entropy $\mathrm{H}\{\mathfrak{q}(\boldsymbol{\theta}; \mu_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\}$ and the expectation $\mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{\theta} \mid \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1)]$, we obtain the result stated in Proposition 3.2.6. □

The optimal parameters $\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}$ and $\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}$ are obtained by maximizing the $\boldsymbol{\theta}$-localized component of the lower bound, $\log \underline{p}(\boldsymbol{y}; \mathfrak{q})^{[\boldsymbol{\theta}]}$. The closed-form expression of this component is provided in the following proposition.

Proposition 3.2.7. *The $\boldsymbol{\theta}$-localized component of the lower bound is given by*

$$\begin{aligned}\log \underline{p}(\boldsymbol{y}; \mathfrak{q})^{[\boldsymbol{\theta}]} = &\sum_{i=1}^{n} \left\{ y_i \mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} - \exp\left(\mathbf{x}_i^\top \mu_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2} \mathbf{x}_i^\top \boldsymbol{\Sigma}_{\mu_{\mathfrak{q}(\boldsymbol{\theta})}} \mathbf{x}_i \right) - \log(y_i!) \right\} \\ &+ \frac{1}{2} |\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}| + \frac{p}{2} - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\log \tau_s^2)} - \frac{p}{2} \log v_0 \\ &- \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\gamma_s)} \log v_0 - \frac{1}{2} \sum_{s=1}^{p} \mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(\log(1/v_1))} \\ &- \sum_{s=1}^{p} \frac{(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})_s^2 + (\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})_{ss}}{2} \mu_{\mathfrak{q}_s(1/\tau_s^2)} \Bigg(\mu_{\mathfrak{q}_s(\gamma_s)} \mu_{\mathfrak{q}(1/v_1)} \\ &+ \frac{1}{v_0} (1 - \mu_{\mathfrak{q}_s(\gamma_s)}) \Bigg).\end{aligned}$$

Proof.

$$\begin{aligned}\log \underline{p}(\boldsymbol{y}; \mathfrak{q})^{[\boldsymbol{\theta}]} &= \mathrm{H}\{\mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\} + \overline{\mathrm{H}}\{\mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\} \\ &= \mathrm{E}_{\mathfrak{q}}[-\log \mathfrak{q}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})] + \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{y} \mid \boldsymbol{\theta})] \\ &\quad + \mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{\theta} \mid \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1)] \end{aligned} \tag{3.13}$$

These components have already been computed in Proposition 3.2.6. By substituting them into equation (3.13) and simplifying the common term $\frac{p}{2} \log 2\pi$, which appears in both the entropy $\mathrm{H}\{\mathfrak{q}(\boldsymbol{\theta}; \mu_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})\}$ and the expectation $\mathrm{E}_{\mathfrak{q}}[\log p(\boldsymbol{\theta} \mid \boldsymbol{\gamma}, \boldsymbol{\tau}^2, v_1)]$, we obtain the result stated in Proposition 3.2.7. □

From the Result 2 in Rohde and Wand (2016) the natural fixed-point iteration for $\mathfrak{q}(\boldsymbol{\theta})$ corresponding to the $N(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})$ density function is equivalent to the following updating scheme:

$$\begin{cases} \boldsymbol{v}_{\mathfrak{q}(\boldsymbol{\theta})} \leftarrow \mathbf{D}_{\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}} \mathrm{NonEntropy}(\mathfrak{q}; \mu_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})^\top; \\ \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})} \leftarrow \left\{ -\mathbf{H}_{\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}} \mathrm{NonEntropy}(\mathfrak{q}; \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}, \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) \right\}^{-1}; \\ \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} \leftarrow \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})} \boldsymbol{v}_{\mathfrak{q}(\boldsymbol{\theta})}; \end{cases}$$

where

$$\begin{aligned}\overline{\mathrm{H}}(\mathfrak{q};\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})},\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) = \sum_{i=1}^{n}\left\{y_i\mathbf{x}_i^{\top}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} - \exp\left\{\mathbf{x}_i^{\top}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2}\mathbf{x}_i^{\top}\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\mathbf{x}_i\right\} - \log(y_i!)\right\} \\ - \frac{p}{2}\log 2\pi - \sum_{i=1}^{p}\left\{\frac{\mu_{\mathfrak{q}_i(\log\tau_i^2)}}{2} - \frac{(1-\mu_{\mathfrak{q}_i(\gamma_i)})}{2}\log\frac{1}{v_0} - \frac{\mu_{\mathfrak{q}_i(\gamma_i)}}{2}\mu_{\mathfrak{q}(\log(1/v_1))}\right. \\ \left. + \frac{(\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})_i^2 + (\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})_{ii}}{2}\mu_{\mathfrak{q}_i(1/\tau_i^2)}\left(\mu_{\mathfrak{q}_i(\gamma_i)}\mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1-\mu_{\mathfrak{q}_i(\gamma_i)})\right)\right\}.\end{aligned}$$

Then:

$$\begin{aligned}\mathbf{D}_{\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}}\mathrm{NonEntropy}(\mathfrak{q};\mu_{\mathfrak{q}(\boldsymbol{\theta})},\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) &= \sum_{i=1}^{n}\mathbf{x}_i^{\top}y_i - \sum_{i=1}^{n}\mathbf{x}_i^{\top}\exp\left\{\mathbf{x}_i^{\top}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2}\mathbf{x}_i^{\top}\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\mathbf{x}_i\right\} \\ &\quad - \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}^{\top}\boldsymbol{\Lambda},\end{aligned}$$

where

$$\boldsymbol{\Lambda} = \mathrm{diag}\begin{bmatrix}\mu_{\mathfrak{q}_1(1/\tau_1^2)}\left\{\mu_{\mathfrak{q}_1(\gamma_1)}\mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1-\mu_{\mathfrak{q}_1(\gamma_1)})\right\} \\ \vdots \\ \mu_{\mathfrak{q}_p(1/\tau_p^2)}\left\{\mu_{\mathfrak{q}_p(\gamma_p)}\mu_{\mathfrak{q}(1/v_1)} + \frac{1}{v_0}(1-\mu_{\mathfrak{q}_p(\gamma_p)})\right\}\end{bmatrix},$$

and

$$\begin{aligned}\mathbf{H}_{\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}}\overline{\mathrm{H}}(\mathfrak{q};\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})},\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}^{\top}}\mathbf{D}_{\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}}\mathrm{NonEntropy}(\mathfrak{q};\mu_{\mathfrak{q}(\boldsymbol{\theta})},\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}) \\ &= -\sum_{i=1}^{n}\mathbf{x}_i^{\top}\exp\left\{\mathbf{x}_i^{\top}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2}\mathbf{x}_i^{\top}\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\mathbf{x}_i\right\}\mathbf{x}_i - \boldsymbol{\Lambda}.\end{aligned}$$

So we have update scheme for the parameters of $\mathfrak{q}(\boldsymbol{\theta};\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})},\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})$ as follow

$$\boldsymbol{\lambda} \longleftarrow \exp\left\{\mathbf{X}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2}\mathrm{diagonal}(\mathbf{X}\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\mathbf{X}^{\top})\right\};$$

$$\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})} \longleftarrow \left\{\mathbf{X}^{\top}\mathrm{diag}(\boldsymbol{\lambda})\mathbf{X} + \boldsymbol{\Lambda}\right\}^{-1};$$

$$\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} \longleftarrow \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\left\{\mathbf{X}^{\top}(\boldsymbol{y}-\boldsymbol{\lambda}) - \boldsymbol{\Lambda}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}\right\};$$

with the convergence assessed by checking the negligible increase of $\log \underline{p}(\boldsymbol{y}; \mathfrak{q})^{[\boldsymbol{\theta}]}$ after each update. In practice, for simplicity, instead of checking convergence by comparing the difference between the full lower bound and the $\boldsymbol{\theta}$-localized component, we iterate the algorithm until the relative change in the updated parameters becomes negligible. This procedure is summarized in Algorithm 4. Subsequently, variable selection is performed by comparing $\mu_{\mathfrak{q}_s(\gamma_s)}$ against a fixed threshold, for example 0.5.

The structure learning procedure then proceeds as described in Algorithm 1, with the distinction that variable selection for the Poisson regression model is carried out using the MFVB algorithm instead of MCMC sampling.

3.3 Simulation studies

In this section, we apply the previously introduced MFVB algorithm to recover the graph structures using the datasets generated in the previous chapter. In what follows, we will refer the application of MFVB to ChainLPGM as MFVBChain-LPGM, whereas the application to Bayes-LPGM will be referred to as MFVBBayes-LPGM. Our objective is to reduce the running time of the MCMC algorithms while maintaining a high level of accuracy. Similar to the sampling algorithms based on MCMC, the value $v_0 = 0.00025$ is selected, and the hyperparameter are set to $(a_\omega, b_\omega) = (1, 1)$, while two cases are considered for (a_τ, b_τ), namely $(5, 25)$ and $(3, 50)$. For v_1, a non-informative inverse-gamma prior is adopted by setting $(a_{v_1}, b_{v_1}) = (1, 1)$. The Monte Carlo averages of TP, PPV, Se, and runtime, obtained by simulating 500 datasets from graphs with $p = 10$ and $p = 100$, are reported in Tables 3.1 and 3.2, respectively. The reported computational time corresponds to the runtime for a single dataset. Runtime measurements were performed on an AMD Epyc 7763 2.45GHz processor running R 4.2.1 with 16 cores.

A comparison between the results presented in Chapter 2 and those reported in this chapter demonstrates the computational superiority of the MFVB approach over MCMC, while maintaining a high level of predictive

Algorithm 4: *Iterative scheme for semiparametric MFVB.*

Data input: $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$.

Hyperparameter input: $\upsilon_0 > 0$ (small), $a_\tau > 0$, $b_\tau > 0$, $a_{\upsilon_1} > 0$, $b_{\upsilon_1} > 0$, $a_\omega > 0$ and $b_\omega > 0$.

Initialize: $\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} \in \mathbb{R}^p$, $\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}$ symmetric positive definite $p \times p$ matrix, $\mu_{\mathfrak{q}_s(1/\tau_s^2)} > 0$, $\mu_{\mathfrak{q}_1(\gamma_1)}, \ldots, \mu_{\mathfrak{q}_N(\gamma_p)} \in [0,1]$, $\mu_{\mathfrak{q}(1/\upsilon_1)} > 0$, $\mu_{\mathfrak{q}(\log(\upsilon_1))} \in \mathbb{R}$, $\mu_{\mathfrak{q}(\log(\omega))} \leq 0$, $\mu_{\mathfrak{q}(\log(1-\omega))} \leq 0$.

Set: $\alpha_{\mathfrak{q}_s(\tau_s^2)} = 1/2 + a_\tau$, $s = 1, \ldots, p$.

while *convergence not reached* **do**

  while *convergence not reached* **do**

    $\boldsymbol{\lambda} \longleftarrow \exp\left\{\mathbf{X}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \frac{1}{2}\text{diagonal}(\mathbf{X}\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\mathbf{X}^\top)\right\}$

    $\boldsymbol{\Lambda} \longleftarrow \text{diag}\begin{bmatrix} \mu_{\mathfrak{q}(1/\tau_1^2)}\left\{\mu_{\mathfrak{q}(\gamma_1)}\mu_{\mathfrak{q}(1/\upsilon_1)} + \frac{1}{\upsilon_0}(1 - \mu_{\mathfrak{q}(\gamma_1)})\right\} \\ \vdots \\ \mu_{\mathfrak{q}(1/\tau_p^2)}\left\{\mu_{\mathfrak{q}(\gamma_p)}\mu_{\mathfrak{q}(1/\upsilon_1)} + \frac{1}{\upsilon_0}(1 - \mu_{\mathfrak{q}(\gamma_p)})\right\} \end{bmatrix}$

    $\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})} \longleftarrow \left\{\mathbf{X}^\top \text{diag}(\boldsymbol{\lambda})\mathbf{X} + \boldsymbol{\Lambda}\right\}^{-1}$

    $\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} \longleftarrow \boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})} + \boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}\left\{\mathbf{X}^\top(\boldsymbol{y} - \boldsymbol{\lambda}) - \boldsymbol{\Lambda}\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}\right\}$

  end while

  for $s = 1, \ldots, p$ **do**

    $\mu_{\mathfrak{q}(\theta_s^2)} \longleftarrow (\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})})_s^2 + (\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})})_{ss}$

    $\kappa_s \longleftarrow \frac{1}{2}\left\{\mu_{\mathfrak{q}(\theta_s^2)}\mu_{\mathfrak{q}_s(1/\tau_s^2)}\left(\frac{1}{\upsilon_0} - \mu_{\mathfrak{q}(1/\upsilon_1)}\right) + \log(\upsilon_0) - \mu_{\mathfrak{q}(\log(\upsilon_1))}\right\}$

