Latent Factor Analysis and Regression of Multivariate Poisson Lognormal Counts using Amortized Variational Inference

A thesis submitted in fulfilment of the requirements

for the degree of Master of Science by

Ananyapam De

under the supervision of ***Dr. Johannes Söding***,

Max Planck Institute for Multidisciplinary Sciences

and co-supervisor ***Dr. Satyaki Mazumder***





Department of Mathematics and Statistics

Indian Institute of Science Education and Research Kolkata

May 2023

Certificate

It is certified that the work contained in this thesis entitled "Latent Factor Analysis and Regression of Multivariate Poisson Lognormal Counts using Amortized Variational Inference" by "Ananyapam De" affiliated to the Indian Institute of Science Education and Research Kolkata has been carried out under my supervision and that it has not been submitted elsewhere for a degree.



Dr. Satyaki Mazumder

Assistant Professor

Department of Mathematics and Statistics

Indian Institute of Science Education and Research Kolkata

Dr. Johannes Söding

Group Leader

Quantitative and Computational Biology

Max Planck Institute for Multidisciplinary Sciences

Abstract

Name of the student: **Ananyapam De** Roll No: **18MS075**

Degree: **Master of Science** Department: **Mathematics and Statistics**

Thesis title: **Latent Factor Analysis and Regression of Multivariate Poisson Lognormal Counts using Amortized Variational Inference**

Thesis supervisor: **Dr. Johannes Söding**

Thesis co-supervisor: **Dr. Satyaki Mazumder**

Month and year of thesis submission: **May 2023**

Correlated count variables are ubiquitous and are typically modelled using normal distributions to analyze their correlations or to conduct regression. However, the normal approximation is not very accurate when many of these counts are less than 10. The Poisson lognormal describes D correlated counts $y_i \in \{0, 1, 2 \ldots\}$ as a mixture of Poissons with rates $\mu_i^y = e^{z_i}$ with a per-sample latent variables $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This distribution is a natural alternative to the normal distribution for count vectors since it models overdispersed and correlated distributions. In this work we present a variational inference method to train models based on this distribution, that allows us to perform regression and latent factor analysis for overdispersed correlated counts. We use *amortization* allowing us to perform accurate inference by learning only a small number of parameters. We begin by employing Laplace approximations and later move on to using neural networks. Based on the results of our study, we assert that our model outperforms *PLNmodels*, the highly effective approach developed by *Chiquet et al.* for learning correlated count data and present evidence to support this claim which highlight the distinctive features of our model that contribute to its performance. Further we extend this approach to other distributions like the Bernoulli, Gamma, Gumbel etc.

Acknowledgements

First and foremost, I'd like to thank my supervisor, Dr. Johannes Soeding, for all his guidance and support throughout this journey. Johannes, you've been an amazing supervisor and I couldn't have done it without you. Thank you giving me the much needed push after the first semester which fueled me to work harder for the project!

I'd also like to thank my guide, Amelie, for her invaluable insights and suggestions. Amelie, you have consistently inspired me to enhance my presentation skills. Your patient review of my thesis countless times has been instrumental in shaping it into what it is now. I would also like to extend my appreciation to Benjamin and Etienne for their expert advice and encouragements which have been invaluable in helping me overcome the obstacles I faced during the project.

To my mentors Satyaki and Anirvan sir who have been instrumental in shaping my academic career, I deeply respect both of you. Thank you both for your unwavering support in always helping me out with my doubts whenever I dropped by your office unannounced!

To my friends Abhay, Divyanshu, Rithik, Rahul, Harshal, Shashank, Dakshesh, Adwait, Shreyas, Praful and Ankit - thanks for always having my back. Thank you for always being there to lend a listening ear, to help me out when I messed up or just to hang out and have a good time. You've made this journey so much more enjoyable. I truly cannot imagine how I would have persevered through the rigors of this demanding department (DMS) without each and every one of you by my side.

To my beloved seniors Vivek, Siddhikant and Prince, thanks a lot for guiding me through my journey. You guys were never too busy to help out and your wisdom and advice have shaped my path. Thank you from the bottom of my heart for being such influential mentors in my life.

Of course, none of this would have been possible without the love and support of my amazing parents and family members. Baba and Maa, thank you for everything.

And last but not least, a very special thank you to Srushti. You've been my source of motivation and my constant companion. Thank you for being there through the highs and the lows. I couldn't have done it without you. (Not to mention how grateful I am to you for helping me with my plots and making me realize I have no aesthetic sense at all!)

Cheers to many more adventures together!

Contents

Certificate **i**

Abstract **ii**

Acknowledgements **iii**

Contents **iv**

List of Figures **vii**

List of Tables **viii**

Abbreviations **ix**

Symbols **x**

1 Introduction **1**
1.1 Motivation 1
1.2 Challenges with GLMMs 3
1.2.1 Markov Chain Monte Carlo 3
1.2.2 Variational Inference 4
1.3 Recent Works and Our Contribution 4

2 Variational Inference **5**
2.1 Introduction 5
2.2 Variational Inference 6
2.2.1 Setup 6
2.2.2 KL Divergence 7
2.2.3 The Evidence Lower Bound 7
2.2.4 Maximizing the ELBO 9

3 Generalized Linear Mixed Models **10**
3.1 Introduction 10

3.2 The Generalized Linear Model (GLM) . . . 10
3.3 The Generalized Linear Mixed Model (GLMM) . . . 12
3.3.1 Discovering Latent Factors . . . 12
3.3.2 Special Case: Poisson lognormal . . . 13

4 Methodology **14**
4.1 Introduction . . . 14
4.2 Amortized Variational Inference . . . 15
4.3 Laplace Approximation . . . 16
4.4 Neural network for amortization . . . 17
4.5 Designing The Variational Distribution . . . 17
4.6 The Evidence Lower Bound (ELBO) . . . 18
4.7 The Expected Conditional Log Probability . . . 21
4.7.1 Gauss-Hermite Quadrature . . . 21
4.8 Optimizing the ELBO . . . 22
4.8.1 Initialization of the parameters . . . 23
4.8.1.1 Initializing $\mathbf{B}$ and $\boldsymbol{\mu}$ . . . 24
4.8.1.2 Initializing $\boldsymbol{\phi}$ and the network weights . . . 24
4.8.2 Regularization . . . 25
4.9 Time Complexity . . . 26

5 Results **27**
5.1 Introduction . . . 27
5.2 Sample Demonstration . . . 28
5.2.1 Results . . . 28
5.3 Simulation Study . . . 30
5.3.1 Results . . . 31
5.4 Real Datasets . . . 32
5.4.1 Barents Fish Dataset . . . 32
5.4.2 Trichoptera Dataset . . . 33
5.4.3 Oaks Amplicon Dataset . . . 34
5.5 Runtime Analysis . . . 35

6 Conclusion and Future Work **37**
6.1 Conclusion . . . 37
6.2 Future Work . . . 38

A Efficient Implementation and Time Complexity **39**
A.1 Introduction . . . 39
A.1.1 Prerequisites . . . 39
A.1.2 Weinstein–Aronszajn identity. . . . 40
A.1.3 Woodbury matrix identity. . . . 40
A.2 Complexity of Computing the ELBO . . . 40

A.3 Computing $|\mathbf{S}_n|$ . . . 41
A.4 Computing $(\mathbf{S}_n)_{ii}$ . . . 41
A.5 Computing $|\boldsymbol{\Sigma}|$ . . . 42
A.6 Computing $\mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right)$ . . . 42
A.7 Computing $(\boldsymbol{\mu} - \boldsymbol{m}_n)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{m}_n)$ . . . 43
A.8 Overall Time Complexity . . . 44

B Laplace Approximations for Response Variable Distributions 45
B.1 Bernoulli. . . . 45
B.2 Binomial. . . . 46
B.3 Gamma. . . . 46
B.4 Natural exponential family. . . . 46
B.5 Gumbel. . . . 47

C Expected Conditional Log Probability for Response Variables 48
C.1 Bernoulli. . . . 48
C.2 Binomial. . . . 49
C.3 Gamma. . . . 49
C.4 Natural Exponential Family. . . . 49
C.5 Gumbel. . . . 50

D Entropy and the Cross-Entropy of the Multivariate Gaussian 51
D.1 Entropy of the Multivariate Gaussian . . . 51
D.2 Cross Entropy between Multivariate Gaussians . . . 52

E Derivation of ELBO for one-dimensional case 53

Bibliography 56

List of Figures

2.1 Schematic of Variational Inference 6
2.2 Decomposition of the evidence 8

3.1 Plate Notation for GLM and GLMM's 11

4.1 Neural Network for Amortization 17

5.1 Demonstration on sample dataset- (a) Convergence of μ (b) Convergence of Σ (c) Convergence of $|||\phi|||$'s (d) Convergence of ELBO. 29
5.2 Correlation heatmaps of sample dataset: (a) Correlation matrix with which the count data $\mathbf{Y}$ where generated. (b) Correlation matrix derived using our method. (c) Correlation matrix derived using PLNmodels. 30
5.3 Comparison of our method with PLNmodels with different parameter settings- $D = \{2, 10, 90\}$, $Q = \{2, 5, 10\}$ and $N = \{50, 500, 1000\}$: Box plots of total absolute difference of the parameter estimates and true parameter values computed over $R = 50$ replicates. 31
5.4 Barents dataset: (a) Count heatmap (b) Covariate heatmap. 32
5.5 Correlation heatmaps of Barents dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels. . . 33
5.6 Trichoptera Dataset: (a) Count Heatmap (b) Covariate Heatmap. 33
5.7 Correlation heatmaps of Trichoptera dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels. 34
5.8 Oaks Amplicon Dataset: (a) Count Heatmap (b) Covariate Heatmap (c) Factor Histogram. 34
5.9 Correlation heatmaps of Oaks dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels. . . 35
5.10 Comparison of average runtime of our method and PLNmodels. 35

List of Tables

5.1 Comparison of our method with PLNmodels on sample dataset. 28

Abbreviations

GLMM	**G**eneralized **L**inear **M**ixed **M**odel
GLM	**G**eneralized **L**inear **M**odel
VI	**V**ariational **I**nference
EM	**E**xpectation-**M**aximization
MLE	**M**aximum **L**ikelihood **E**stimation
MCMC	**M**arkov **C**hain **M**onte **C**arlo
ELBO	**E**vidence **L**ower **B**ound
KL divergence	**K**ullback-**L**eibler divergence
LBFGS	**L**imited-Memory **B**royden-**F**letcher-**G**oldfarb-**S**hanno
ADAM	**A**d**a**ptive **M**oment Estimation
AVX	**A**dvanced **V**ector E**x**tension

Symbols

N	Number of Observations
D	Number of Dimensions
K	Number of Latent Factors
Q	Number of Covariates
$\mathbf{Y}$	Response matrix (Count data) / Target variable
$\mathbf{X}$	Covariate matrix/ Observed variables
$\boldsymbol{\theta}$	Model Parameters
$\boldsymbol{\phi}$	Variational Parameters
H	Number of neurons in Neural Network
L	Number of layers in Neural Network
$\mathcal{H}$	Entropy
t	Number of iterations

To my beloved parents,
for helping me become what I am or ever will be ...

Chapter 1

Introduction

> "Count data is like a box of chocolates - you never know what you're gonna get, but it's usually some variation of a Poisson distribution."
>
> Forrest Gump (or probably not)

1.1 Motivation

Count data refers to a type of data that consists of non-negative integer values, representing the number of occurrences or events within a specific time period, space, or other unit of analysis. Many measurements are in the form of counts rather than being continuous. For instance, in epidemiology, count data is often utilized to monitor the incidence of diseases or health conditions within a population. In marketing, count data tracks the number of clicks or conversions on advertisements or websites. In finance, count data keeps a record of the number of trades or transactions within a market. Several other examples of count data include the number of customers entering a store, the frequency of accidents on a road, the volume of calls received by a call center, the count of trees in a forest, or the population size of certain animals.

Correlated count data are another type counts in which the counts in one variable are related to the counts in another variable. This correlation can arise due to various reasons such as common underlying factors, spatial or temporal auto-correlation, or feedback mechanisms. For example, in ecology, the number of predators and prey in an ecosystem can be correlated as an increase in the number of predators may lead to a decrease in the number of prey. Similarly, in

social sciences, the number of crimes and arrests in a neighborhood can be correlated because an increase in the number of crimes may lead to an increase in the number of arrests.

Although techniques that account for correlation between variables can be useful, when the expected counts are high enough, correlated counts can be approximated well by multivariate normal distributions. In that case, we can use a plethora of models from linear mixed models for latent factor analysis to Gauss-Markov models in network analysis. However, when many counts are less than 5 or 10, the normal approximation turns out to be quite inaccurate.

For instance, in transcriptomics, gene expression data is often low and highly correlated due to the shared biological mechanisms underlying gene regulation [1]. This correlation can pose significant challenges for data analysis, as it can lead to biased estimates of the effects of interest and inflated type I error rates. Additionally, gene expression data obtained from single-cell RNA-seq experiments are often sparse and contain a large number of zero counts [2]. This sparsity and over-dispersion further complicates the analysis of gene expression data, as it violates the assumptions of traditional statistical models such as the Poisson distribution or the normal approximation as discussed earlier.

To overcome these challenges, specialized methods are required that can account for the correlation and sparsity of gene expression data. In recent years, many statistical methods have been developed for the analysis of transcriptomic data, including Bayesian hierarchical models [3, 4], zero-inflated models [5, 6], and Generalized Linear Models (GLMs) [7, 8]. These methods allow for the incorporation of multiple sources of variability in the data, including batch effects, sample-to-sample variation, and technical noise. Despite the advances in statistical methodology for the analysis of transcriptomic data, the problem of low counts and high correlation remains a challenge.

Generalized Linear Mixed Models (GLMMs) [9] have emerged as a powerful tool for analyzing such data, as they can account for both the correlation and the sparsity of count data [10]. While we will delve into the details of GLMMs in later chapters, it's important to note that they also present several challenges.

1.2 Challenges with GLMMs

A major challenge in utilizing multivariate generalizations of univariate distributions like GLMMs has been the difficulty in learning these distributions from training data, primarily due to the absence of closed form solutions. Additionally, incorrect specification of the model may result in convergence issues, leading to unreliable or inaccurate outcomes. One common approach to addressing the challenges of learning multivariate generalizations of univariate distributions like GLMMs is through Markov Chain Monte Carlo algorithms.

1.2.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) have been widely adopted in statistical inference to train such models and simulate intricate stochastic systems. The fundamental concept behind MCMC is to generate a Markov chain that ultimately converges to the target distribution, which may be intractable or challenging to sample directly from. It is advantageous in exploring high-dimensional spaces, even when the distribution of interest is multimodal. This is achieved by constructing a Markov chain that has the target distribution as its stationary distribution and subsequently using it to generate samples from the distribution of interest. Popular MCMC algorithms include the Metropolis-Hastings and Gibbs sampling.

However, MCMC methods are notoriously slow and scale badly to large data sets since they involve repeatedly simulating a Markov chain, which involves generating a sequence of random samples from the posterior distribution of the parameters. The slowness of MCMC methods is mainly due to the fact that they require a large number of iterations to generate a sufficiently large and accurate sample of the posterior distribution. The number of iterations required depends on the complexity of the model, the number of parameters, and the number of data points. The samples are generated by making small changes to the current state of the chain, based on a set of transition probabilities, and accepting or rejecting the proposed changes based on the likelihood of the new sample. This process is repeated many times to generate a large sample of the posterior distribution. Additionally, MCMC can be slowed down by convergence issues, such as high auto-correlation or low acceptance rates, which can cause the chain to get stuck in a region of parameter space. To speed up MCMC, several methods have been proposed, including parallelization, adaptive MCMC algorithms, and importance sampling. However, these methods come at the cost of increased complexity or reduced accuracy.

1.2.2 Variational Inference

Variational Inference (VI) [11, 12, 13] has emerged as a compelling alternative to Markov Chain Monte Carlo (MCMC) for probabilistic inference tasks. This class of methods approximates complex distributions with simpler ones, making the problem more tractable. VI simplifies complex distributions and improves computational efficiency, making it attractive for tackling challenging data analysis problems at a faster pace, which has led to its increasing popularity in recent years. In Chapter 2, we provide a detailed discussion on VI and derive the expression of ELBO, which is a crucial component of the VI framework.

1.3 Recent Works and Our Contribution

Research in the field has primarily centered around refining Markov Chain Monte Carlo algorithms to improve the efficiency and accuracy of learning GLMMs [14]. The use of popular software tools such as Just Another Gibbs Sampler (JAGS), Bayesian inference Using Gibbs Sampling (BUGS) and STAN have been extensively adopted for this purpose [15, 16]. However these approaches are extremely slow and recent work has focused on developing scalable MCMC algorithms for large datasets like the stochastic gradient MCMC [17], parallel tempering MCMC [18, 19], Hamiltonian Monte Carlo [20], and sub-sampling MCMC [21]. Despite the improvement in MCMC speed achieved through these approaches, they are still notoriously slow and scale badly to large datasets [22, 23].

Variational methods for learning GLMMs such as Penalized Quasi-Likelihood [24], Laplace approximation [25], and Gaussian variational Approximation [26] have also been proposed. Recently, *Chiquet et al.* have developed a successful variational inference approach in their package *PLNmodels* to speed up learning by orders of magnitude [27, 28, 29], though at the cost of accuracy in parameter and error estimates. This approach uses the variational EM algorithm to learn model parameters, where instead of computing the exact posterior distribution of the latent variables, it approximates it with a simpler Gaussian distribution.

Our novel approach uses variational inference with amortization to learn multivariate GLMMs, such as the Poisson lognormal model, with a carefully designed variational distribution that requires fewer parameters to learn. We compute the ELBO objective analytically using Gauss-Hermite quadrature or exact integration, resulting in more accurate, and easily generalizable results compared to MCMC approaches and the unamortized EM approach of *Chiquet et al.*

Chapter 2

Variational Inference

> "Variational inference is that thing you implement while waiting for your Gibbs sampler to converge."
>
> David Blei

2.1 Introduction

Variational methods have their origins rooted in calculus of variations. Unlike standard calculus which help us examine the extremas of functions, calculus of variations is concerned with functionals. A functional is a special type of function, a mapping that takes a function to a scalar. A simple example of a functional would be the expectation value E associated with a probability distribution p which maps p to $\mathbb{R}$ such that

$$E[p] = \int xp(x)\,dx.$$

Similar to ordinary calculus, we can introduce the notion of a *functional derivative* which expresses how much the value of the functional changes upon small changes in the input function. This way we can adjust the input function to optimize an objective functional.

2.2 Variational Inference

Suppose we have a distribution $p(\mathbf{Z}|\mathbf{X})$ which poses challenges in direct sampling. We circumvent this issue by seeking a distribution $q(\mathbf{Z}|\phi)$ from a more tractable class of distributions, where ϕ represents the parameters. The objective is to find the closest approximation of the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ by optimizing ϕ within this class. By optimizing ϕ, we aim to align $q(\mathbf{Z}|\phi)$ with $p(\mathbf{Z}|\mathbf{X})$. This transformation shifts our original sampling problem towards an optimization problem. Figure (2.1) illustrates the underlying principles of variational inference where ϕ is optimized to minimize "a measure of dissimilarity" between the distributions $p(\mathbf{Z}|\mathbf{X})$ and $q(\mathbf{Z}|\phi)$.