    $\quad + \mu_{\mathfrak{q}(\log(\omega))} - \mu_{\mathfrak{q}(\log(1-\omega))}$

    $\mu_{\mathfrak{q}_s(\gamma_s)} \longleftarrow e^{\kappa_s}/(1 + e^{\kappa_s})$

    $\beta_{\mathfrak{q}_s(\tau_s^2)} \longleftarrow \frac{1}{2}\mu_{\mathfrak{q}(\theta_s^2)}\left\{\mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(1/\upsilon_1)} + \frac{1}{\upsilon_0}(1 - \mu_{\mathfrak{q}_s(\gamma_s)})\right\} + b_\tau$

    $\mu_{\mathfrak{q}_s(1/\tau_s^2)} \longleftarrow \alpha_{\mathfrak{q}_s(\tau_s^2)}/\beta_{\mathfrak{q}_s(\tau_s^2)}$

  end for

  $\alpha_{q(\upsilon_1)} \longleftarrow \frac{1}{2}\sum_{s=1}^p \mu_{\mathfrak{q}_s(\gamma_s)} + a_{\upsilon_1}$

  $\beta_{q(\upsilon_1)} \longleftarrow \frac{1}{2}\sum_{s=1}^p \mu_{\mathfrak{q}_s(\gamma_s)}\mu_{\mathfrak{q}(\theta_s^2)}\mu_{\mathfrak{q}_s(1/\tau_s^2)} + b_{\upsilon_1}$

  $\mu_{q(1/\upsilon_1)} \longleftarrow \alpha_{q(\upsilon_1)}/\beta_{q(\upsilon_1)}$

  $\mu_{q(\log(\upsilon_1))} \longleftarrow \log(\beta_{q(\upsilon_1)}) - \psi(\alpha_{q(\upsilon_1)})$, where ψ is digamma function

  $\alpha_{q(\omega)} \longleftarrow a_\omega + \sum_{s=1}^p \mu_{\mathfrak{q}_s(\gamma_s)}$

  $\beta_{q(\omega)} \longleftarrow p + b_\omega - \sum_{s=1}^p \mu_{\mathfrak{q}_s(\gamma_s)}$

  $\mu_{\mathfrak{q}(\log(\omega))} \longleftarrow \psi(\alpha_{q(\omega)}) - \psi(\alpha_{q(\omega)} + \beta_{q(\omega)})$

  $\mu_{\mathfrak{q}(\log(1-\omega))} \longleftarrow \psi(\beta_{q(\omega)}) - \psi(\alpha_{q(\omega)} + \beta_{q(\omega)})$

end while

Output: $\boldsymbol{\mu}_{\mathfrak{q}(\boldsymbol{\theta})}$, $\boldsymbol{\Sigma}_{\mathfrak{q}(\boldsymbol{\theta})}$, $\alpha_{q(\upsilon_1)}$, $\beta_{q(\upsilon_1)}$, $\alpha_{q(\omega)}$, $\beta_{q(\omega)}$, $\alpha_{\mathfrak{q}_s(\tau_s^2)}$, $\beta_{\mathfrak{q}_s(\tau_s^2)}$, and $\mu_{\mathfrak{q}_s(\gamma_s)}$ for $s = 1, \ldots, p$.

(a_τ, b_τ)	type	n	MFVBChain-LPGM				MFVBBayes-LPGM			
			TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
(5,25)	Random	100	5.074	0.869	0.507	0.043	3.598	0.972	0.360	0.066
		200	8.470	0.903	0.847	0.097	7.228	0.998	0.723	0.193
		500	9.808	0.777	0.981	0.205	9.732	1.000	0.973	0.388
	Scale-free	100	4.266	0.859	0.427	0.035	2.214	0.985	0.221	0.067
		200	6.552	0.794	0.655	0.079	3.604	0.996	0.360	0.153
		500	8.294	0.736	0.829	0.365	6.244	0.999	0.624	0.402
	Hub	100	3.004	0.726	0.376	0.025	1.482	0.852	0.185	0.072
		200	4.754	0.671	0.594	0.023	2.450	0.979	0.306	0.119
		500	7.260	0.622	0.908	0.358	6.470	0.999	0.809	0.585
(3,50)	Random	100	8.870	0.927	0.887	0.015	8.084	0.982	0.808	0.027
		200	9.782	0.973	0.978	0.011	9.502	0.998	0.950	0.018
		500	9.998	0.953	0.999	0.026	9.972	1.000	0.997	0.045
	Scale-free	100	6.778	0.873	0.678	0.035	4.844	0.985	0.484	0.063
		200	8.156	0.846	0.816	0.041	6.572	0.992	0.657	0.090
		500	9.194	0.822	0.919	0.170	8.780	0.999	0.878	0.108
	Hub	100	5.052	0.769	0.632	0.031	3.983	0.934	0.498	0.081
		200	7.206	0.742	0.901	0.017	6.171	0.990	0.771	0.045
		500	7.932	0.625	0.992	0.046	7.688	0.998	0.961	0.035

Table 3.1: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 100,\ 200,\ 500$ from graph with $p = 10$ variables with Poisson node conditional distribution with $(a_{v_1}, b_{v_1}) = (1, 1),\ v_0 = 0.00025$ and $\theta_{st} = -1,\ \forall\ (s,t) \in E$ using MFVB.

accuracy. Specifically, for the case $p = 10$, the MFVB algorithm runs approximately 20 to 40 times faster than MCMC. Moreover, with suitably chosen hyperparameter, the running time of MFVB can even outperform that of the frequentist PC-LPGM approach in certain scenarios. The difference in computational time becomes even more pronounced when $p = 100$, where MFVB is about 100 times faster than MCMC under appropriate hyperparameter settings. These comparisons clearly indicate that the MFVB algorithm substantially outperforms MCMC in terms of computational efficiency while preserving a high degree of accuracy.

We now focus on a comparison between MFVBChain-LPGM and MFVBBayes-

(a_τ, b_τ)	type	n	MFVBChain-LPGM				MFVBBayes-LPGM			
			TP	PPV	Se	Time(mins)	TP	PPV	Se	Time(mins)
(5,25)	Random	500	79.928	0.885	0.807	0.375	77.092	0.999	0.779	0.640
		1000	90.890	0.832	0.918	0.835	91.312	0.999	0.922	1.247
		1500	93.950	0.797	0.949	1.184	95.202	1.000	0.962	1.441
	Scale-free	500	72.494	0.867	0.725	0.343	68.932	0.999	0.689	0.597
		1000	85.724	0.843	0.857	0.988	85.190	1.000	0.852	1.360
		1500	89.810	0.834	0.898	1.621	91.714	1.000	0.917	2.057
	Hub	500	34.444	0.594	0.405	0.152	1.332	0.312	0.016	0.306
		1000	48.572	0.650	0.571	0.380	11.062	0.950	0.130	1.288
		1500	57.188	0.629	0.673	0.636	19.594	0.989	0.231	2.608
(3,50)	Random	500	96.386	0.978	0.974	0.059	93.384	0.999	0.943	0.085
		1000	98.590	0.980	0.996	0.147	97.898	0.999	0.989	0.197
		1500	98.892	0.980	0.999	0.222	98.704	1.000	0.997	0.376
	Scale-free	500	94.314	0.970	0.943	0.075	88.274	0.999	0.887	0.137
		1000	98.066	0.966	0.981	0.153	95.226	1.000	0.952	0.261
		1500	99.058	0.962	0.991	0.271	97.306	1.000	0.973	0.441
	Hub	500	61.520	0.772	0.724	0.224	19.090	0.911	0.225	0.412
		1000	73.622	0.894	0.866	0.257	32.208	0.981	0.379	0.446
		1500	78.804	0.914	0.927	0.417	37.564	0.994	0.442	0.896

Table 3.2: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 500, 1000, 1500$ from graph with $p = 100$ variables with Poisson node conditional distribution with $(a_{v_1}, b_{v_1}) = (1, 1)$, $v_0 = 0.00025$ and $\theta_{st} = -1$, $\forall\ (s,t) \in E$ using MFVB.

LPGM. For the case $p = 10$, the results reported in Table 3.1 demonstrate the good performance of the MFVB algorithm, particularly for MFVBBayes-LPGM. Moreover, although MFVBChain-LPGM yields substantial improvements in terms of TP and Se, it also leads to a reduction in PPV in certain scenarios. As discussed in Chapter 2, this phenomenon may arise because the data do not strictly conform to the model specification and when the joint distribution is not Markov with respect to the underlying graph.

A comparison between the hyperparameter settings $(a_\tau, b_\tau) = (5, 25)$ and $(a_\tau, b_\tau) = (3, 50)$ reveals a clear influence of these parameters on the performance of the algorithm. Specifically, replacing $(a_\tau, b_\tau) = (5, 25)$ with $(3, 50)$ leads to substantial improvements in the results. The values of TP, PPV, and Se all increase, with the effect being particularly pronounced for smaller

sample sizes. For example, when $n = 100$, the performance of MFVBChain-LPGM on random graphs improves from TP = 5.074, PPV = 0.869, and Se = 0.507 to TP = 8.870, PPV = 0.927, and Se = 0.887. The improvement is even more evident for MFVBBayes-LPGM, where TP, PPV, and Se increase from 3.598, 0.972, and 0.360 to 8.084, 0.982, and 0.808, respectively.

With respect to runtime, as expected, MFVBChain-LPGM demonstrates improved computational efficiency compared with MFVBBayes-LPGM. Moreover, the choice of hyperparameter also affects runtime, as it influences the number of iterations required for convergence. For instance, when $n = 100$, the runtime of MFVBChain-LPGM on random graphs decreases from 0.043 seconds to 0.015 seconds. Notably, when using the hyperparameter setting $(a_\tau, b_\tau) = (3, 50)$, the runtimes of both MFVBChain-LPGM and MFVBBayes-LPGM with a sample size of $n = 200$ are even shorter than those observed with $n = 100$. This phenomenon occurs because the number of iterations required to achieve convergence is substantially smaller for $n = 200$ than for $n = 100$.

For the case $p = 100$, MFVBBayes-LPGM continues to perform well on random and scale-free graphs; however, its performance on hub graphs is considerably limited, as reflected by the low TP values even when the sample size is as large as $n = 1500$. In contrast, MFVBChain-LPGM proves to be particularly effective for hub graphs. For example, when $n = 1500$ and $(a_\tau, b_\tau) = (3, 50)$, MFVBChain-LPGM achieves TP = 78.804 and Se = 0.927, which are markedly higher than the corresponding values of MFVBBayes-LPGM (TP = 37.564 and Se = 0.442), while still maintaining a high PPV of 0.914 compared to 0.994.

The impact of hyperparameter selection remains evident in the high-dimensional setting with $p = 100$, particularly for smaller sample sizes such as $n = 500$. Specifically, when (a_τ, b_τ) is changed from $(5, 25)$ to $(3, 50)$, the performance of the algorithm improves substantially. For example, in the case of random graphs, TP, PPV, and Se increase from 79.928, 0.885, and 0.807 to 96.386, 0.978, and 0.974, respectively. For MFVBBayes-LPGM,

the corresponding values improve from 77.092, 0.999, and 0.779 to 93.384, 0.999, and 0.943. For larger sample sizes, this adjustment of hyperparameter notably enhances the PPV of MFVBChain-LPGM; for instance, when $n = 1500$, PPV increases from 0.797 to 0.980 for random graphs, from 0.834 to 0.962 for scale-free graphs, and from 0.629 to 0.914 for hub graphs. By contrast, the effect of hyperparameter changes on MFVBBayes-LPGM becomes less pronounced at larger sample sizes, except in the case of hub graphs, where TP, PPV, and Se increase from 19.594, 0.989, and 0.231 to 37.564, 0.994, and 0.442 for $n = 1500$. Notably, under the hyperparameter setting $(a_\tau, b_\tau) = (3, 50)$, MFVBChain-LPGM proves particularly effective for hub graphs, achieving consistently high TP, PPV, and Se values of 78.804, 0.914, and 0.927, respectively.

In terms of runtime, MFVBChain-LPGM continues to demonstrate superior computational efficiency, achieving faster running times compared with MFVBBayes-LPGM. Furthermore, runtime can be substantially improved through appropriate hyperparameter settings; in particular, the configuration $(a_\tau, b_\tau) = (3, 50)$ facilitates faster convergence, thereby leading to a notable reduction in computational time.

3.4 Real data analysis

In this section, the real dataset focusing on the olfactory sensory neuron lineage, which was analyzed in the previous chapter, is re-examined. Based on the findings obtained in the previous chapter, the hyperparameter are specified as $(a_\tau, b_\tau) = (5, 25)$. The structure of the gene network was reconstructed by applying MFVBChain-LPGM and MFVBBayes-LPGM to the normalized dataset. The resulting reconstructions are visualized in Figures 3.1 and 3.2, respectively.

When applying MFVBBayes-LPGM, we again obtained a very sparse graph containing only 15 edges. In contrast, the application of MFVBChain-LPGM produced a biologically more meaningful structure, in which we iden-

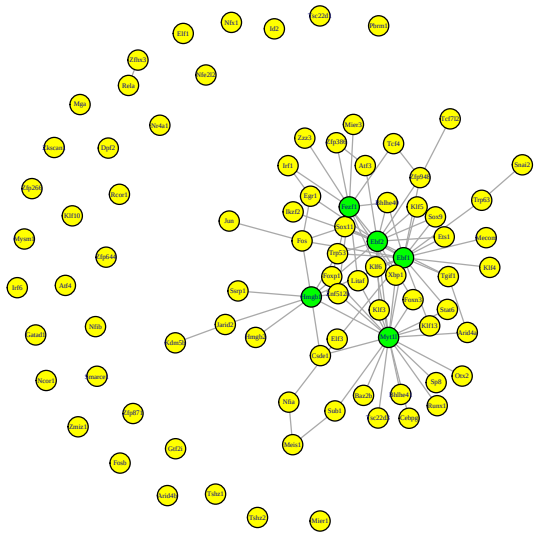


Figure 3.1: Olfactory Neuron gene network estimated by the algorithm using MFVBChain-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$ (hub nodes coloured green).

tified five hub nodes with at least 10 edges each: *Fezf1, Ebf1, Ebf2, Myt1l,* and *Hmgb1*. In addition to the roles of *Ebf1* and *Ebf2*, which were discussed in the previous chapter, it is noteworthy that *Hmgb1* is essential for the proliferation and differentiation of neural stem/progenitor cells (Zhao et al., 2020). Comparing these results with those presented in the previous chapter, the outcomes obtained using MFVB and MCMC are largely consistent. While MFVBBayes-LPGM and Bayes-LPGM both produce sparse graphs with 15 and 18 edges, respectively, MFVBChain-LPGM and Chain-LPGM, despite certain differences, are still able to identify key genes, with the central nodes being *Ebf1*, *Ebf2*, and *Myt1l*.

For Bayes-LPGM, we also examined the modified ORMFVBBayes-LPGM

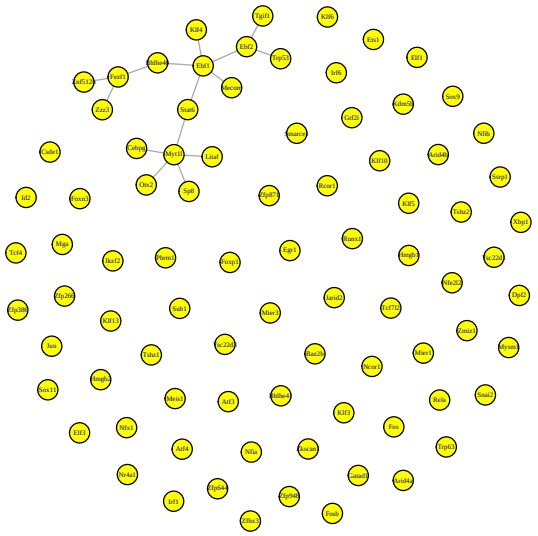


Figure 3.2: Olfactory Neuron gene network estimated by the algorithm using MFVBBayes-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$.

algorithm, as introduced in the previous chapter. The results obtained from this modified algorithm are illustrated in Figure 3.3. In this case, we identified four hub nodes: *Fezf1, Ebf1, Ebf2,* and *Myt1l*. Compared with MFVBChain-LPGM, which additionally detected *Hmgb1* as a hub node, ORMFVBBayes-LPGM yields a more conservative network structure, suggesting that MFVBChain-LPGM may offer greater sensitivity in uncovering biologically relevant hubs.

3.5 Conclusion

In this chapter, we provided an overview of the MFVB approach for approximating posterior distributions by seeking an approximating distribution

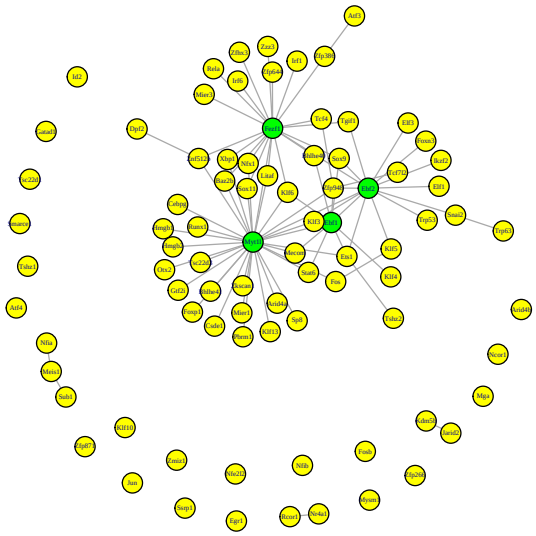


Figure 3.3: Olfactory Neuron gene network estimated by the algorithm using ORMFVBBayes-LPGM with $(a_\tau, b_\tau) = (5, 25)$ and $v_0 = 0.00025$ (hub nodes coloured green).

that minimized the Kullback–Leibler divergence under certain assumptions. In the non-parametric setting, MFVB assumed a mean-field independence structure, whereas in the semi-parametric setting, additional assumptions were imposed such that the approximating distribution belonged to a known parametric family. We applied semi-parametric MFVB to the problem of variable selection in Poisson regression models for structure learning. In this context, we employed a spike-and-slab prior similar to that introduced in Chapter 2, with minor modifications to enhance model flexibility while retaining computational advantages. Subsequently, we applied this approach to structure learning by replacing the MCMC sampling step in Algorithm 1 with the MFVB method for variable selection in Poisson regression.

Simulation results demonstrated the effectiveness of MFVB in substantially reducing computation time while maintaining a high level of accuracy. They also highlighted the sensitivity of the method to hyperparameter choices, showing that appropriate specification of hyperparameter could significantly improve performance. In addition, the results confirmed the advantage of Chain-LPGM in reducing computational cost, as well as its superior performance, particularly for hub graphs.

The analysis of real data further supported these findings. Although some discrepancies arose when compared with results obtained via MCMC sampling, our method was still able to identify key hub nodes that were biologically important in the process under study.

Chapter 4

Structure learning of differential graphical models

Differential network analysis is particularly valuable in applications such as biology and neuroscience, where the aim is to identify changes in interaction patterns across different conditions, thereby providing insights into the underlying mechanisms of complex systems. In this chapter, we present a novel approach of structure learning for differential network analysis. We propose a Bayesian variable selection approach in the context of density ratio estimation. Tailored on exponential undirected graphical models, the approach allows to easily infer the difference in the structures of the models in two conditions, avoiding separate estimations of the two network structures.

4.1 Overview of the problem

In this section, we provide a brief overview of the problem of detecting changes in the structure of a graphical model between two conditions. Our proposal works for models belonging to the exponential family. For this reason, the presentation will be centered on this particular instance.

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \dots, p\}$. Consider an exponential family expression of an undirected graphical model,

where the joint distribution of $\mathbf{X} = (X_1, \ldots, X_p)$ takes the form

$$\mathbb{P}_{(\boldsymbol{\eta},\boldsymbol{\theta})}(\mathbf{x}) = \exp\Bigg(\sum_{s\in V} \eta_s B_s(x_s) + \sum_{s,t\in V} \theta_{st} B_s(x_s) B_t(x_t) + \sum_{s\in V} C_s(x_s) - A(\boldsymbol{\eta},\boldsymbol{\theta}) \Bigg), \tag{4.1}$$

where $\{\{B_s(x_s)\}_{s\in V}, \{B_s(x_s)B_t(x_t)\}_{s,t\in V}\}$ is the sufficient statistic associated with the exponential law under consideration, $\sum_{s\in V} C_s(x_s)$ is the base measure, and $A(\boldsymbol{\eta},\boldsymbol{\theta})$ is the normalizing constant. The full parameter vector is defined as

$$(\boldsymbol{\eta},\boldsymbol{\theta}) = \big(\eta_1, \ldots, \eta_p, \theta_{11}, \ldots, \theta_{1p}, \theta_{22}, \ldots, \theta_{2p}, \ldots, \theta_{pp}\big).$$

Two common examples are the Gaussian graphical models and the Poisson graphical models. Here, we have:

- **Gaussian graphical models:** here, $B_s(x_s) = x_s$, so that the corresponding sufficient statistic at each node becomes $\{x_s, x_s^2\}$;
- **Poisson graphical models:** in this case, the joint distribution follows a log-linear form with base measure $\prod_{s=1}^p \frac{1}{x_s!}$, and $B_s(x_s) = x_s$, where θ_{st} correspond to the coefficients controlling the log-linear interactions. See also (2.5).

As in previous chapters, without loss of generality, we can set $\boldsymbol{\eta} = \mathbf{0}$, since our primary interest lies in estimating the conditional dependencies encoded in $\boldsymbol{\theta}$.

In the context of differential graphical models, we consider two sets of independent samples drawn separately from an exponential law on $\mathbb{R}^p$ under two conditions, 1 and 2 say, i.e.,

$$\mathbf{x}^{(1)} = \{\mathbf{x}_1^{(i)}\}_{i=1}^{n_1} \overset{iid}{\sim} \mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x}); \;\; \mathbf{x}^{(2)} = \{\mathbf{x}_2^{(i)}\}_{i=1}^{n_2} \overset{iid}{\sim} \mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x}).$$

Both samples come from the same exponential law, possibly characterized in the two conditions by two different values for the parameter $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$ say.

In differential graphical modeling, the objective is to detect and characterise the changes in the model parameters from 1 to 2, that is, to infer the difference

$$\boldsymbol{\theta} \;=\; \boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(2)}.$$

In structure learning, in particular, it is of interest recovering the support of $\boldsymbol{\theta}$, i.e.,

$$S_{\boldsymbol{\theta}} \;=\; \{(s,t) : \theta_{st} \neq 0\},$$

which defines the differential network, i.e., the set of edges whose associated parameters change between the two conditions. In practice, we seek sparse and statistically reliable estimates of $\boldsymbol{\theta}$ that permit both accurate detection of structural changes and interpretability in high-dimensional settings.

Recent advances in differential network estimation have focused on developing direct estimation methods that eliminate the need to infer individual networks separately. Traditional approaches estimate the dependence matrices for each condition and subsequently compute their difference. However, such methods are computationally demanding and may perform poorly when the individual networks are dense, even if their differential structure is sparse.

To address these limitations, Zhao et al. (2014) proposed a Direct Differential Graphical Model within the Gaussian graphical model framework. Their method directly estimates the difference between two precision matrices by solving an L_1-regularized optimization problem, assuming sparsity only on the differential matrix. This formulation ensures statistical consistency and computational efficiency, while relaxing the sparsity assumption on the individual networks.

Building upon a similar motivation but from a broader perspective, Liu et al. (2014) introduced a density ratio based approach for detecting structural changes in Markov networks. Instead of modeling two networks independently, their method estimates the ratio of the two probability densities directly using the Kullback–Leibler Importance Estimation Procedure. This formulation enables efficient computation of normalization terms through sample averages and naturally incorporates sparsity via L_1 or elastic-net

regularization. Notably, this framework extends beyond the Gaussian assumption, allowing for non-Gaussian distributions in a more general setting.

4.2 Our proposal

It is worth noting that the difference vector naturally arises when considering the density ratio between the two distributions (see also Liu et al., 2014)

$$\begin{aligned}\frac{\mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})}{\mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})} &= \exp\left(\sum_{1\leq s\leq t\leq p}\left(\theta_{st}^{(1)}-\theta_{st}^{(2)}\right)B_s(x_s)B_t(x_t)-\left(A(\boldsymbol{\theta}^{(1)})-A(\boldsymbol{\theta}^{(2)})\right)\right)\\ &\propto \exp\left(\sum_{1\leq s\leq t\leq p}\theta_{st}B_s(x_s)B_t(x_t)\right),\end{aligned} \tag{4.2}$$

where $\theta_{st}=\theta_{st}^{(1)}-\theta_{st}^{(2)}$, In this setting, it is not necessary to estimate each individual parameter $\theta_{st}^{(1)}$ and $\theta_{st}^{(2)}$. Instead, it suffices to estimate the difference θ_{st}, which is key to identify the changes in the network structure.

To this aim, we propose an approach for learning the structure of a differential network which leverages the estimation of the density ratio between the two distributions, and solves this task in a probabilistic classification setting.