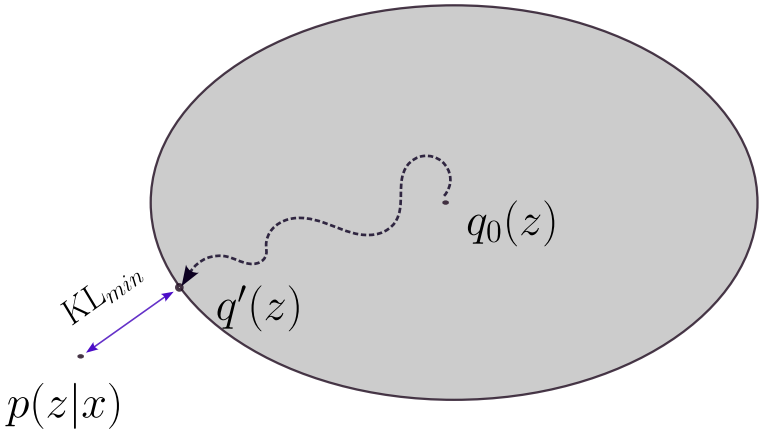


FIGURE 2.1: Schematic of Variational Inference

2.2.1 Setup

Suppose we have a generative model with the observed variables $\boldsymbol{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ the latent variables $\boldsymbol{Z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n)$ and the model parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots \boldsymbol{\theta}_m)$. Then the model likelihood will be given by:

$$p(\boldsymbol{X}|\boldsymbol{\theta}) = \int p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\theta}) p(\boldsymbol{Z}|\boldsymbol{\theta}) d\mathbf{Z}.$$

If we have a discriminative model with the target variables denoted as $\boldsymbol{Y}$, then the model will be given by:

$$p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}) p(\boldsymbol{Z}|\boldsymbol{\theta}) d\mathbf{Z}.$$

Our goal with such models are mainly to either learn the model parameters $\boldsymbol{\theta}$ or to sample from posterior $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})$ or $p(\boldsymbol{Z}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta})$. Since sampling from $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})$ directly is difficult, we sample from another class of distributions $q(\boldsymbol{Z}|\boldsymbol{\phi})$ after optimizing $\boldsymbol{\phi}$ such that $q(\boldsymbol{Z}|\boldsymbol{\phi}) \approx$

$p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})$, where $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_k)$ parametrize the distribution q. This reduces our problem from an sampling problem to an optimization problem. The distribution $q(\boldsymbol{Z}|\boldsymbol{\phi})$ is called the *variational distribution* and the parameters $\boldsymbol{\phi}$ are called as the *variational parameters*. Let us now precisely define closeness or discrepancy between two probability distributions.

2.2.2 KL Divergence

The Kullback–Leibler divergence (KL divergence) is a powerful measure of the discrepancy between two probability distributions. The KL divergence quantifies the difference in information content between two distributions, where the information content is defined as the negative logarithm of the probability density. For two continuous probability distributions p and q, it is defined as

$$D_{\mathrm{KL}}(p||q) = \int p(x) \ln \frac{p(x)}{q(x)}\, dx$$

where the integral is over the support of p and where $q(x) = 0$ implies $p(x) = 0$. This expression measures the average difference in information content between the true posterior distribution p and the simpler distribution q that we use to approximate it. It is worth noting that the choice of divergence measure depends on the specific problem at hand and the nature of the distributions being compared. For example, the Tsallis α-divergence [30] has been shown to be more effective in some scenarios than the KL divergence. The asymmetry of the KL divergence, which disqualifies it as a metric, turns out to be an advantage, as we will now see.

2.2.3 The Evidence Lower Bound

We now compute the KL divergence between the two distributions $p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})$ and $q(\boldsymbol{Z}|\boldsymbol{\phi})$:

$$\begin{aligned}
D_{\mathrm{KL}}\big(q(\boldsymbol{Z}|\boldsymbol{\phi})||p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})\big) &= \int q(\boldsymbol{Z}|\boldsymbol{\phi}) \ln \frac{q(\boldsymbol{Z}|\boldsymbol{\phi})}{p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})}\, d\boldsymbol{Z}, && (2.1)\\
&= \int q(\boldsymbol{Z}|\boldsymbol{\phi}) \ln \frac{q(\boldsymbol{Z}|\boldsymbol{\phi}) p(\boldsymbol{X}|\boldsymbol{\theta})}{p(\boldsymbol{Z},\boldsymbol{X}|\boldsymbol{\theta})}\, d\boldsymbol{Z}, && (2.2)\\
&= \int q(\boldsymbol{Z}|\boldsymbol{\phi}) \ln p(\boldsymbol{X}|\boldsymbol{\theta})\, d\boldsymbol{Z} - \int q(\boldsymbol{Z}|\boldsymbol{\phi}) \ln \frac{p(\boldsymbol{Z},\boldsymbol{X}|\boldsymbol{\theta})}{q(\boldsymbol{Z}|\boldsymbol{\phi})}\, d\boldsymbol{Z}, && (2.3)\\
&= \ln p(\boldsymbol{X}|\boldsymbol{\theta}) - \mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi}), && (2.4)
\end{aligned}$$

where

$$\mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi}) = \int q(\boldsymbol{Z}|\boldsymbol{\phi}) \ln \frac{p(\boldsymbol{Z},\boldsymbol{X}|\boldsymbol{\theta})}{q(\boldsymbol{Z}|\boldsymbol{\phi})}\, d\boldsymbol{Z} \;= \mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{\phi})}\left[\ln \frac{p(\boldsymbol{Z},\boldsymbol{X}|\boldsymbol{\theta})}{q(\boldsymbol{Z}|\boldsymbol{\phi})} \right]$$

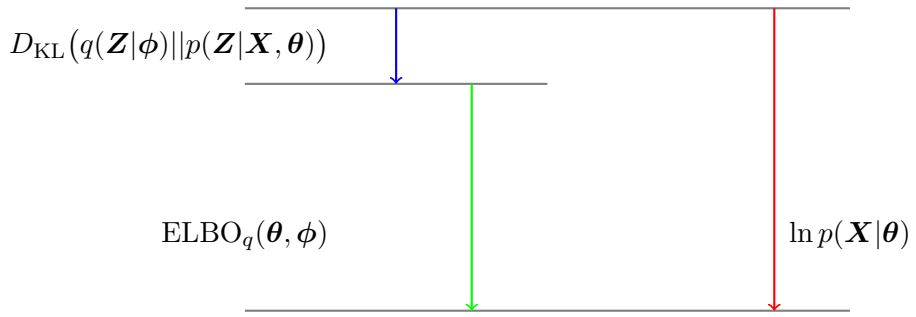


FIGURE 2.2: Decomposition of the evidence

which is called as *Evidence Lower Bound (ELBO)*. The term $\ln p(\boldsymbol{X}|\boldsymbol{\theta})$ is called the *model evidence*. The KL divergence is non-negative because $\ln x \leq x-1$ with "=" if and only if $x=1$, we have

$$-D_{KL}(q\|p) = \sum_z q(z) \ln \frac{p(z)}{q(z)} \leq \sum_z q(z) \left(\frac{p(z)}{q(z)} - 1 \right) = 0$$

with equality if and only if $x_z = p(z)/q(z) = 1$ for all z summed over, that is $p \equiv q$ and hence

$$D_{\mathrm{KL}}\big(q(\boldsymbol{Z}|\boldsymbol{\phi})||p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})\big) \geq 0, \tag{2.5}$$

$$\ln p(\boldsymbol{X}|\boldsymbol{\theta}) - \mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi}) \geq 0, \tag{2.6}$$

$$\ln p(\boldsymbol{X}|\boldsymbol{\theta}) \geq \mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi}). \tag{2.7}$$

The term $\mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi})$ acts as a lower bound for the model evidence, as the name suggests. Notice that the model evidence can be decomposed as a sum of the KL divergence and the ELBO. This decomposition holds for any choice of the variational distribution $q(\boldsymbol{Z}|\boldsymbol{\phi})$,

$$\ln p(\boldsymbol{X}|\boldsymbol{\theta}) = D_{\mathrm{KL}}\big(q(\boldsymbol{Z}|\boldsymbol{\phi})||p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})\big) + \mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi}) \tag{2.8}$$

as shown in Fig.(2.2). Hence, the smaller the KL divergence, the higher will be the ELBO, the tighter will be the bound. Hence minimizing the KL divergence with respect to $\boldsymbol{\phi}$ and maximizing the ELBO are one and the same (keeping $\boldsymbol{\theta}$ fixed). This way we have transformed our objective function from the KL divergence to the ELBO. We will now discuss about our next objective to maximize the $\mathrm{ELBO}_q(\boldsymbol{\theta},\boldsymbol{\phi})$.

2.2.4 Maximizing the ELBO

It is a natural inquiry to consider which parameters should be maximized to optimize the Evidence Lower Bound (ELBO). By maximizing the ELBO with respect to the variational parameters, denoted as $\boldsymbol{\phi}$, we minimize the KL divergence between the true distribution, $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})$, and the variational distribution, $q(\boldsymbol{Z}|\boldsymbol{\phi})$. In turn, this approximation is used to optimize the lower bound of $\ln p(\boldsymbol{X}|\boldsymbol{\theta})$ by maximizing the ELBO with respect to the model parameters, $\boldsymbol{\theta}$. The ELBO is tighter when the KL divergence is smaller, leading to a more accurate approximation and a maximum closer to the true value of $\ln p(\boldsymbol{X}|\boldsymbol{\theta})$. Therefore, the goal is to maximize the ELBO with respect to both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

In the following chapter, we discuss about Generalized linear mixed models to provide the reader some background on the kind of models that we aim to perform inference on and discuss why that has been a difficult task to accomplish.

Chapter 3

Generalized Linear Mixed Models

> "Linear models are like teenagers - they work great when they're well-behaved, but they can be a nightmare when they're not."

3.1 Introduction

Generalized Linear Mixed Models (GLLMs) are a type of statistical model that allows for the analysis of data that have non-normal distributions, such as count data, binary data, or ordinal data. They are an extension of the Generalized Linear Models (GLMs) that account for the correlation among the observations by including random effects.

In GLMMs, the response variable is modeled as a function of both fixed and random effects, where the fixed effects are the explanatory variables that are of primary interest, and the random effects account for the correlation among the observations. The distribution of the response variable is specified using a link function, and the model can include both continuous and categorical explanatory variables. This allows for modelling correlations among the observations while simultaneously accounting for the influence of the explanatory variables.

3.2 The Generalized Linear Model (GLM)

Suppose we have a univariate distribution $p\big(y_i|\mu_i^y, \boldsymbol{\psi}_i\big)$ such as a Poisson, Bernoulli, Binomial, Multinomial, Gamma, or Gumbel distribution to name but a few. Here $\mu_i^y = \mathbb{E}[y_i]$ is the location

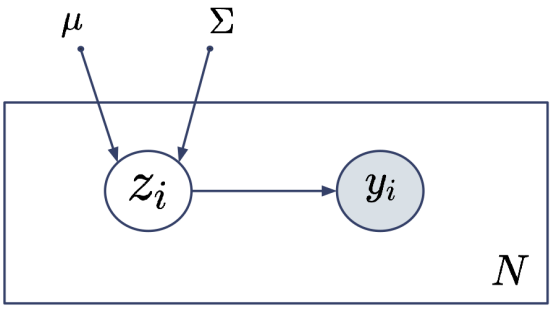


FIGURE 3.1: Plate Notation for GLM and GLMM's

parameter of this distribution and $\boldsymbol{\psi}_i$ may contain further scale and shape parameters.

Suppose we require a multivariate version of it to model correlations between the y_i for which we might assume that the μ_i^y themselves are normally distributed with some mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. This would yield the model

$$p(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\psi}) = \int \left(\prod_{i=1}^{D} p\big(y_i|\mu_i^y = z_i, \boldsymbol{\psi}_i\big) \right) \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\, \mathrm{d}\mathbf{z}.$$

However, for many distributions such as the Poisson distribution the location parameter μ_i^y cannot be negative, for others such as the Bernoulli distribution it has to be between 0 and 1. Therefore, we map μ_i^y onto $\mathbb{R}$ using a link function $g(\cdot)$ and assume that these mapped values $z_i = g(\mu_i^y)$ are normally distributed with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$. The term "generalized" refers to the fact that it generalizes the normal distribution using a link function. The model therefore is

$$p(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\psi}) = \int \left(\prod_{i=1}^{D} p\big(y_i|\mu_i^y = g^{-1}(z_i), \boldsymbol{\psi}_i\big) \right) \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\, \mathrm{d}\mathbf{z}\,. \tag{3.1}$$

We call this a *Generalized multivariate mixture distribution* and can be denoted using plate notation as shown in Fig.(3.1).

3.3 The Generalized Linear Mixed Model (GLMM)

This multivariate distribution of correlated y_i can be used for *multivariate regression* using response variables $\mathbf{Y}^{\mathrm{T}} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) \in \mathbb{R}^{D\times N}$ and covariates $\mathbf{X}^{\mathrm{T}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{R}^{Q\times N}$,

$$\begin{aligned} p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{B}, \boldsymbol{\psi}) &= \int \left(\prod_{i=1}^{D} p\big(y_{ni}|\mu_{ni}^{y} = g^{-1}(\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_n + z_{ni}), \boldsymbol{\psi}_i\big) \right) \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})\, \mathrm{d}\mathbf{z}_n, \\ &= \int \left(\prod_{i=1}^{D} p\big(y_{ni}|\mu_{ni}^{y} = g^{-1}(z_{ni}), \boldsymbol{\psi}_i\big) \right) \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma})\, \mathrm{d}\mathbf{z}_n\,, \end{aligned} \tag{3.2}$$

with regression weights $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_D) = \mathbf{B}^{\mathrm{T}} \in \mathbb{R}^{Q\times D}$.

3.3.1 Discovering Latent Factors

To discover *latent factors* in $(\mathbf{y}_1, \ldots, \mathbf{y}_N)$, we can write the covariance matrix as

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{\mathrm{T}} + \mathbf{D}, \tag{3.3}$$

the sum of a low-rank matrix $\mathbf{L}\mathbf{L}^{\mathrm{T}}$ with $\mathbf{L} \in \mathbb{R}^{D\times K}$ and a diagonal matrix $\mathbf{D} \in \mathbb{R}^{D\times D}$, revealing the K latent factors as columns of $\mathbf{L}$. We assume that we know the number of latent factors, K beforehand.

This model in equation (3.2) can then also be written as

$$\begin{aligned} \mathbf{z}_n &\sim \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \\ \mathbf{y}_n &\sim \prod_{i=1}^{D} p\big(y_i|\mu_i^{y} = g^{-1}(\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_n + z_i), \boldsymbol{\psi}_i\big). \end{aligned} \tag{3.4}$$

The latent variables $\mathbf{z}_n$ can be sampled as $\mathbf{z}_n = \boldsymbol{\mu} + \mathbf{L}\mathbf{u}_n + \boldsymbol{\epsilon}_n$, with $\mathbf{u}_n \in \mathbb{R}^K$ and $\boldsymbol{\epsilon}_n \in \mathbb{R}^D$. So the model can be written as (see, for example, eqs. (2.113 - 2.115) in [13] with $\mathbf{x} \leftarrow \mathbf{u}, \mathbf{L}^{-1} \leftarrow \mathbf{D}, \mathbf{A} \leftarrow \mathbf{L}, \mathbf{y} \leftarrow \mathbf{L}\mathbf{x} + \boldsymbol{\epsilon}$)

$$\begin{aligned} \boldsymbol{\epsilon}_n &\sim \mathcal{N}(\mathbf{0}, \mathbf{D}), \\ \mathbf{u}_n &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ \mathbf{y}_n &\sim \prod_{i=1}^{D} p\big(y_{ni}|\mu_{ni}^{y} = g^{-1}(\mu_i + \boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_n + (\mathbf{L}\mathbf{u}_n)_i + \epsilon_{ni}), \boldsymbol{\psi}_i\big). \end{aligned} \tag{3.5}$$

Here the *fixed effects* are $\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_n$ since the $\boldsymbol{\beta}_i$ and $\mathbf{x}_n$ are fixed while the *random effects* are $\mathbf{L}\mathbf{u}_n + \boldsymbol{\epsilon}_n$ since the $\mathbf{u}_n$ and $\boldsymbol{\epsilon}_n$ are drawn randomly from normal distributions. Generalized linear models containing fixed and random effects are called *mixed-effect models*. For this reason, the model in equations (3.2) and (3.5) is called a *multivariate Generalized Linear Mixed Model*.

3.3.2 Special Case: Poisson lognormal

A particularly interesting instance of multivariate distribution is the Poisson lognormal model for correlated counts $\mathbf{y}_n \in \{0, 1, 2, \ldots\}^D$,

$$p(\mathbf{y}_n|\boldsymbol{\theta}) = \int \left(\prod_{i=1}^{D} \mathrm{Pois}(y_{ni}|\mu_{ni}^y = e^{z_{ni}}) \right) \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \, \mathrm{d}\mathbf{z}_n \,, \tag{3.6}$$

where the Poisson distribution is $\mathrm{Pois}(y_{ni}|\mu_{ni}^y) = \frac{(\mu_{ni}^y)^{y_{ni}}}{y_{ni}!} e^{-\mu_{ni}^y}$. This model can again be used for multivariate regression,

$$p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\theta}) = \int \left(\prod_{i=1}^{D} \mathrm{Pois}\big(y_{ni}|\mu_{ni}^y = e^{z_{ni}}\big) \right) \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma}) \, \mathrm{d}\mathbf{z}_n \,, \tag{3.7}$$

with regression weights $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_D) = \mathbf{B}^{\mathrm{T}} \in \mathbb{R}^{Q \times D}$.

Such integrals are intractable which make mGLMMs hard to learn and we use variational inference for learning these kinds of GLMM's. In the following chapter, we will describe our methodology of learning such kinds of GLMM's by cleverly designing our proxy distribution and then optimizing the ELBO. We will also see how this approach can be easily extended for a variety of other models like the Bernoulli, Binomial, Gumbel etc.

Chapter 4

Methodology

> "Designing the variational distribution is an art, while constructing the objective function is the science."
>
> Johannes Söding

4.1 Introduction

In this chapter, we describe our methodology for learning GLMMs using variational inference using a technique called *amortization*. Amortized variational inference is a technique that enables us to estimate the posterior distribution for an entire sample by learning a fixed set of variational parameters, rather than computing them for each sample individually. This drastically reduces the number of parameters the model needs to learn leading to fast inference. We demonstrate the use a neural network for this purpose.

We show that our approach is computationally efficient, accurate and can scale to large datasets with complex hierarchical structures. Additionally, our approach provides a way to incorporate both fixed and random effects into the model, which can be important in many real-world applications.

4.2 Amortized Variational Inference

Recall that our model had the structure as follows-

$$p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{B}, \boldsymbol{\psi}) = \int \left(\prod_{i=1}^{D} p\big(y_{ni}|\mu_{ni}^{y}=g^{-1}(z_{ni}), \boldsymbol{\psi}_i\big) \right) \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma})\, \mathrm{d}\mathbf{z}_n.$$

To learn $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{B})$, we apply variational inference which consists of approximating the posterior distribution of the latent variables

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\theta}) &\propto p(\mathbf{y}_n|\mathbf{z}_n, \mathbf{x}_n, \boldsymbol{\psi})\, p(\mathbf{z}_n|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{D} p\big(y_{ni}|\mu_{ni}^{y}=g^{-1}(z_{ni}), \boldsymbol{\psi}_i\big) \times \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma}) \end{aligned} \tag{4.1}$$

with some suitable variational distribution $q(\mathbf{z}_n|\boldsymbol{\phi})$. The variational parameters $\boldsymbol{\phi}$ have to be learned jointly with $\boldsymbol{\theta}$ by maximizing the evidence lower bound (ELBO), as we discussed previously. Here Chiquet et al. use a multivariate Gaussian distribution $q(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\phi}_n) = \mathcal{N}(\mathbf{z}_n|\mathbf{m}_n + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Lambda}_n)$ *for each sample* n [27, 28, 29] and hence the variational parameters $\boldsymbol{\phi}_n = (\mathbf{m}_n, \boldsymbol{\Lambda}_n)$ have to be learned individually for each sample n. Additionally they assume $\boldsymbol{\Lambda}_n$ to be a diagonal covariance matrix which does not accurately model the correlations among the counts.