Following Sugiyama et al. (2012), we introduce a binary random variable Y, taking values 0 and 1. Let us assign value 1 to realizations from $\mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})$ and 0 to realizations from $\mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})$, respectively. Then, we can write

$$\mathbf{X}_1 \sim \mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x}) \equiv f(\mathbf{x}\mid Y=1),$$

$$\mathbf{X}_2 \sim \mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x}) \equiv f(\mathbf{x}\mid Y=0).$$

The two densities $\mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})$ and $\mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})$ are written as conditional distributions $f(\mathbf{x}\mid Y=1)$ and $f(\mathbf{x}\mid Y=0)$, respectively. Moreover, $f(\mathbf{x})=f(\mathbf{x}\mid Y=1)p(Y=1)+f(\mathbf{x}\mid Y=0)p(Y=0)$. An application of Bayes' theorem,

$$p(Y|\mathbf{X})f(\mathbf{X})=f(\mathbf{X}|Y)p(Y),$$

yields

$$\frac{p(Y=1 \mid \mathbf{x})f(\mathbf{x})}{p(Y=0 \mid \mathbf{x})f(\mathbf{x})} = \frac{f(\mathbf{x} \mid Y=1)p(Y=1)}{f(\mathbf{x} \mid Y=0)p(Y=0)} = \frac{\mathbb{P}_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})}{\mathbb{P}_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})} \frac{p(Y=1)}{p(Y=0)}. \tag{4.3}$$

Noting that, in practice, the ratio $\dfrac{p(Y=1)}{p(Y=0)}$ can be estimated by the ratio of the number of samples from the two distributions, and combining this with expression (4.2), we obtain

$$\log\left(\frac{p(Y=1 \mid \mathbf{x})}{p(Y=0 \mid \mathbf{x})}\right) = \text{const} + \sum_{1 \le s \le t \le p} \theta_{st} B_s(x_s) B_t(x_t). \tag{4.4}$$

Equation (4.4) takes the form of a logistic regression model. Therefore, we propose to detect changes in the network structure by performing a Bayesian variable selection on the logistic regression model (4.4), where the nonzero coefficients θ_{st} correspond to edges whose associated parameters differ between the two graphical models.

One limitation of this approach is the rapid increase in dimensionality: for a graph consisting of p nodes, the associated logistic regression model requires estimating $p(p+1)/2$ parameters, among which $p(p-1)/2$ are the interaction parameters denoted as θ_{st}. To address this issue, we propose a local structure learning strategy. Specifically, for each $s \in V$, we consider the conditional random variable $X_s \mid \mathbf{x}_{V\setminus\{s\}}$, and perform inference based on its local conditional model. This formulation allows us to decompose the global structure learning problem into a series of smaller, node-wise subproblems, thereby substantially reducing computational complexity.

For each node $s \in V$, we consider conditional models for the conditional random variable $X_s \mid \mathbf{x}_{V\setminus\{s\}}$.

$$\mathbb{P}_{\boldsymbol{\theta}_s}(\mathbf{X}_s = \mathbf{x}_s \mid \mathbf{x}_{V\setminus\{s\}}) \propto \exp\left\{\sum_{t \in V} \theta_{st} B_s(x_s) B_t(x_t)\right\}. \tag{4.5}$$

By applying the argument in (4.3) to the conditional variable $X_s \mid \mathbf{x}_{V\setminus\{s\}}$, we obtain

$$\frac{p(Y=1 \mid x_s; \mathbf{x}_{V\setminus\{s\}})}{p(Y=0 \mid x_s; \mathbf{x}_{V\setminus\{s\}})} = \frac{\mathbb{P}_{\boldsymbol{\theta}_s^{(1)}}(x_s \mid \mathbf{x}_{V\setminus\{s\}})}{\mathbb{P}_{\boldsymbol{\theta}_s^{(2)}}(x_s \mid \mathbf{x}_{V\setminus\{s\}})} \cdot \frac{p(Y=1)}{p(Y=0)}.$$

As stated earlier, the ratio $\frac{p(Y=1)}{p(Y=0)}$ can be estimated from the relative sample sizes of the two groups. Combining this with expression (4.5) and taking the logarithm on both sides, we obtain

$$\log\left(\frac{p(Y=1 \mid x_s; \mathbf{x}_{V\setminus\{s\}})}{p(Y=0 \mid x_s; \mathbf{x}_{V\setminus\{s\}})}\right) = C(-s) + \sum_{t\in V} \left(\theta_{st}^{(1)} - \theta_{st}^{(2)}\right) B_s(x_s) B_t(x_t), \quad (4.6)$$

where $C(-s)$ collects terms that do not depend on x_s.

Thus, for each node s, we obtain a local logistic regression model in which the predictors are $B_s(x_s)B_t(x_t)$, and the regression coefficients are the interaction parameters θ_{st}. Estimating the nonzero components of $\boldsymbol{\theta}_s = \boldsymbol{\theta}_s^{(1)} - \boldsymbol{\theta}_s^{(2)} = \{\theta_{st} : t \in V, \}$ therefore, identifies the neighborhood of node s in the differential graph.

From this stage forward, the process of learning differential network structures was conducted in the following manner. For each node $s \in V$, we first performed a Bayesian variable selection for the logistic regression model given in (4.6). Specifically, we placed a spike-and-slab prior on the parameter vector $\boldsymbol{\theta}_s$ as follows:

$$\begin{aligned} \theta_{st} \mid \gamma_{st} &\overset{\text{prior}}{\sim} (1-\gamma_{st})\,\delta_0 + \gamma_{st} N(0, \sigma^2), \\ \gamma_{st} \mid \omega &\overset{\text{prior}}{\sim} \text{Bernoulli}(\omega), \\ \omega &\overset{\text{prior}}{\sim} \text{Beta}(a, b), \end{aligned}$$

where δ_0 denotes the Dirac measure at 0. Variable selection was then performed by comparing the posterior inclusion probability of γ_{st} with a fixed threshold, typically 0.5. In other words, we considered $\theta_{st} \neq 0$ if $\pi(\gamma_{st} \mid \cdot) \geq 0.5$. Edges were then determined by aggregating the selection results across all local models, where an edge between nodes s and t was included in the differential graph if both the corresponding coefficients θ_{st} and θ_{ts} were identified as nonzero. This procedure is summarized in Algorithm 5.

Notably, the expected value $\text{E}(\omega) = \frac{a}{a+b}$ reflects the prior inclusion probability of a coefficient. Accordingly, the hyperparameter (a, b) may be determined empirically, for instance, based on LASSO regression results, or

specified manually when prior knowledge regarding the number of edges in the differential graph is available. Specifically, in order to determine (a, b) based on LASSO regression, a LASSO variable selection for the logistic regression model (4.6) can be performed. Subsequently, a can be set equal to the number of nonzero coefficients, while b is set equal to the number of zero coefficients obtained from the LASSO estimation.

To perform variable selection, we employed the MFVB approach. This allows for efficient approximate inference while maintaining reasonable accuracy. The implementation was carried out using the `sparsevb` package in `R` (Ray and Szabó, 2022).

Algorithm 5: Local structure learning for differential graphical models

Input: Two sets of independent samples $\{\mathbf{x}_1^{(i)}\}_{i=1}^{n_1} \overset{iid}{\sim} P_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})$; $\{\mathbf{x}_2^{(i)}\}_{i=1}^{n_1} \overset{iid}{\sim} P_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})$.

Output: An estimated differential graph $\hat{G} = (V, \hat{E})$ display the changes in the model parameters from condition 1 to 2.

for $s \in V$ **do**

  Run the stochastic search variable selection for the Logistic regression (4.6)

end for

for $s \in V$ **do**

  for $t \in V$ **do**

    set $(s, t) \in \hat{E}$ if $\theta_{st} \neq 0$ and $\theta_{ts} \neq 0$

  end for

end for

4.3 Simulation studies

4.3.1 Data Generation

In this section, we examine three graph structures: random, scale-free, and hub, with two different numbers of variables, $p = 10$ and $p = 100$, as presented in Chapter 2. In addition, we consider a lattice graph with $p = 100$ nodes. A lattice graph is characterized by its nodes being arranged in a two-dimensional grid structure, which in this case forms a square, as illustrated in Figure 4.1.

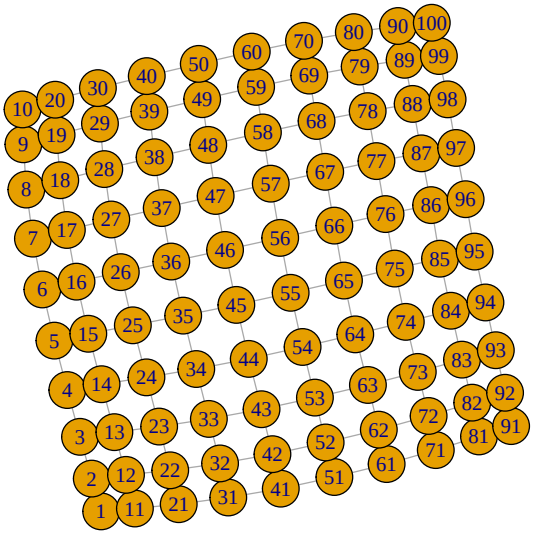


Figure 4.1: Lattice graph structure used in the simulation study.

The samples $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are drawn from the probability distributions $P_{\boldsymbol{\theta}^{(1)}}(\mathbf{x})$ and $P_{\boldsymbol{\theta}^{(2)}}(\mathbf{x})$, which correspond to the graphs $G^{(1)} = (V, E^{(1)})$ and $G^{(2)} = (V, E^{(2)})$, respectively. In this study, we consider two types of graphical models, namely Gaussian graphical models and Poisson graphical models,

which represent continuous and count data, respectively.

Gaussian graphical models

In this case, the joint distributions of the random vectors corresponding to the nodes of the graph follow multivariate normal distributions, i.e.,

$$\mathbf{X}^{(1)} \sim N(0, \mathbf{\Sigma}^{(1)}), \quad \mathbf{X}^{(2)} \sim N(0, \mathbf{\Sigma}^{(2)}),$$

In this setting, the structures of the graphs can be determined through the precision matrices $(\mathbf{\Sigma}^{(1)})^{-1}$ and $(\mathbf{\Sigma}^{(2)})^{-1}$, leading to the differential graph being defined by the number of differing elements between $(\mathbf{\Sigma}^{(1)})^{-1}$ and $(\mathbf{\Sigma}^{(2)})^{-1}$. We consider $(\mathbf{\Sigma}^{(1)})^{-1}$ as a matrix with diagonal elements equal to 2, and off-diagonal elements corresponding to the edges of the graph taking the value 0.4. The matrix $(\mathbf{\Sigma}^{(2)})^{-1}$ is then obtained from $(\mathbf{\Sigma}^{(1)})^{-1}$ by randomly switching the sign to d off-diagonal elements. Consequently, the resulting differential graph contains d edges.

For the graph structures introduced in Chapter 2, we consider $d = 3$ when $p = 10$ and $d = 10$ when $p = 100$. Moreover, we consider also a lattice graph, with $p = 100$ nodes and we examine four scenarios with varying numbers of differential edges, namely $d = 10, 20, 30$, and 90. For each scenario, 500 datasets were generated under three sample size configurations: $n_1 = n_2 = n = 200$, 300, and 400.

Poisson graphical models

For the Poisson scenario, we considered lattice and scale-free graphs with $p = 100$ nodes. In this case, the graph structures are determined by the dependency parameter matrices $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$, meaning that the nonzero off-diagonal elements in these matrices correspond to the edges present in the graphs. We consider the matrix $\boldsymbol{\theta}^{(1)}$ with zeros on the diagonal and off-diagonal elements corresponding to edges in $E^{(1)}$ set to -1, ensuring therefore the existence of a joint distribution. Meanwhile, the matrix $\boldsymbol{\theta}^{(2)}$ was

constructed from $\boldsymbol{\theta}^{(1)}$ by randomly setting to zero $d = 10$ elements, resulting in a differential graph containing 10 edges.

It should be noted that the simulated Poisson data in our study often contained a substantial number of zeros. As a result, certain pairwise interaction terms may consist solely of zeros, potentially impairing the accuracy of the MFVB estimation. To address this limitation, a small random perturbation was introduced into the observations using the `jitter` function available in `R`, after which the algorithm was applied to the adjusted dataset.

For each scenario, 500 datasets were generated under three sample size configurations: $n_1 = n_2 = n = 200$, 300, and 400 for lattice graph and $n_1 = n_2 = n = 500$, 1000, and 1500 for scale-free graph.