Here, we employ *amortization* instead. We learn a set of parameters $\boldsymbol{\phi}$ that are common to all samples in a dataset by conditioning on $\mathbf{y}_n$, as opposed to sample-specific parameters. In other words, they are parameters that are shared across all observations in the dataset. We approximate the response variable density using a Gaussian density:

$$q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}, \boldsymbol{\theta}) \propto \prod_{i=1}^{D} \mathcal{N}\big(z_{ni}|\eta_{ni}(y_{ni}, \boldsymbol{\phi}), \lambda_{ni}^{-1}(y_{ni}, \boldsymbol{\phi})\big) \times \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma}). \tag{4.2}$$

The trick here is that the correlations among the $\mathbf{z}_n$ imprinted via $\mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma})$ are separated out explicitly. Therefore, as in the posterior in equation (4.1) that we aim to approximate, we can express the remaining term simply as a product over i. Hence, now we only need to learn the variational parameters $\boldsymbol{\phi}$ for each dimension i and not for each sample n. This substantially reduces the number of parameters to learn and subsequently increases the speed of convergence. Further, this improves generalization, since capturing the commonalities across all samples improves the generalization performance of a model. This is useful in since our goal is to make accurate predictions on new, unseen data.

The complexity, however, is now in the dependence of $\eta_{ni}(y_{ni}, \boldsymbol{\phi})$ and $\lambda_{ni}^{-1}(y_{ni}, \boldsymbol{\phi})$. If the response has shape/scale parameters $\boldsymbol{\psi}_i$, we need to take that into account and formulate the $\eta_{ni}(y_{ni}, \boldsymbol{\phi}, \boldsymbol{\psi}_i)$ and $\lambda_{ni}^{-1}(y_{ni}, \boldsymbol{\phi}, \boldsymbol{\psi}_i)$ to also depend on $\boldsymbol{\psi}_{ni}$.

Now to find $\eta_{ni}(y_{ni}, \boldsymbol{\phi}, \boldsymbol{\psi}_i)$ and $\lambda_{ni}^{-1}(y_{ni}, \boldsymbol{\phi}, \boldsymbol{\psi}_i)$, we discuss two methods - first using the *Laplace approximation* and second using a *Neural Network*. The Laplace approximation is an experimental method to test the viability of the model and we aim to use Neural Networks in our model for amortization due to their flexibility.

4.3 Laplace Approximation

The *Laplace approximation* is a method for approximating a probability distribution with a normal (Gaussian) distribution, by matching the mode of the distribution and its curvature at that point. Here we summarize the Laplace approximations of the likelihoods of z_n for various response variable distributions.

For the Poisson distribution with $\mathbb{E}[y] = e^z$ (inverse link function $\mu = g^{-1}(z) = e^z$),

$$\begin{aligned} f(z) &= \ln \mathrm{Pois}(y|\mu{=}e^z) \\ &= y \ln \mu - \mu - \ln y! \\ &= yz - e^z + \mathrm{const}, \end{aligned} \tag{4.3}$$

has a mode at $z^* = \ln y$ and second derivative at the mode of $-e^{z^*} = -y$. The Laplace approximation is $f(z) \approx \ln \mathcal{N}(z|\ln y, 1/y) + \mathrm{const}$. We therefore set

$$\eta_i(y_i, \boldsymbol{\phi}) = \phi_{i,0} \ln \left(e^{\phi_{i,1}} + y_i \right) \tag{4.4}$$

$$\ln \lambda_i(y_i, \boldsymbol{\phi}) = \phi_{i,2} \ln \left(e^{\phi_{i,3}} + y_i \right) . \tag{4.5}$$

Detailed derivations for parameterizing $\boldsymbol{\phi}$ for different response variable distributions like the Binomial, Bernoulli, Gamma, Gumbel etc are discussed in the Appendix B.

4.4 Neural network for amortization

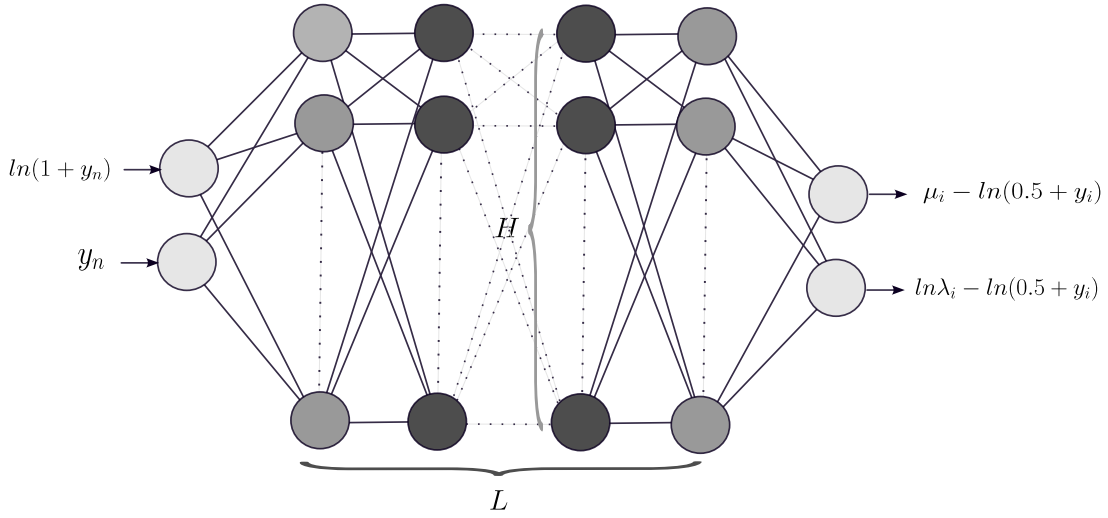


FIGURE 4.1: Neural Network for Amortization

The most flexible and general approach is to use for all response variable probability densities $p(\mathbf{y}_n|\mathbf{z}_n, \mathbf{x}_n, \boldsymbol{\psi})$ the same type of dense neural network, with two inputs $(\ln(1+y_n), y_n)$, L hidden layers with a H nonlinear arc tan nodes each, and two linear output units, one for η_i and the other for $\ln \lambda_i$:

$$\begin{pmatrix} \eta_i - \ln(0.5+y_i) \\ \ln \lambda_i - \ln(0.5+y_i) \end{pmatrix} \leftarrow \text{dense}(\text{lin}, 2, H) \circ \big(\text{dense}(\text{atan}, H, H)\big)^L \circ \text{dense}(\text{atan}, H, 2).$$

as shown in Fig.(4.1). The first argument in the layers is the response function, the second is the number of nodes in the layer. The network should be initialized with low weights and biases to give low values in the beginning, such that μ_i and $\ln \lambda_i$ will get initialized with values close to $\ln(0.5+y_i)$. H denotes layer size and typically good values are 3 or 5 and the values of L which denotes the depth of the network should typically be really shallow like 2 or 3, or it should be really deep like 20 or even more. In this context, the network weights are the variational parameters $\boldsymbol{\phi}$. Automatic differentiation yields the partial derivatives of μ_i and $\ln \lambda_i$ with respect to the variational parameters $\boldsymbol{\phi}$ which we can use for optimization.

4.5 Designing The Variational Distribution

Now that we have μ_{ni} and the λ_{ni}, we know our variational distribution. According to (4.2), since our variational distribution can be written as a product of Gaussians, we find the form of

the distribution explicitly as follows:

$$\begin{aligned}
&\ln q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}) \\
&= \sum_{i=1}^{D} \ln \mathcal{N}\big(z_{ni}|\eta_{ni}(y_{ni}, \boldsymbol{\phi}), \lambda_{ni}^{-1}(y_{ni}, \boldsymbol{\phi})\big) + \ln \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n, \boldsymbol{\Sigma}) + \text{const}, \qquad (4.6)\\
&= -\frac{1}{2}\sum_{i=1}^{D} \lambda_{ni}(y_{ni}, \boldsymbol{\phi})\big(z_{ni} - \eta_{ni}(y_{ni}, \boldsymbol{\phi})\big)^2 - \frac{1}{2}(\mathbf{z}_n - \boldsymbol{\mu} - \mathbf{B}\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{z}_n - \boldsymbol{\mu} - \mathbf{B}\mathbf{x}_n) + \text{const}\,.
\end{aligned}$$

Defining $\boldsymbol{\Lambda}_n = \text{diag}\left(\lambda_{ni}(y_{ni}, \boldsymbol{\phi})\right) \in \mathbb{R}^{D\times D}$ and $\mathbf{m}_n = (\eta_{ni}(y_{ni}, \boldsymbol{\phi})) \in \mathbb{R}^D$, we can write this more concisely as

$$\begin{aligned}
\ln q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}, \boldsymbol{\theta}) &= -\frac{1}{2}\Big(\mathbf{z}_n^{\mathrm{T}}\big(\boldsymbol{\Lambda}_n + \boldsymbol{\Sigma}^{-1}\big)\mathbf{z}_n - 2\,\mathbf{z}_n^{\mathrm{T}}\big(\boldsymbol{\Lambda}_n\mathbf{m}_n + \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}+\mathbf{B}\mathbf{x}_n)\big)\Big) + \text{const} \\
&= \ln \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{S}_n)\,, \qquad (4.7)
\end{aligned}$$

with

$$\mathbf{S}_n = (\boldsymbol{\Lambda}_n + \boldsymbol{\Sigma}^{-1})^{-1} \qquad (4.8)$$

$$\boldsymbol{\mu}_n = \mathbf{S}_n\big(\boldsymbol{\Lambda}_n\mathbf{m}_n + \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}+\mathbf{B}\mathbf{x}_n)\big). \qquad (4.9)$$

Hence, the variational distribution $q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}, \boldsymbol{\theta})$ turns out to be a multivariate Gaussian $\mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{S}_n)$ and we denote it by q_n for brevity. It is important to note that $\boldsymbol{\phi}$ can be the parameters of distributions obtained via Laplace approximations or via the neural network. We now derive the ELBO objective that we need to maximize. For clarity, we also present a worked out example for this derivation in one-dimension for the Poisson lognormal in the Appendix E.