4.3.2 Results

Our objective was to evaluate the accuracy of the algorithm in detecting the differential edges between the two conditions 1 and 2. To this end, we considered, as before, two metrics—positive predictive value (PPV) and sensitivity (Se)—computed on the differential graph. The definitions of these metrics were provided in Chapter 2. Throughout this section, we set $\sigma = 1$, and consider two approaches for selecting the hyperparameter (a, b):

i) (a, b) are selected based on the results of the LASSO regression;

ii) (a, b) are chosen according to prior information regarding the expected number of differential edges in the graph.

For the second approach, we suggest a strategy in which (a, b) is determined on the basis of the expected number of significant coefficients. When setting $a+b$ equal to the number of interaction terms that enter local models and computing the probability of edge inclusion as the number $\bar{d}$ of expected edges in the differential graph over the total number of possible edges -given by $p(p-1)/2$-, then a can be set to be equal to $\frac{2\bar{d}(a+b)}{p(p-1)}$. In our simulations, with d known, we set a equal to $\frac{2d(a+b)}{p(p-1)}$.

Tables 4.1, 4.2, and 4.3 present the Monte Carlo averages of True Positives (TP), Positive Predictive Value (PPV), Sensitivity (Se), and computational time for the proposed algorithm under three distinct graph structures, corresponding respectively to $p = 10$, $p = 100$, and the lattice graph with $p = 100$ in the Gaussian graphical models. Table 4.4 summarizes the corresponding results for the Poisson graphical models with lattice and scale-free graph structures, where $p = 100$. The reported computational time corresponds to the runtime for a single dataset. Runtime measurements were performed on an AMD Epyc 7763 2.45GHz processor running R 4.2.1 with 16 cores.

type	n	i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	200	2.697	0.941	0.899	0.092	2.614	0.967	0.871	0.124
	300	2.952	0.952	0.984	0.111	2.932	0.981	0.977	0.109
	400	2.994	0.962	0.998	0.127	2.992	0.980	0.997	0.129
Scale-free	200	2.774	0.923	0.925	0.093	2.734	0.97	0.911	0.091
	300	2.958	0.931	0.986	0.110	2.956	0.969	0.985	0.110
	400	3.000	0.921	1.000	0.126	2.994	0.969	0.998	0.123
Hub	200	2.655	0.940	0.885	0.090	2.571	0.971	0.857	0.093
	300	2.950	0.961	0.983	0.108	2.918	0.984	0.973	0.116
	400	2.992	0.958	0.997	0.129	2.992	0.987	0.997	0.126

Table 4.1: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from three graph structures with $p = 10$ variables with Gaussian graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.7, 9.3)$ for $d = 3$.

For the case $p = 10$ in the Gaussian graphical models, the results reported in Table 4.1 demonstrate the good performance of the algorithm across all three types of graph structures. Most of the differential edges are successfully detected. The hyperparameter selected based on the LASSO regression tend to yield higher TP and Se values compared to the case where the hyperparameter are fixed using prior information; however, the PPV values are slightly lower. For example, in the random graph with a sample size of 200,

type	n	i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	200	7.758	0.820	0.776	3.606	6.074	0.960	0.607	3.646
	300	9.506	0.861	0.951	4.184	8.744	0.980	0.874	4.292
	400	9.926	0.881	0.993	5.112	9.754	0.984	0.975	5.145
Scale-free	200	7.982	0.808	0.798	3.521	6.464	0.967	0.646	3.594
	300	9.670	0.867	0.967	4.198	9.080	0.981	0.908	4.303
	400	9.944	0.888	0.994	5.001	9.818	0.986	0.982	5.187
Hub	200	7.804	0.818	0.780	3.518	6.020	0.966	0.602	3.645
	300	9.500	0.867	0.950	4.187	8.738	0.981	0.874	4.362
	400	9.944	0.879	0.994	5.015	9.724	0.982	0.972	4.947

Table 4.2: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from three graph structures with $p = 100$ variables with Gaussian graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$.

the TP and Se values for case (i) are 2.697 and 0.899, compared to 2.614 and 0.871 for case (ii). The corresponding PPV values are 0.941 and 0.967, respectively. Although some differences are observed, both approaches maintain high levels of performance as the sample size increases. Specifically, for the random graph with a sample size of 400, the PPV and Se values for the two approaches are 0.962, 0.998 and 0.980, 0.997, respectively, all of which remain at high levels.

For the case $p = 100$ in the Gaussian graphical models, the results presented in Table 4.2 show a similar phenomenon to that observed for $p = 10$. The TP and Se values in case (i) remain higher than those in case (ii), while the PPV values are lower. However, the differences are more pronounced in this setting due to the substantially higher dimensionality. For example, in the hub graph with a sample size of 400, the TP and Se values under the hyperparameter setting (i) are 9.944 and 0.994, whereas for case (ii) they are 9.724 and 0.972, indicating only minor differences. In contrast, the PPV values differ more noticeably, with 0.879 for case (i) and 0.982 for case (ii).

d	n	i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
10	200	8.268	0.837	0.827	3.544	6.560	0.970	0.656	3.704
	300	9.690	0.876	0.969	4.205	9.086	0.983	0.909	4.378
	400	9.964	0.889	0.996	5.021	9.876	0.984	0.988	5.173
20	200	16.404	0.900	0.820	3.524	14.402	0.971	0.720	3.636
	300	19.434	0.926	0.972	4.212	18.768	0.984	0.938	4.404
	400	19.900	0.934	0.995	5.048	19.768	0.987	0.988	5.283
30	200	24.708	0.927	0.824	3.552	21.976	0.974	0.733	3.675
	300	29.002	0.945	0.967	4.167	28.014	0.985	0.934	4.292
	400	29.838	0.949	0.995	5.087	29.596	0.988	0.987	5.167
90	200	70.424	0.968	0.782	3.612	65.350	0.985	0.726	3.715
	300	85.234	0.980	0.947	4.237	82.784	0.992	0.920	4.305
	400	89.078	0.982	0.990	5.100	88.312	0.993	0.981	5.286

Table 4.3: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from lattice graph with $p = 100$ variables with Gaussian graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$; $(a, b) = (0.4, 99.6)$ for $d = 20$; $(a, b) = (0.6, 99.4)$ for $d = 30$; $(a, b) = (1.8, 98.2)$ for $d = 90$.

This result suggests that incorporating prior information into the model can improve the algorithm's performance, particularly when the sample size is small.

For the case of the lattice graph with $p = 100$ in the Gaussian graphical models, the results presented in Table 4.3 further demonstrate the good performance of the algorithm across different values of the number of differential edges d. Once again, the TP and Se values in case (i) are higher than those in case (ii), while the PPV values are lower. Moreover, the number of differential edges also influences the algorithm's performance under different choices of hyperparameter, particularly when the sample size is small. Specifically, for $d = 90$ with a sample size of 200, selecting the hyperparameter via LASSO regression allows the algorithm to detect approximately 70

type	n	i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Lattice	200	6.554	0.994	0.655	12.020	4.648	0.996	0.465	12.037
	300	8.954	0.996	0.895	11.224	7.146	0.999	0.715	11.194
	400	9.772	0.998	0.977	11.432	8.892	0.999	0.889	11.673
Scale-free	500	8.168	0.987	0.817	11.308	6.932	0.995	0.693	11.357
	1000	8.984	0.984	0.898	19.020	8.754	0.995	0.875	17.803
	1500	9.316	0.973	0.932	26.067	9.058	0.997	0.906	25.307

Table 4.4: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from lattice graph and $n = 500,\ 1000,\ 1500$ from scale-free graph with $p = 100$ variables with Poisson graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$.

differential edges, whereas using approach (ii) detects only about 65 edges. This difference becomes less pronounced as the sample size increases. For instance, with a sample size of 400, the TP, PPV, and Se values are, respectively, (i) 89.078, 0.982, and 0.990, compared to (ii) 88.312, 0.993, and 0.981. These findings once again indicate that the algorithm can be improved by incorporating prior knowledge into the model.

Whereas the results obtained from the Gaussian graphical models indicate the robustness of the proposed algorithm with respect to different graph structures and varying numbers of differential edges, those derived from the Poisson graphical models, as reported in Table 4.4, reveal several noteworthy observations. For the lattice graph, the algorithm exhibits relatively poorer performance compared with the Gaussian graphical model results reported in Table 4.3, particularly in terms of True Positives (TP) and Sensitivity (Se), while still achieving a consistently high Positive Predictive Value (PPV). In particular, for $n = 200$, the TP and Se values in case (i) obtained from the Poisson graphical model are 6.554 and 0.655, respectively, compared to 8.268 and 0.827 for the Gaussian graphical model, with the corresponding PPV values being 0.994 and 0.837. The results also reveal a dependence on

the choice of tuning parameters, similar to that observed for the Gaussian graphical models. A noteworthy finding is that, in this scenario, the performance of the proposed algorithm exhibits a pronounced dependence on the underlying graph structure. Specifically, the algorithm achieves satisfactory performance on the lattice graph even with moderately small sample sizes, whereas substantially larger samples are required for the scale-free graph to attain comparable accuracy. Under the hyperparameter setting (i), the TP, PPV, and Se values for the lattice graph attain high levels at a sample size of $n = 400$, while for the scale-free graph, a sample size of $n = 1500$ is necessary.

In terms of computational efficiency, the use of the MFVB method substantially optimizes the running time of the algorithm, even in high-dimensional scenarios. Although the choice of hyperparameter has a certain influence on the computational time, the differences remain relatively minor. Furthermore, the running time of the algorithm for the Poisson graphical models is considerably higher than that for the Gaussian graphical models, reflecting the increased computational complexity associated with handling count data. Interestingly, in some certain cases, the algorithm exhibits shorter running times at sample sizes of $n = 300$ and $n = 400$ than at $n = 200$. As elaborated in Chapter 3, this counterintuitive behavior can be attributed to the fact that larger sample sizes facilitate faster convergence of the MFVB algorithm.

Following one reviewer's suggestion, we considered controlling the false discovery rate (FDR) when selecting variables based on posterior inclusion probabilities, instead of using a fixed threshold, which was set to 0.5 in our initial analysis.

We observed that, in some scenarios, selecting the hyperparameters (a, b) via Lasso regression resulted in relatively low positive predictive values (PPV). To address this issue, we therefore implemented an FDR control procedure following Müller et al. (2007) to explicitly regulate the proportion of false positives when the hyperparameters (a, b) were chosen using Lasso. Specifically, rather than fixing the inclusion threshold at 0.5, we selected a datadriven threshold such that the expected FDR was controlled at a prespecified level

of 0.05, while simultaneously minimizing the expected false negative rate (FNR).

In our setting, let $d_i \in \{0, 1\}$ denote the decision of whether the ith regression coefficient is nonzero, and let v_i denote the corresponding posterior inclusion probability. The expected false discovery rate (FDR) and false negative rate (FNR), conditional on the observed data $\boldsymbol{y}$, are then defined as

$$\mathrm{E}(\mathrm{FDR} \mid \boldsymbol{y}) = \frac{\sum_i (1 - v_i)\, d_i}{\sum_i d_i + \epsilon},$$
$$\mathrm{E}(\mathrm{FNR} \mid \boldsymbol{y}) = \frac{\sum_i (1 - d_i)\, v_i}{p - \sum_i d_i + \epsilon},$$

where the small constant ϵ is introduced to ensure that the denominators are nonzero.

d	n	LASSO with FDR control				i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
10	200	6.762	0.975	0.676	3.611	8.268	0.837	0.827	3.544	6.560	0.970	0.656	3.704
	300	9.230	0.987	0.923	4.279	9.690	0.876	0.969	4.205	9.086	0.983	0.909	4.378
	400	9.890	0.989	0.989	5.187	9.964	0.889	0.996	5.021	9.876	0.984	0.988	5.173
20	200	13.056	0.985	0.653	3.572	16.404	0.900	0.820	3.524	14.402	0.971	0.720	3.636
	300	18.346	0.991	0.917	4.272	19.434	0.926	0.972	4.212	18.768	0.984	0.938	4.404
	400	19.714	0.993	0.986	5.132	19.900	0.934	0.995	5.048	19.768	0.987	0.988	5.283
30	200	20.448	0.983	0.685	3.586	24.708	0.927	0.824	3.552	21.976	0.974	0.733	3.675
	300	27.690	0.991	0.923	4.280	29.002	0.945	0.967	4.167	28.014	0.985	0.934	4.292
	400	29.574	0.992	0.986	5.167	29.838	0.949	0.995	5.087	29.596	0.988	0.987	5.167
90	200	60.906	0.988	0.677	3.634	70.424	0.968	0.782	3.612	65.350	0.985	0.726	3.715
	300	82.212	0.993	0.913	4.223	85.234	0.980	0.947	4.237	82.784	0.992	0.920	4.305
	400	88.394	0.994	0.982	5.050	89.078	0.982	0.990	5.100	88.312	0.993	0.981	5.286

Table 4.5: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from lattice graph with $p = 100$ variables with Gaussian graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$; $(a, b) = (0.4, 99.6)$ for $d = 20$; $(a, b) = (0.6, 99.4)$ for $d = 30$; $(a, b) = (1.8, 98.2)$ for $d = 90$.