4.6 The Evidence Lower Bound (ELBO)

Now that we have our variational distribution, we can compute the Evidence Lower Bound (ELBO), using the following expression:

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}, \boldsymbol{\psi}_i)}\left[\ln \frac{p(\mathbf{y}_n, \mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta})}{q(\mathbf{z}_n|\mathbf{y}_n, \boldsymbol{\phi})}\right]. \qquad (4.10)$$

We can then simplify the ELBO and since expectations are linear, compute each of the expectations term-wise:

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi})} \Bigg[\ln p(\mathbf{y}_n, \mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\psi}_i) - \ln q(\mathbf{z}_n|\mathbf{x}_n, \mathbf{y}_n, \boldsymbol{\phi}) \Bigg]. \tag{4.11}$$

Now using the chain rule of probability, we can express the joint density as the product of the conditional and the marginal densities:

$$p(\mathbf{y}_n, \mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\psi}_i) = \prod_{i=1}^{D} p\big(y_{ni}|\mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}_i\big) \times \mathcal{N}(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}). \tag{4.12}$$

Taking logarithms on both sides, we can also write this as follows-

$$\ln p(\mathbf{y}_n, \mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\psi}_i) = \sum_{i=1}^{D} \ln p\big(y_{ni}|\mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}_i\big) + \ln \mathcal{N}(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}). \tag{4.13}$$

Taking the expectations on both sides wrt $q_n \equiv q(\mathbf{z}_n|\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi})$, we get:

$$\mathbb{E}_{q_n}\Big[\ln p(\mathbf{y}_n, \mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\psi}_n)\Big] = \mathbb{E}_{q_n}\Bigg[\sum_{i=1}^{D} \ln p\big(y_{ni}|\mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}_i\big) + \ln \mathcal{N}(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta})\Bigg].$$

This gives simplifies the first term of the ELBO which can hence be written as:

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{n=1}^{N} \text{ELBO}_n(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{4.14}$$

where

$$\begin{aligned} &\text{ELBO}_n(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) \\ &\quad = \mathbb{E}_{q_n}\Bigg[\sum_{i=1}^{D} \ln p\big(y_{ni}|\mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}_i\big) + \ln \mathcal{N}(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}) - \ln q(\mathbf{z}_n|\mathbf{y}_n, \boldsymbol{\phi})\Bigg]. \end{aligned} \tag{4.15}$$

This is the expression of the ELBO that we will need to maximize. However, we are yet to use the form of our variational distribution which is a simple and elegant multivariate Gaussian. By leveraging the expressions for the entropy and cross entropy of the multivariate Gaussian, we can simplify this expression.

We can now apply the expectation on each of the expressions due to linearity.

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \Bigg[\underbrace{\mathbb{E}_{q_n}\Big(\sum_{i=1}^{D} \ln p\big(y_{ni} | \mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}\big) \Big)}_{\text{Expected Conditional Log Probability: } \rho(\boldsymbol{\theta},\boldsymbol{\phi})} - \underbrace{\mathcal{H}\Big(\boldsymbol{q}_n, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\Big)}_{\text{Cross Entropy}} + \underbrace{\mathcal{H}(\boldsymbol{q}_n)}_{\text{Entropy}} \Bigg] \tag{4.16}$$

where $\boldsymbol{q}_n = q(\mathbf{z}_n | \mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\phi}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_n, \mathbf{S}_n)$. Notice that the second and the third term in this expression are precisely the definitions of cross-entropy and entropy of mulitvariate Gaussians which we outline in the Appendix D. We define the first term to be the *Expected Conditional Log Probability* denoted by $\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi})$. Unlike the entropy and the cross-entropy, this term depends on the distribution of the response variable. Before we move on to outlining the form of the expected conditional log probability for various response variable distributions, we will see the explicit expressions for the entropy and the cross-entropy for completeness.

Using the expression of the cross entropy between multivariate Gaussians in (D.16), we write and simplify the second term explicitly. Since our final goal is to maximize this as an objective function, we omit computations involving the constants like D and $\ln(2\pi e)$ which do not affect our optimization:

$$\mathcal{H}\Big(\mathcal{N}(\boldsymbol{\mu}_n, \mathbf{S}_n), \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\Big) = \frac{1}{2}\Big[\ln|\boldsymbol{\Sigma}| + (\boldsymbol{\mu}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_n - \boldsymbol{\mu}) + \text{Tr}\left\{\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right\}\Big] + \text{const.}$$

Similarly for the third term, we use the expression of the entropy of the Gaussian as outlined in (D.10)

$$\mathcal{H}\Big(\mathcal{N}(\boldsymbol{\mu}_n, \mathbf{S}_n)\Big) = \frac{1}{2}\ln|2\pi e \boldsymbol{\Sigma}_n| = \frac{1}{2}\ln|\boldsymbol{\Sigma}_n| + \text{const.}$$

Hence, the final expression of the ELBO that we need to maximize is the following:

$$\begin{aligned} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \Bigg[& \mathbb{E}_{q_n}\Big(\sum_{i=1}^{D} \ln p\big(y_{ni} | \mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}\big) \Big) \\ & - \frac{1}{2}\Big[\ln|\boldsymbol{\Sigma}| + (\boldsymbol{\mu}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_n - \boldsymbol{\mu}) + \text{Tr}\left\{\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right\}\Big] \\ & - \frac{1}{2}\ln|\boldsymbol{\Sigma}_n| \Bigg]. \end{aligned} \tag{4.17}$$

4.7 The Expected Conditional Log Probability

We now simplify the first term with the conditional expectation-

$$\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{n=1}^{N} \mathbb{E}_{\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{m}_n, \mathbf{S}_n)} \left[\sum_{i=1}^{D} \ln p\big(y_{ni} | \mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}\big) \right] \tag{4.18}$$

$$= \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{z_{ni} \sim \mathcal{N}(m_{ni}, S_{nii})} \left[\ln p\big(y_{ni} | \mu_i^y = g^{-1}(z_{ni}), \boldsymbol{\theta}, \boldsymbol{\psi}\big) \right] \tag{4.19}$$

$$= \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{u \sim \mathcal{N}(0,1)} \left[\ln p\big(y_{ni} | \mu_i^y = g^{-1}(m_{ni} + A_{nii} u), \boldsymbol{\theta}, \boldsymbol{\psi}\big) \right] \tag{4.20}$$

where $A_{nii} = \sqrt{S_{nii}}$. Note that here $g^{-1}(\cdot)$ is the inverse link function while $g(z_{ni})$ is the i'th component of the sample $\mathbf{z}_n$. Here we use the fact that if $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then, $z_i \sim \mathcal{N}(\mu_i, \Sigma_{ii})$ to pull out the second summation in (4.19) and the reparametrization trick to sample from a parameter free normal distribution in (4.20). Hence, except for $\rho(\boldsymbol{\theta}, \boldsymbol{\phi})$, for which we *might* consider using Monte-Carlo estimates, we have obtained closed form expressions for all of the terms that comprise the ELBO. The derivations for the different forms of the conditional expectations $\rho(\boldsymbol{\theta}, \boldsymbol{\phi})$ for different response variable distributions such as the Binomial, Bernoulli, Gamma, Gumbel, and others in the Appendix C. However, since the integral is one-dimensional we can completely bypass the need for using Monte-Carlo estimates by exploiting the idea of the Gauss-Hermite Quadrature, which helps us evaluate such integrals to very high accuracy.

4.7.1 Gauss-Hermite Quadrature

Gauss-Hermite quadrature is a numerical integration technique used to approximate the definite integral of a given function using a finite number of function evaluations. It is particularly useful for functions that exhibit oscillatory or rapidly decaying behavior, as it uses a set of weighted points that are optimized for such functions.

The method is based on transforming the integral to a simpler form that can be easily approximated using a set of orthogonal polynomials. In the case of Gauss-Hermite quadrature, the transformation involves replacing the integral of a function $f(x)$ over the interval $(-\infty, +\infty)$ with an integral of the form $g(x)\exp(-x^2)$ over the same interval, where $g(x)$ is a polynomial of degree $n - 1$.

In order to obtain good accuracy it is not even necessary to use the adaptive version that would require determining the mode and second derivative of the integrand each time the parameters

change. Choosing I sampling points, we can use the approximation

$$\mathbb{E}_{z_{ni}\sim\mathcal{N}(\boldsymbol{m}_{ni},\mathbf{S}_{nii})}\left[\ln p\big(y_{ni}|\mu_i^y=g^{-1}(z_{ni}),\boldsymbol{\theta}\big)\right] \approx \frac{1}{\sqrt{\pi}}\sum_{\iota=1}^{I} w_\iota \ln p\big(y_{ni}|\mu_i^y=g^{-1}(m_{ni}+\sqrt{2}A_{nii}z_{ni}),\boldsymbol{\theta}\big) \tag{4.21}$$

where $A_{nii} = \sqrt{S_{nii}}$ and z_ι and w_ι are fixed sample points and weights for Gauss-Hermite quadrature, which can be precomputed for any desired I. Even values as low as 6 will already yield highly accurate results, with values around 15 typically reaching float precision. The form in equation (4.21) can be automatically differentiated to yield any desired gradient.

For the case of Poisson distribution, the $g^{-1}(z) = e^z$ and

$$\rho(\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\psi}) = \sum_{i=1}^{D}\sum_{n=1}^{N}\left[y_{ni}m_{ni} - e^{m_{ni}+S_{nii}/2}\right] + \text{const} \tag{4.22}$$

we notice that we can compute the expectations exactly and hence would not need to use the Gauss-Hermite quadrature.

4.8 Optimizing the ELBO

We have implemented the optimization of the evidence lower bound (ELBO) in PyTorch using automatic differentiation with respect to the model parameters $\boldsymbol{\mu}$, $\mathbf{L}$, $\mathbf{D}$, and $\mathbf{B}$, as well as the variational parameters $\boldsymbol{\phi}$, leveraging both the ADAM [31] and the LBFGS [32] optimizers. Each optimizer has its own unique advantages and drawbacks, and we have decided to compare their performance on our specific problem.

ADAM is a stochastic gradient descent-based optimizer that utilizes an exponentially decaying average of past gradients and squared gradients. Its adaptive learning rate and momentum have proven effective for a variety of deep learning problems.

In contrast, LBFGS is a quasi-Newton method that approximates the inverse Hessian matrix by leveraging the gradients of the objective function. This optimizer is known to converge to the optimal solution more quickly than stochastic gradient descent-based methods, particularly for problems with a large number of parameters. However, it requires more memory than ADAM.

Our experiments have demonstrated that LBFGS converges more quickly than ADAM for our problem, which has a relatively small number of parameters. Therefore, we have selected LBFGS as our optimizer of choice for our experiments.

While speed is our key objective, we recognize that convergence issues may arise in certain situations. As such, we have carefully chosen the ADAM as a backup option to ensure the most efficient optimization of our models. This approach provides us with the necessary flexibility and adaptability to achieve the best possible results, even in the face of unforeseen challenges.

4.8.1 Initialization of the parameters

Initialization in an optimization procedure is crucial because it can greatly affect the performance and convergence of the optimization algorithm. The initialization step sets the initial values of the parameters being optimized, which are then updated through the optimization process to maximize the ELBO objective.

If the initial values are chosen poorly, the algorithm may get stuck in local optima or take a longer time to converge to the global optima or even sometimes diverge. A good initialization provides a good starting point for the algorithm and can help it converge faster and more reliably- we use method of moment's estimator for initializing the parameters $\boldsymbol{\mu}$, $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{B}$.

For a Poisson lognormal distribution, the moments are as follows:

$$\mathbb{E}\big[y_j\big] = e^{\mu_j + \Sigma_{jj}/2}, \tag{4.23}$$

$$\mathbb{V}\big[y_j\big] = \mathbb{E}y_j + \left(e^{\Sigma_{jj}} - 1\right)\left(\mathbb{E}\big[y_j\big]\right)^2, \tag{4.24}$$

$$\mathrm{Cov}\big[\,(y_j, y_k)\,\big] = \left(e^{\Sigma_{jk}} - 1\right)\mathbb{E}\big[y_j\big]\mathbb{E}\big[y_k\big]. \tag{4.25}$$

where the data is given as $Y = (y_1, y_2, \cdots, y_n)$ (and $X = (x_1, x_2, \cdots, x_n)$ for regression) Hence, we can compute

$$(\boldsymbol{\mu})_j + \mathbf{B}\mathbf{x}_j = \ln\left(\bar{y_j}\right) - \frac{1}{2}\ln\left(\frac{\mathrm{Var}(y_j) - \bar{y_j}}{\bar{y_j}^2} + 1\right)$$

and we can estimate the covariance matrix $\boldsymbol{\Sigma}$ using:

$$\boldsymbol{\Sigma}_{jj} = \ln\left(\frac{\mathrm{Var}(y_j) - \bar{y_j}}{\bar{y_j}^2} + 1\right)$$

and

$$\boldsymbol{\Sigma}_{jk} = \ln\left(\frac{\mathrm{Cov}(y_j, y_k)}{\bar{y_j}\bar{y_k}} + 1\right).$$

Using this, we have an rough estimate of the quantities $\mathbf{\Sigma}$ and $\boldsymbol{\mu} + \mathbf{Bx}$. Note that this is just a crude estimate and it might even be possible that the $\mathbf{\Sigma}$ estimated from this is not positive definite. But estimating the quantities $\boldsymbol{\mu}$, $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{B}$ accurately from just these two values is not possible, we resort to using some heuristic procedures.

Since there can be multiple combinations of $\mathbf{L}$ and $\mathbf{D}$ which give the same $\mathbf{\Sigma}$ satisfying the relation $\mathbf{LL}^T + \mathbf{D} = \mathbf{\Sigma}$, we do the following procedure to arrive at a suitable estimate:

1. Initialize $\mathbf{L}$ using samples $\mathcal{N}(0, 1)$ and set $\mathbf{D} = \mathbf{\Sigma}_{jj}$.
2. Run a preliminary optimization on $\mathbf{L}$ and $\mathbf{D}$ by minimizing the Frobenius norm of the quantity $\mathbf{LL}^T + \mathbf{D} - \mathbf{\Sigma}$.

This will give us suitable starting values for $\mathbf{L}$ and $\mathbf{D}$ since the norms of the true $\mathbf{L}$ and $\mathbf{D}$ should be close to these values. To implement the optimization procedure described above, we use the ADAM optimizer, because although LBFGS may reach a solution quicker, it is by far more important to start the central optimization using good initial values.

4.8.1.1 Initializing B and $\boldsymbol{\mu}$

For initializing $\boldsymbol{\mu}$ and $\mathbf{B}$, we tried two different approaches. The first was a random initialization where the values of $\boldsymbol{\mu}$ and $\mathbf{B}$ are drawn from a standard normal distribution $\mathcal{N}(0, 1)$. This approach was observed to fail at a lot of instances and the model could not converge. The second approach has been observed to be quite robust, which was to initialize $\boldsymbol{\mu}$ by fitting a linear model to $\ln(1 + \mathbf{Y})$ and then extracting the regression coefficients $\boldsymbol{\mu}_0$ and the variance covariance matrix of the Pearson residuals $\mathbf{\Sigma}_0$. We now set $\boldsymbol{\mu} = \boldsymbol{\mu}_0$ and $\mathbf{\Sigma} = \mathbf{\Sigma}_0^{(Q)}$ where $\mathbf{\Sigma}_0^{(Q)}$ is the best Q rank approximation, obtained by retaining the first Q dimensions of the singular value decomposition of $\mathbf{\Sigma}_0$.

4.8.1.2 Initializing ϕ and the network weights

The variational parameter vectors $\boldsymbol{\phi}$ are initialized as follows:

$$\boldsymbol{\phi}_0 = 1, \boldsymbol{\phi}_1 = -3, \boldsymbol{\phi}_2 = 1, \boldsymbol{\phi}_3 = -3.$$

These values are rough point estimates chosen by inspecting the equations (4.5). $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_3$ are set to negative values to prevent the model likelihood from becoming too peaked and overconfident if it ends up in a local optima. When using a neural network, the network is initialized with zero or low weights.

4.8.2 Regularization

Regularization is primarily used to address the problem of overfitting, which occurs when a model learns to fit the training data too well and fails to generalize to new, unseen data. Additionally, it also imposes constraints on the model parameters, preventing divergence and improving robustness.

For our purposes, we use the L^2 regularization, also known as *weight decay*. L^2 regularization adds a penalty term to the objective function that is proportional to the sum of the squares of the model parameters. This penalty term encourages the model to learn smaller parameter values, which can help prevent overfitting by reducing the model's capacity to fit the training data too precisely. L^2 regularization can also be interpreted as a form of Bayesian prior on the model parameters, where the prior distribution is assumed to be a zero mean Gaussian distribution. This Gaussian prior is applied on the $\ln \boldsymbol{D}$ since it was observed that $\boldsymbol{D}$ often takes too large values on bad initializations. Hence we optimize the function:

$$\text{ELBO}_{reg}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda(t)||\ln D||^2.$$

Typically, $\lambda(t)$ depends on the iteration t allowing the function to start off with high values and gradually decay so that the influence of the prior does not affect the optimization to a large degree. For our purpose, we have use the decay function as

$$\lambda(t) = \lambda(0)\exp(-bt)$$

where $\lambda(0) \approx 30$ and $b \approx 2$.

4.9 Time Complexity

We have done a detailed discussion of the time complexity of our algorithm in Appendix D. The time complexity of our algorithm is $O(NDK^2 + NQD) \approx O(NDK^2)$. Our method needs about $t = 1000$ iterations to converge and we perform approximately 50 operations in every iteration. If we process 8 operations by AVX per clock cycle of 0.4 ns, we take a time per NDK^2 of $1000 \times 50 \times 5\text{E}{-11}\,\text{s} = 2.5\text{E}{-5}\,\text{s}$, which should be about 100 times faster than *Chiquet et al.*, although our main goal has been to be more accurate than them by not making the assumption that $p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta})$ has diagonal covariance.

Chiquet et al. should have the time complexity of $O(NSDK)$ per iteration, S is the number of samples, where they need 1200 seconds on a single core for a dataset of $N = 155$ samples with $D = 4000$ variables and $K = 20$ latent factors. (They denote N, D and K with n, p, q.) This means they take 1E-4 seconds per NDK.

In the following section, we will see the results that we obtain by using our methodology on a variety of datasets.

Chapter 5

Results

"Make better plots."