The results obtained by applying this procedure are reported in Tables 4.5, 4.6, and 4.7. For ease of comparison, we also include in these tables the corresponding results obtained without applying FDR control. As expected, the results show a substantial improvement in PPV when the FDR

type	n	LASSO with FDR control				i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Random	200	5.796	0.972	0.580	4.609	7.758	0.820	0.776	3.606	6.074	0.960	0.607	3.646
	300	8.624	0.986	0.862	4.226	9.506	0.861	0.951	4.184	8.744	0.980	0.874	4.292
	400	9.728	0.989	0.973	4.942	9.926	0.881	0.993	5.112	9.754	0.984	0.975	5.145
Scale-free	200	6.248	0.973	0.625	3.410	7.982	0.808	0.798	3.521	6.464	0.967	0.646	3.594
	300	9.012	0.987	0.901	4.244	9.670	0.867	0.967	4.198	9.080	0.981	0.908	4.303
	400	9.800	0.990	0.980	5.123	9.944	0.888	0.994	5.001	9.818	0.986	0.982	5.187
Hub	200	5.856	0.974	0.586	3.397	7.804	0.818	0.780	3.518	6.020	0.966	0.602	3.645
	300	8.746	0.987	0.875	4.242	9.500	0.867	0.950	4.187	8.738	0.981	0.874	4.362
	400	9.720	0.989	0.972	4.973	9.944	0.879	0.994	5.015	9.724	0.982	0.972	4.947

Table 4.6: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from three graph structures with $p = 100$ variables with Gaussian graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$.

type	n	LASSO with FDR control				i)				ii)			
		TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)	TP	PPV	Se	Time(s)
Lattice	200	5.924	0.996	0.592	12.017	6.554	0.994	0.655	12.020	4.648	0.996	0.465	12.037
	300	8.614	0.997	0.861	11.131	8.954	0.996	0.895	11.224	7.146	0.999	0.715	11.194
	400	9.688	0.999	0.969	11.501	9.772	0.998	0.977	11.432	8.892	0.999	0.889	11.673
Scale-free	500	8.026	0.989	0.803	11.406	8.168	0.987	0.817	11.308	6.932	0.995	0.693	11.357
	1000	8.970	0.987	0.897	17.621	8.984	0.984	0.898	19.020	8.754	0.995	0.875	17.803
	1500	9.036	0.964	0.904	23.847	9.316	0.973	0.932	26.067	9.058	0.997	0.906	25.307

Table 4.7: Monte Carlo means of TP, PPV, Se and runtime obtained by simulating 500 data sets with three sample sizes, $n = 200,\ 300,\ 400$ from lattice graph and $n = 500,\ 1000,\ 1500$ from scale-free graph with $p = 100$ variables with Poisson graphical models using $\sigma = 1$, (a, b) is chosen by: i) LASSO and ii) $(a, b) = (0.2, 99.8)$ for $d = 10$.

control procedure is applied. For example, in Table 4.6, for the scale-free graph with sample size $n = 200$, the PPV increases from 0.808 without FDR control to 0.973 with FDR control. As anticipated, this improvement comes at the cost of slightly lower TP and sensitivity (Se), which decrease from 7.982 and 0.798 to 6.248 and 0.625, respectively. However, this trade off becomes negligible for larger sample sizes. In particular, when $n = 400$, the TP and Se values with FDR control (9.800 and 0.980) are very close to those obtained without FDR control (9.944 and 0.994), while the PPV remains substantially higher (0.990 compared to 0.888). Notably, the performance obtained under FDR control closely matches that achieved when the hyperparameters are chosen based on prior knowledge.

4.4 Real data analysis

In this section, we reanalyze the dataset previously used by Tu et al. (2021) to estimate the differential network between two breast cancer subtypes: luminal A and basal-like. Luminal A breast cancer is hormone receptor positive, whereas basal-like breast cancer is hormone receptor negative. The basal-like subtype is typically more aggressive and associated with a poorer prognosis than luminal A breast cancer (Schnitt, 2010). Identifying alterations in gene network connectivity between the different cancer subtypes may provide valuable insights into the molecular mechanisms underlying breast cancer.

The gene expression datasets were downloaded from The Cancer Genome Atlas (TCGA, level 3, Agilent G450 microarray, version dated May 6, 2017) (Network, 2012). The final dataset includes 231 luminal A samples and 95 basal-like samples. We consider 189 genes, 139 of which belonging to the breast cancer pathway (hsa05224), obtained from the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa and Goto, 2000). The dataset also includes 50 highly variable genes with no significant differential expression between the two conditions and not included in the breast cancer pathway.

The differential network obtained by applying the algorithm using the

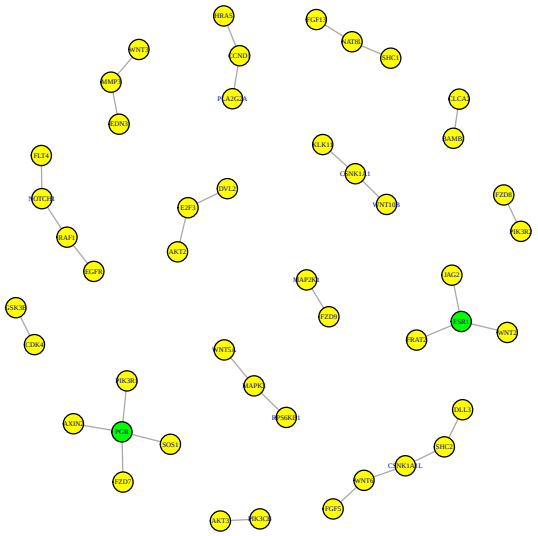


Figure 4.2: Differential network estimated by the algorithm from breast cancer gene expression datasets (hub nodes colored green).

LASSO strategy to the breast cancer gene expression datasets is illustrated in Figure 4.2. For clarity, the figure displays only nodes that are connected. The network consists of 31 edges connecting 46 genes, of which 38 are differentially expressed and 8 are nondifferentially expressed.

In this case, the true structure of the differential network between the luminal A and basal-like subtypes is unknown; therefore, the accuracy of the algorithm cannot be assessed by comparison with a reference network. To evaluate the algorithm's performance, we focus instead on the functional analysis of the central nodes in the differential network, which are expected to play an important role in distinguishing between the luminal A and basal-like subtypes. Most nodes in the differential network have one or two edges, while only two nodes, *ESR1* with three edges and *PGR* with four edges,

exhibit higher connectivity and are identified as hub nodes.

The genes *ESR1* (encoding estrogen receptor alpha, ERα) and *PGR* (encoding progesterone receptor, PR) play a pivotal role in distinguishing the molecular subtypes of breast cancer, particularly between the luminal A and basal-like subtypes. Luminal A tumors typically exhibit strong expression of *ESR1* and *PGR*, reflecting a hormone receptor–positive phenotype, whereas basal-like tumors usually lack expression of these genes, corresponding to a hormone receptor–negative phenotype (Eliyatkın et al., 2015). Therefore, the differential expression of *ESR1* and *PGR* serves as a crucial molecular signature for classifying luminal A and basal-like breast cancer subtypes.

Motivated by one reviewer's comment, we additionally compare our proposed algorithm with the six frequentist methods introduced in Plaksienko et al. (2025). Figure 4.3 illustrates the results obtained by applying our

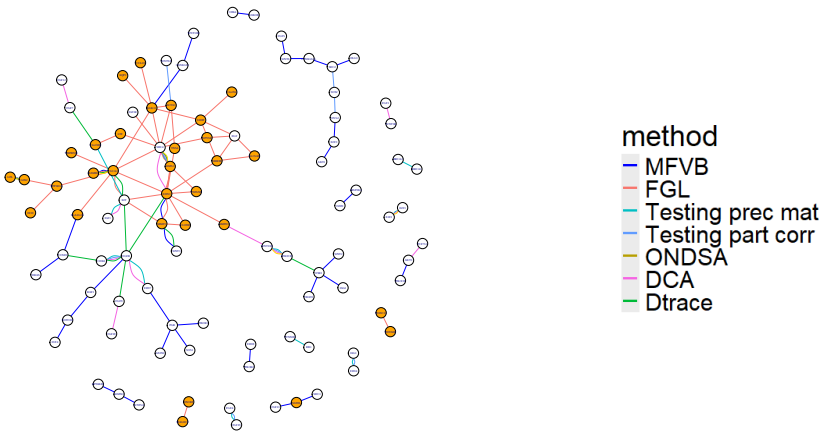


Figure 4.3: Differential network estimated using the various methods (see Plaksienko et al. (2025) for acronym definitions). Edge multiplicity indicates the number of methods that identified a given edge, while edge color denotes the specific method.

proposed algorithm alongside the six frequentist methods reviewed in Plak-

sienko et al. (2025). For clarity, the figure displays only nodes that are connected. Genes not belonging to the breast cancer pathway are shown in orange. Several noteworthy observations can be drawn from the results. The links identified by the fused graphical lasso (FGL) method are predominantly between genes that are not part of the pathway, indicating poor performance of the algorithm, as also noted in Plaksienko et al. (2025).

Notably, several biologically important interactions were identified exclusively by our proposed algorithm. For example, our method detected an interaction between *EGFR* and *RAF1*. The *EGFR-RAF1* (c-Raf) axis is known to play a critical role in both diagnosis and prognosis in breast cancer. In basal-like carcinomas, *EGFR*-mediated activation of *RAF1* leads to strong phosphorylation of *MEK* and *ERK*, thereby promoting aggressive cell proliferation and the invasive phenotype characteristic of this subtype. In contrast, in luminal A tumors, the *EGFR-RAF1* pathway is typically inactive; however, its aberrant activation has been associated with the development of endocrine resistance, where *RAF1* can bypass estrogen receptor signaling to promote cell survival (Wu et al., 2014; Hoadley et al., 2007). Another example concerns interactions involving *ESR1*, which, as discussed earlier, play a central role in distinguishing between luminal A and basal-like breast cancer subtypes. Notably, these *ESR1*-related interactions were identified exclusively by our proposed algorithm. This further suggests that our method is capable of uncovering biologically meaningful associations that are subtle and not easily detected by competing approaches. Finally, our algorithm identified substantially fewer edges involving genes outside the pathway of interest (nodes highlighted in orange) compared to the competing methods. This behavior is consistent with biological expectations and suggests improved specificity in detecting biologically relevant interactions.

4.5 Conclusion

In this chapter, we provided an overview of the structure learning problem for differential networks, which was modeled through the density ratio between two distributions. We proposed to learn the structure of the differential graph by linking the density ratio to a logistic regression model, thereby reformulating the structure learning task as a variable selection problem in logistic regression. One major challenge of this approach lay in the rapid increase in dimensionality. To address this issue, we proposed a local structure learning strategy, in which instead of considering the ratio between two joint densities, we examined the conditional densities of each node given all the others. This decomposition transformed the global problem into smaller subproblems, effectively mitigating the curse of dimensionality in logistic regression. The variable selection problem was then solved using the MFVB approach, which substantially reduced computational time while maintaining high estimation accuracy.

The results of the simulation studies demonstrated the effectiveness of the proposed algorithm across various graph structures, with highly competitive computational efficiency. Moreover, investigating different hyperparameter settings highlighted their influence on the algorithm's performance, indicating that incorporating prior knowledge into the model could substantially enhance its effectiveness. This feature was particularly beneficial in real-world applications, where the number of observations was limited but some prior information—such as biological interactions—was available.

The analysis of the real breast cancer gene expression data further emphasized the practical utility of the proposed method. The identified differential network between the luminal A and basal-like subtypes highlighted several key genes that played important roles in distinguishing between these two molecular subtypes. Analyzing real data not only validated the effectiveness of the proposed algorithm beyond simulation settings but also provided meaningful biological insights into the underlying molecular mechanisms of breast cancer heterogeneity. By uncovering changes in gene–gene interac-

tions between different subtypes, differential network analysis offered a complementary perspective to traditional differential expression analysis. This demonstrated the potential of the proposed framework as a valuable tool for exploring complex biological systems and generating hypotheses for future experimental studies.

Chapter 5

Conclusions

Structure learning of undirected graphical models is known to be an NP-hard problem. This computational intractability arises from the need to search over an exponential number of possible graph configurations, as the number of potential edges grows quadratically with the number of nodes. In Chapters 2 and 3 of this dissertation, we addressed the problem in the context of count data. In addition, we considered a potential extension related to structure learning for differential networks in Chapter 4. Several open problems remain for future investigation. In what follows, we summarize the main findings and outline possible research directions that can be further developed based on the results of this dissertation.

Firstly, we proposed a new model specification for count data that overcame certain limitations regarding the dependence constraints among parameters and reduced the model complexity compared to classical formulations. We showed that this specification leads to a joint distribution that is Markov with respect to the graph for certain classes of graphs. Future research could focus on identifying additional classes of graphs that satisfy this property. Furthermore, we believe that this model specification provides a useful framework for handling high dimensional problem by investigating optimal node orderings in the model formulation. Future work may focus on determining such optimal orderings by leveraging prior knowledge.

Secondly, we performed structure learning by solving variable selection problems for local Poisson regression models within a Bayesian framework based on posterior inference. Our initial approach relied on MCMC sampling, which achieved high predictive accuracy but involved substantial computational costs. To address this limitation, we proposed to adopt the MFVB approach, which provided competitive computational efficiency while maintaining comparable estimation accuracy. A comparison of the simulation results in Chapters 2 and 3 showed that the MFVB approach markedly outperformed MCMC in terms of computational time, with only marginal differences in accuracy, particularly for large sample sizes. Future research could extend this framework to mixed graphical models, where heterogeneous variable types—such as Gaussian, Poisson, and binary—are modeled through their respective distributions.

Thirdly, we briefly discussed the problem of structure learning for differential graphs and proposed an approach that formulates this task as a variable selection problem in a logistic regression model linked to the density ratio. We addressed the issue of rapidly increasing dimensionality—an inherent limitation of this approach—by introducing a local structure learning framework, in which, instead of considering the density ratio between two joint distributions, we focused on the ratios between conditional distributions. The simulation results on both Gaussian and Poisson graphical models, despite certain limitations with count data, have demonstrated the effectiveness of the proposed method. Nevertheless, the approach can be readily extended to any member of the exponential family of distributions. A promising direction for future research is to apply this method to mixed graphical models, in which variables may follow different types of distributions. In addition, another potential line of investigation involves optimizing the algorithm for applications with count data.

Throughout Chapters 2, 3, and 4, we also discussed how the performance of the proposed algorithms is influenced by the choice of hyperparameter. While this characteristic reflects an advantage of Bayesian ap-

proaches—allowing prior information to be incorporated into the model via hyperparameter—it also poses a challenge when such prior information is unavailable, making appropriate hyperparameter selection nontrivial. Future research could therefore focus on developing more principled strategies for selecting hyperparameter, as well as exploring their potential role as mechanisms for controlling the sparsity level of the learned graph structures.

Appendix A

A.1

Here, we report the sampler used in Chain-LPGM and Bayes-LPGM.

- Choose $a_\omega = b_\omega = 1$, and fix some values for $a_\tau,\ b_\tau$.
- Set initial values for $\tau_{st}^{2(0)}, \gamma_{st}^{(0)}, \omega^{(0)}$ by sampling from the prior distributions. For convenience, set an initial value $\boldsymbol{\theta}_A^{(0)}$ equal to the estimates of the Poisson regression parameters obtained, for example, via iterative weighted least squares (IWLS).
- For $r = 1, 2, ..., R$
  - For each regressor $t \in A$ in (2.6) a new value θ_{st}^p is proposed by drawing a random number from a Gaussian proposal distribution $q(\theta_{st}^p \mid \theta_{st}^{(r-1)})$ with variance and mean:

$$v_t = \left(\mathbf{x}_t^\top W \mathbf{x}_t + \frac{1}{\gamma_{st}^{(r-1)} \tau_{st}^{2(r-1)}} \right)^{-1} \qquad m_t = v_t \cdot \mathbf{x}_t^\top W (\tilde{\mathbf{x}}_s - \tilde{\eta}). \tag{A.1}$$

  Here, $W = W(\boldsymbol{\theta}_A^{(r-1)}) = \text{diag}(w_1, ..., w_n)$ is the weight matrix for IWLS with $w_i = w_i(\boldsymbol{\theta}_A^{(r-1)}) = \exp\{\langle \boldsymbol{\theta}_A^{(r-1)}, \mathbf{x}_A^{(i)} \rangle\}$. The vector $\tilde{\mathbf{x}}_s = (\tilde{x}_{1s}, \ldots, \tilde{x}_{ns})^\top$ contains the working observations, defined as:

$$\tilde{x}_{is} = \tilde{x}_{is}(\theta_{st}^{(r-1)}) = \log(w_i) + \frac{(x_{is} - w_i)}{w_i}.$$

Finally, $\tilde{\eta} = (\tilde{\eta}_1, \ldots, \tilde{\eta}_n)^\top$ with $\tilde{\eta}_i = \exp\{\langle \boldsymbol{\theta}^{(r-1)}_{A\setminus\{t\}}, \mathbf{x}^{(i)}_{A\setminus\{t\}} \rangle\}$ where $\boldsymbol{\theta}^{(r-1)}_{A\setminus\{t\}} = \{\theta^{(r-1)}_{sj},\ j \in A, j \neq t\}$ is the part of the predictor associated with all effects in the model but the t-th one.

Let $\boldsymbol{\theta}^p_A$ be the vector $\boldsymbol{\theta}^{(r-1)}_A$ where the t-th element $\theta^{(r-1)}_{st}$ is replaced by θ^p_{st}. Denote $L(\boldsymbol{\theta}^k_A) = \prod_{i=1}^n \mathbb{P}_{\boldsymbol{\theta}^k_A}(x_{is} \mid \mathbf{X}^{(i)}_A = \mathbf{x}^{(i)}_A)$ with $k \in \{(r-1), p\}$ and $\mathbb{P}$ as in (2.4) and (2.2) respectively.

Compute the acceptance probability:

$$\alpha = \frac{L(\boldsymbol{\theta}^p_A)\pi(\theta^p_{st})q(\theta^{(r-1)}_{st} \mid \theta^p_{st})}{L(\boldsymbol{\theta}^{(r-1)}_A)\pi(\theta^{(r-1)}_{st})q(\theta^p_{st} \mid \theta^{(r-1)}_{st})}.$$

In $\boldsymbol{\theta}^{(r)}_A$, set $\theta^{(r)}_{st} = \theta^p_{st}$ with probability $\min(1, \alpha)$.

– Sample $\tau^{2(r)}_{st}$ from the full conditional distribution:

$$\tau^{2(r)}_{st} \mid . \sim \Gamma^{-1}\left(a_\tau + 1/2, b_\tau + \frac{\theta^{2(r-1)}_{st}}{2\gamma^{(r-1)}_{st}}\right).$$

– Sample $\gamma^{(r)}_{st}$ from the odds ratio:

$$\frac{p(\gamma^{(r)}_{st} = 1 \mid .)}{p(\gamma^{(r)}_{st} = v_0 \mid .)} = v_0^{1/2}\frac{\omega^{(r-1)}}{1-\omega^{(r-1)}}\exp\left(\frac{1-v_0}{2v_0}\frac{\theta^{2(r)}_{st}}{\tau^{2(r)}_{st}}\right).$$

– Sample $\omega^{(r)}$ from the full conditional distribution:

$$\omega^{(r)} \mid . \sim \text{Beta}\left(a_\omega + I_1(\gamma^{(r)}_{st}), b_\omega + I_{v_0}(\gamma^{(r)}_{st})\right).$$

A.2

Here, we present the results of the real data analysis under the hyperparameter setting $(a_\tau, b_\tau) = (3, 50)$. The reconstruction results obtained using Bayes-LPGM and Chain-LPGM are visualized in Figures A.1 and A.2, respectively.

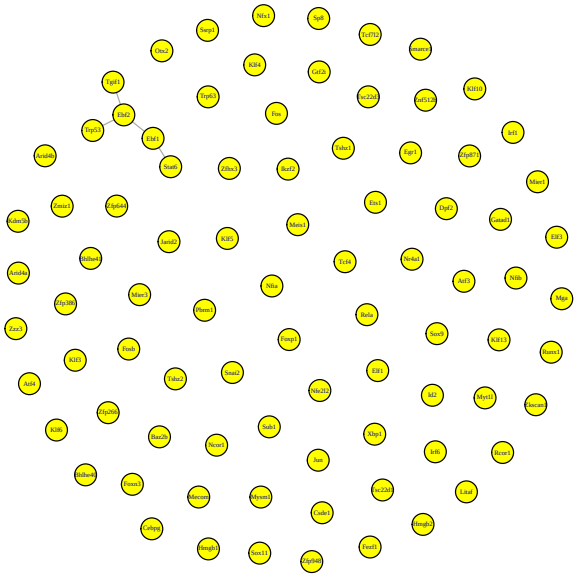


Figure A.1: Olfactory Neuron gene network estimated by the algorithm using Bayes-LPGM with $(a_\tau, b_\tau) = (3, 50)$ and $v_0 = 0.00025$.

A.3

Here, we focus on the scenario where the true value of θ_{st} is close to zero, and investigate the sampling behavior of τ^2_{st} and γ_{st} during the early iterations of the Markov chain. This analysis clearly highlights the limitation of the variable selection process in accurately identifying edges when the true parameter values are very close to zero. Recall that at each iteration, τ^2_{st} and γ_{st} are sampled as follows:

- Sample $\tau^{2(r)}_{st}$ from the full conditional distribution:

$$\tau^{2(r)}_{st} \mid . \sim \Gamma^{-1}\left(a_\tau + 1/2, b_\tau + \frac{\theta^{2(r-1)}_{st}}{2\gamma^{(r-1)}_{st}}\right).$$

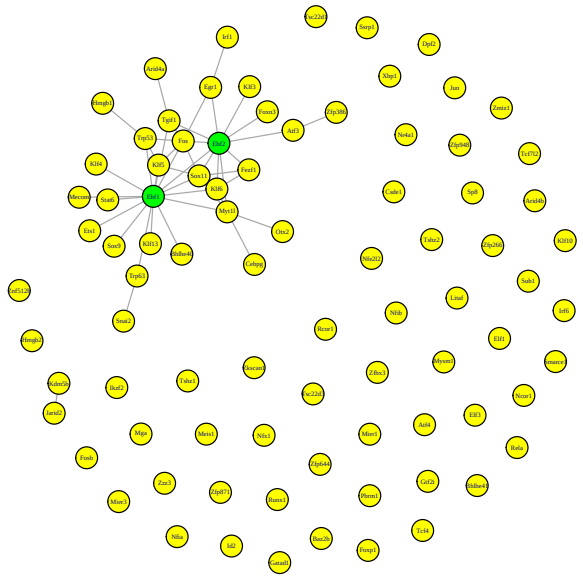


Figure A.2: Olfactory Neuron gene network estimated by the algorithm using Chain-LPGM with $(a_\tau, b_\tau) = (3, 50)$ and $v_0 = 0.00025$ (hub nodes coloured green).

- Sample $\gamma_{st}^{(r)}$ from the odds ratio:

$$\frac{p(\gamma_{st}^{(r)} = 1 \mid .)}{p(\gamma_{st}^{(r)} = v_0 \mid .)} = v_0^{1/2} \frac{\omega^{(r-1)}}{1 - \omega^{(r-1)}} \exp\left(\frac{1 - v_0}{2v_0} \frac{\theta_{st}^{2(r)}}{\tau_{st}^{2(r)}}\right).$$

Assume that at iteration $r = 0$ we initialize $\gamma_{st}^{(0)} = v_0$ and $\omega^{(0)} = 0.5$. Consider the true value $\theta_{st}^2 = v_0 \neq 0$, then the initial value $\theta_{st}^{(0)}$ estimated using IWLS is close to v_0. Then

$$\tau_{st}^{(1)} \mid . \sim \Gamma^{-1}\left(a_\tau + 1/2, b_\tau + \frac{\theta_{st}^{2(0)}}{2\gamma_{st}^{(0)}}\right) \approx \Gamma^{-1}\left(a_\tau + 1/2, b_\tau + \frac{1}{2}\right).$$

We consider $p(\gamma_{st}^{(1)} = v_0 \mid .) < 0.75$ or:

$$\frac{p(\gamma_{st}^{(1)} = 1 \mid .)}{p(\gamma_{st}^{(1)} = v_0 \mid .)} = v_0^{1/2} \exp\left(\frac{1 - v_0}{2v_0} \frac{\theta_{st}^{2(1)}}{\tau_{st}^{2(1)}}\right) > \frac{1}{3}.$$

This implies: $\tau_{st}^{2(1)} < \dfrac{(1 - v_0)\theta_{st}^{2(1)}}{2v_0\left(\log\left(\dfrac{1}{3}\right) - \dfrac{\log(v_0)}{2}\right)} \approx 0.164 \times \dfrac{\theta_{st}^{2(1)}}{v_0}$ with $v_0 =$ 0.00025. Given the true value of $\theta_{st}^2 = v_0$, it is reasonable to assume that $\dfrac{\theta_{st}^{2(1)}}{v_0} < 10$, which implies $\tau_{st}^{2(1)} < 1.64$. However, such a small value of $\tau_{st}^{2(1)}$ is highly unlikely under the chosen hyperparameter $(a_\tau, b_\tau) = (3, 50)$, since the inverse-gamma distribution $\Gamma^{-1}(3.5, 50.5)$ places negligible probability mass below 1.64. Consequently, the probability $p(\gamma_{st}^{(1)} = v_0 \mid \cdot)$ is typically greater than 0.75, indicating that $\gamma_{st}^{(1)}$ is likely to remain equal to v_0. This behavior tends to persist across iterations of the Markov chain, thereby impeding the effective identification of the active (nonzero) regression coefficients. To illustrate this behavior, we conducted a simulation study for the case $p = 10$, using the same graph structures as in the previous simulation, but assigning active coefficients values close to zero (specifically, -0.1). The results presented in Table A.1 highlight the difficulty in accurately identifying active coefficients, even when the sample size is large, as previously discussed.

Type	n	TP	PPV	Se
Random	100	0.660	0.574	0.066
	200	1.027	0.790	0.103
	500	1.125	0.988	0.113
Scale-free	100	0.637	0.563	0.064
	200	0.914	0.755	0.091
	500	1.020	1.000	0.102
Hub	100	0.542	0.447	0.068
	200	0.920	0.780	0.115
	500	1.041	1.000	0.130

Table A.1: Monte Carlo means of TP, PPV, Se obtained by simulating 500 data sets with three sample sizes, $n = 100,\ 200,\ 500$ from graph with $p = 10$ variables with Poisson node conditional distribution with $(a_\tau, b_\tau) = (3, 50),\ v_0 = 0.00025$ and $\theta_{st} = -0.1, \forall\ (s,t) \in E$ using Bayes-LPGM.

Bibliography

Allen, G. and Liu, Z. (2013). A local Poisson graphical model for inferring networks from sequencing data. *NanoBioscience, IEEE Transactions*, 12(3):189–198.

Banerjee, S. and Ghosal, S. (2015). Bayesian structure learning in graphical models. *J. Multivariate Anal.*, 136:147–162.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236.

Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *J. Roy. Statist. Soc. Ser. B*, 78(5):1103–1130.

Bissiri, P. G. and Walker, S. G. (2010). On Bayesian learning from Bernoulli observations. *J. Statist. Plann. Inference*, 140(11):3520–3530.

Bissiri, P. G. and Walker, S. G. (2012). Converting information into probability measures with the Kullback–Leibler divergence. *Ann Inst Stat Math*, 64(6):1139–1160.

Bissiri, P. G. and Walker, S. G. (2019). On general Bayesian inference using loss functions. *Statist. Probab. Lett.*, 152:89–91.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational in-

ference: A review for statisticians. *Journal of the American Statistical Association*, 112:859 – 877.

Carvalho, C. M., Massam, H., and West, M. (2007). Simulation of hyper-inverse wishart distributions in graphical models. *Biometrika*, 94(3):647–659.

Chickering, D. M., Heckerman, D. E., and Meek, C. (2004). Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330.

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Chuang, S. M., Wang, Y., Wang, Q., Liu, K.-M., and Shen, Q. (2012). Ebf2 Marks Early Cortical Neurogenesis and Regulates the Generation of Cajal-Retzius Neurons in the Developing Cerebral Cortex. *Developmental Neuroscience*, 33(6):479–493.

Cooper, G. F. and Herskovits, E. H. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347.

Dawid, A. P. and Lauritzen, S. L. (1993). Hyper markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics*, 21(3):1272–1317.

Drton, M. and Maathuis, M. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393.

Eliyatkın, N., Yalçın, E., Zengel, B., and Vardar, E. (2015). Molecular classification of breast carcinoma: From traditional, old-fashioned way to a new age, and a new way. *The journal of breast health*, 11(2):59–66.

Erdös, P. and Rényi, A. (1959). On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297.

Fabian Scheipl, L. F. and Kneib, T. (2012). Spike-and-slab priors for function selection in structured additive regression models. *Journal of the American Statistical Association*, 107(500):1518–1532.

Fan, L. (2007). *Structure learning with large sparse undirected graphs and its applications*. PhD thesis, School of Computer Science, USA.

Gadye, L., Das, D., Sanchez, M. A., Street, K., Baudhuin, A., Wagner, A., Cole, M. B., Choi, Y. G., Yosef, N., Purdom, E., Dudoit, S., Risso, D., Ngai, J., and Fletcher, R. B. (2017). Injury activates transient olfactory stem cell states with diverse lineage capacities. *Cell Stem Cell*, 21(6):775–790.

Gardner, T. S., di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science (New York, N.Y.)*, 301(5629):102–105.

Garel, S., Marín, F., Mattéi, M.-G., Vesque, C., Vincent, A., and Charnay, P. (1997). Family of Ebf/Olf-1-related genes potentially involved in neuronal differentiation and regional specification in the central nervous system. *Developmental Dynamics*, 210(3):191–205.

Giudici, P. and Green, P. J. (1999). Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243.

Hoadley, K. A., Weigman, V. J., Fan, C., Sawyer, L. R., He, X., Troester, M. A., Sartor, C. I., Rieger-House, T., B., S., P., Carey, L. A., and Perou, C. M. (2007). Egfr associated expression profiles vary with breast tumor subtype. *BMC genomics*, 8.

Holmes, C. C. and Walker, S. G. (2017). Assigning a value to a power likelihood in a general Bayesian model. *Biometrika*, 104(2):497–503.

Hue, N. T. K. and Chiogna, M. (2021). Structure learning of undirected graphical models for count data. *Journal of Machine Learning Research*, 22(50):1–53.

Ishwaran, H. and Rao, J. (2005). Spike and Slab Variable Selection: Frequentist and Bayesian Strategies. *The Annals of Statistics,*, 33(2):730–773.

Jiang, W. and Tanner, M. (2008). Gibbs posterior for variable selection in high-dimensional classification and data mining. *Ann. Statist.*, 36:2207–2231.

Kanehisa, M. and Goto, S. (2000). Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30.

Kim, J., Do, K.-A., Ha, M. J., and Peterson, C. B. (2019). Bayesian inference of hub nodes across multiple networks. *Biometrics*, 75(1):172–182.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

Lee, S. I., Ganapathi, V., and Koller, D. (2006). Efficient structure learning of markov networks using L_1-regularization. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Li, J., Witten, D. M., Johnstone, I. M., and Tibshirani, R. (2012). Normalization, testing, and false discovery rate estimation for RNA-sequencing data. *Biostatistics*, 13 3:523–38.

Liu, S., Quinn, J. A., Gutmann, M. U., and Sugiyama, M. (2014). Direct learning of sparse changes in Markov networks by density ratio estimation. *Neural Comput*, 26:1169–1197.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462.

Mukherjee, S. and Speed, T. P. (2008). Network inference using informative priors. *Proceedings of the National Academy of Sciences*, 105(38):14313–14318.

Müller, P., Parmigiani, G., and Rice, K. (2007). FDR and Bayesian Multiple Comparisons Rules. In *Bayesian Statistics 8: Proceedings of the Eighth Valencia International Meeting June 2–6, 2006*. Oxford University Press.

Network, C. G. A. (2012). Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70.

Ninkovic, J., Steiner-Mezzadri, A., Jawerka, M., Akinci, U., Masserdotti, G., Petricca, S., Fischer, J., von Holst, A., Beckers, J., Lie, C. D., Petrik, D., Miller, E., Tang, J., Wu, J., Lefebvre, V., Demmers, J., Eisch, A., Metzger, D., Crabtree, G., Irmler, M., Poot, R., and Götz, M. (2013). The BAF complex interacts with Pax6 in adult neural progenitors to establish a neurogenic cross-regulatory transcriptional network. *Cell Stem Cell*, 13(4):403–418.

Ormerod, J. T. and Wand, M. P. (2010). Explaining variational approximations. *The American Statistician*, 64(2):140–153.

Pearl, J. and Paz, A. (1986). GRAPHOIDS: Graph-based logic for reasoning about relevance relations or when would x tell you more about y if you already know z? *Probabilistic and Causal Inference*.

Plaksienko, A., Thoresen, M., and Djordjilović, V. (2025). Methods for differential network estimation: an empirical comparison.

Ray, K. and Szabó, B. (2022). Variational bayes for high-dimensional linear regression with sparse priors. *Journal of the American Statistical Association*, 117(539):1270–1281.

Ročková, V. and George, E. I. (2014). Emvs: The em approach to bayesian variable selection. *Journal of the American Statistical Association*, 109(506):828–846.

Rohde, D. and Wand, M. P. (2016). Semiparametric mean field variational bayes: general principles and numerical issues. *The Journal of Machine Learning Research*, 17(1):5975–6021.

Roverato, A. (2002). Hyper inverse wishart distribution for non-decomposable graphs and its application to bayesian inference for gaussian graphical models. *Scandinavian Journal of Statistics*, 29(3):391–411.

Scheipl, F. (2011). spikeSlabGAM: Bayesian Variable Selection, Model Choice and Regularization for Generalized Additive Mixed Models in R. *Journal of Statistical Software*, 43(14):1–24.

Schmidt, M., Niculescu-Mizil, A., and Murphy, K. (2007). Learning graphical model structure using l1-regularization paths. *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07)*, 2:1278–1283.

Schnitt, S. J. (2010). Classification and prognosis of invasive breast cancer: from morphology to molecular taxonomy. *Modern Pathology*, 23:60–64.

Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Soc Sci Comput Rev*, 9(1):62–72.

Spirtes, P., Glymour, C., and Scheines, R. (1990). Causality from probability. *In Conference proceedings: advanced computing for the social sciences*.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

Srebro, N. (2003). Maximum likelihood bounded tree-width markov networks. *Artificial Intelligence*, 143(1):123–138.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density Ratio Estimation in Machine Learning*. Cambridge University Press.

Syring, N. and Martin, R. (2018). Calibrating general posterior credible regions. *Biometrika*, 106(2):479–486.

Talluri, R., Baladandayuthapani, V., and Mallick, B. K. (2014). Bayesian sparse graphical models and their mixtures. *Stat*, 3(1):109–125.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65:31–78.

Tu, J.-J., Ou-Yang, L., Zhu, Y., Yan, H., Qin, H., and Zhang, X.-F. (2021). Differential network analysis by simultaneously considering changes in gene interactions and gene expression. *Bioinformatics*, 37(23):4414–4423.

Vogels, L., Mohammadi, R., Schoonhoven, M., and Birbil, S. I. (2024). Bayesian structure learning in undirected gaussian graphical models: Literature review with empirical comparison. *Journal of the American Statistical Association*, 119(548):3164–3182.

Wand, M., Ormerod, J., Padoan, S., and Fuhrwirth, R. (2011). Mean field variational bayes for elaborate distributions. *Bayesian Analysis*, 6.

Wang, H. (2015). Scaling it up: Stochastic search structure learning in graphical models. *Bayesian Analysis*, 10(2).

Wang, S. S., Tsai, R. Y. L., and Reed, R. R. (1997). The characterization of the Olf-1/Ebf-like HLH transcription factor family: Implications in olfac-

tory gene regulation and neuronal development. *Journal of Neuroscience*, 17(11):4149–4158.

Wu, P., Li, X., Wu, Y., Hu, W., Wang, Y., Gao, L., Chen, Z., and Zheng, W. (2014). Overexpression of raf-1 in basal-like carcinoma of the breast: correlation with clinicopathology and prognosis. *Contemporary Oncology/Współczesna Onkologia*, 18(6):391–395.

Yang, N., Veerabhadran, B., Marina, V., and Francesco C, S. (2022). Bayesian graphical models for modern biological applications. *Stat Methods Appt*, 31(2):197–225.

Zhang, R., Ren, Z., Celed'on, J. C., and Chen, W. (2021). Inference of large modified poisson-type graphical models: Application to rna-seq data in childhood atopic asthma studies. *The Annals of Applied Statistics.*, 15(2):831–855.

Zhang, T. (2006a). From ϵ-entropy to KL-entropy: Analysis of minimum information complexity density estimation. *Ann. Statist.*, 34:2180–2210.

Zhang, T. (2006b). Information theoretical upper and lower bounds for statistical estimation. *IEEE Trans. Inform. Theory*, 52:1307–1321.

Zhao, S. D., Cai, T. T., and Li, H. (2014). Direct estimation of differential networks. *Biometrika*, 101(2):253–268.

Zhao, X., Rouhiainen, A., Li, Z., Li, Z., and Li, Z. (2020). Regulation of neurogenesis in mouse brain by hmgb1. *Cells*, 9(7):1714.