Amelie Hilger

5.1 Introduction

In this chapter, we present a sample demonstration of our method, perform a simulation study and subsequently evaluate our model on real datasets. We designed the simulation study to assess the performance of our proposed model under various scenarios with different hyper-parameter settings where we generated synthetic datasets based on different underlying models and evaluated our model's performance in terms of accuracy and compared with *PLNmodels*. Additionally, we tested our model on three real-world datasets: the Barents fish dataset, the Trichoptera dataset and the Oaks Amplicon dataset, as these datasets were also utilized by Chiquet et al. in their papers. In the following sections, we present a sample demonstration, the detailed experimental setup and a comparison of the results on the simulated and real datasets.

Remark 5.1. Since both the methods estimate the covariance matrix $\boldsymbol{\Sigma}$, to compute the correlation matrix $\boldsymbol{\rho}$, for comparison, we extract $\boldsymbol{D}^{-1/2} = \sqrt{\text{diag}(\boldsymbol{\Sigma})}^{-1}$ and then use $\boldsymbol{\rho} = \boldsymbol{D}^{-1/2}\boldsymbol{\Sigma}\boldsymbol{D}^{-1/2}$

5.2 Sample Demonstration

To demonstrate the efficacy of our inference approach in modeling correlated count data, we employed it in the most rudimentary manner on a representative dataset. The dataset used in this analysis consists of $N = 50000$ observations of $D = 4$ dimensional correlated count data generated from a Poisson lognormal distribution with a known mean $\boldsymbol{\mu}$ and dispersion parameter $\boldsymbol{\Sigma}$ without any covariates by setting the regression weights $\mathbf{B} = \mathbf{0}$.

We use the Laplace approximation for amortization instead of the neural network and start the optimization with a random initialization of the parameters. $\boldsymbol{\mu}$ and $\mathbf{L}$ are initialized by sampling from a $\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.1$ while $\mathbf{D}$ is initialized by first sampling from $\mathcal{N}(0, \sigma^2)$ and then taking the exponential of the values since $\mathbf{D}$ is constrained to be positive. Using the ADAM optimizer we perform this training for $t = 1000$ iterations.

5.2.1 Results

Fig. (5.1) illustrates the convergence of the model parameters, the variational parameters and the ELBO. Observe that the ELBO converges rather quickly to a maximum, within $t = 100$ iterations, after which the convergence gradually slows down. The model and the variational parameters however need the entire training to reach convergence.

We compare the correlation matrix of the raw counts $\mathbf{Y}$ and the correlation matrices estimated by our proposed method and that by *PLNmodels* in Fig. (5.2) and find that our model captures correlation better than *PLNmodels*.

Metric	Our Model	PLNmodels
μ_{MSE}	0.073	3.158
μ_{bias}	0.152	0.641
Σ_{MSE}	11.31	175.368
Σ_{bias}	0.320	2.13

TABLE 5.1: Comparison of our method with PLNmodels on sample dataset.

Table 5.1 provides a detailed comparison of our method with *PLNmodels* on the sample dataset. The metrics used for evaluation are μ_{MSE} (mean squared error of the mean parameter), μ_{bias} (mean bias of the mean parameter), Σ_{MSE} (mean squared error of the dispersion parameter), and Σ_{bias} (mean bias of the dispersion parameter).

Our model demonstrates remarkable performance, as evidenced by the values reported in the table. The μ_{MSE} for our method is significantly lower (0.073) compared to *PLNmodels* (3.158),

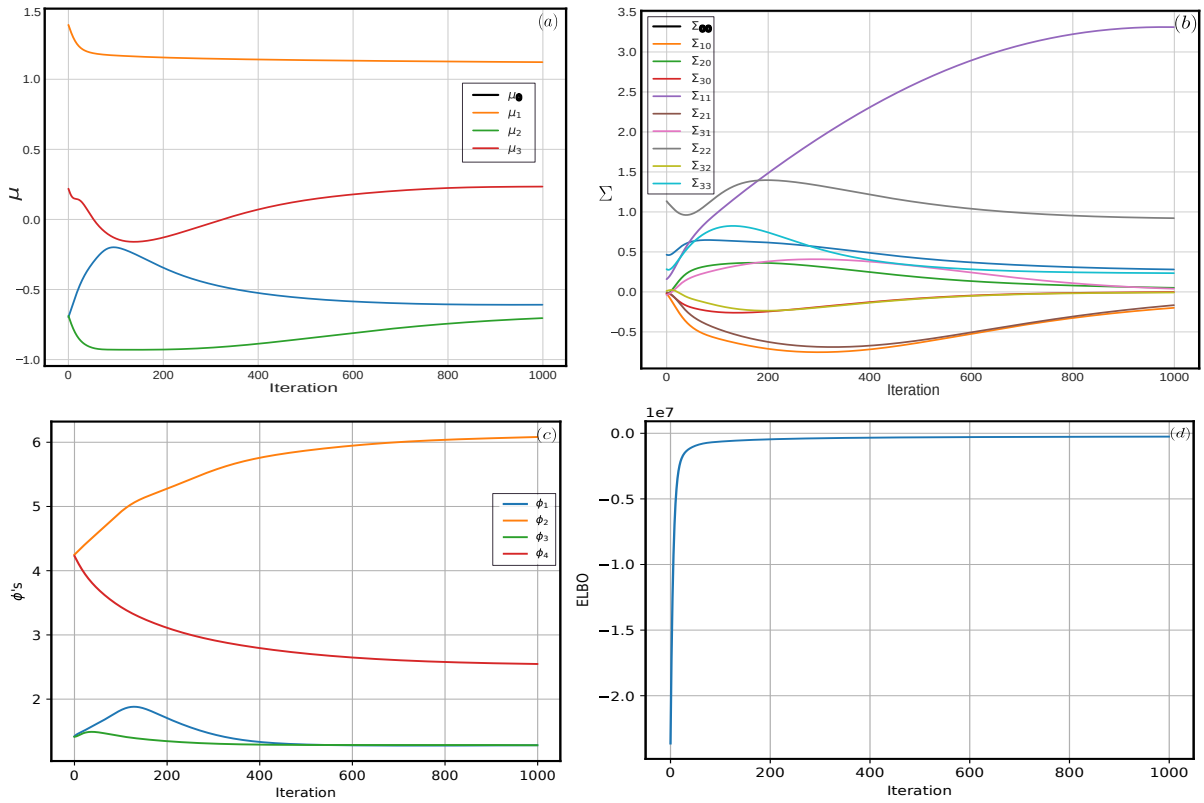


FIGURE 5.1: Demonstration on sample dataset- (a) Convergence of μ (b) Convergence of Σ (c) Convergence of $|||\phi|||$'s (d) Convergence of ELBO.

indicating that our model provides more accurate estimates for the mean parameter. Moreover, the μ_{bias} of our model (0.152) is substantially lower than that of *PLNmodels* (0.641), indicating reduced bias in the estimation of the mean parameter.

In terms of the dispersion parameter, our model also outperforms *PLNmodels*. The Σ_{MSE} value for our method is 11.31, considerably lower than *PLNmodels* (175.368), suggesting superior accuracy in estimating the dispersion parameter. Similarly, the Σ_{bias} for our method (0.320) is significantly lower than *PLNmodels* (2.13), indicating reduced bias in the estimation of the dispersion parameter.

Overall, the results in Table 5.1 clearly demonstrate the superiority of our method over *PLNmodels* in terms of both mean absolute error (MAE) and bias across the estimated model parameters. These findings serve as strong evidence for the effectiveness and accuracy of our approach. However, to further validate the robustness and performance of our method, we conducted a comprehensive simulation study, which will be discussed in the following sections.

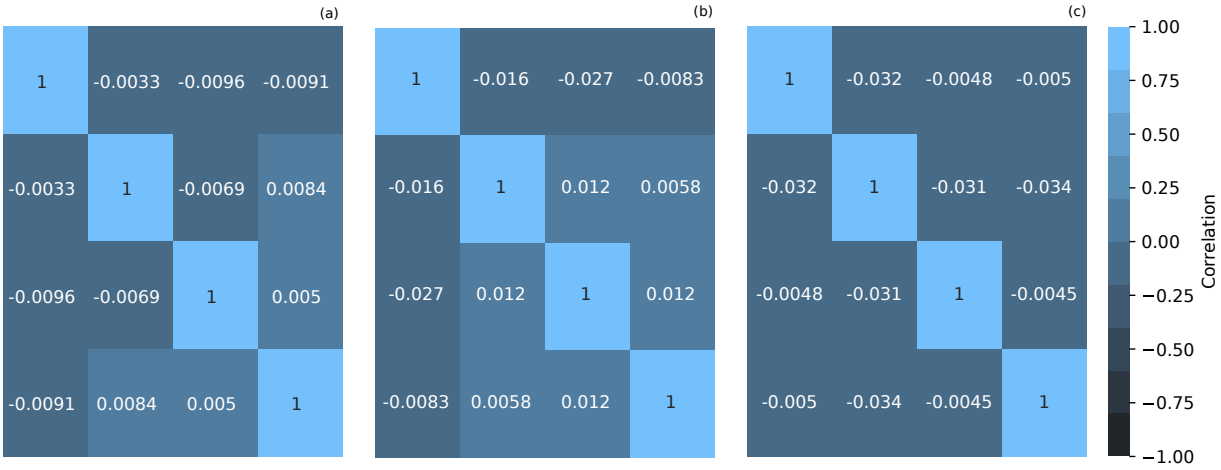


FIGURE 5.2: Correlation heatmaps of sample dataset: (a) Correlation matrix with which the count data $\mathbf{Y}$ where generated. (b) Correlation matrix derived using our method. (c) Correlation matrix derived using PLNmodels.

5.3 Simulation Study

To thoroughly evaluate the performance of our method and demonstrate its superiority, we conducted a simulation study based on a Poisson lognormal model. The simulation study encompassed various difficulty settings that mimic real-world scenarios.

In our simulations, we considered different combinations of parameters, including the number of samples $N \in \{50, 500, 1000\}$, the number of dimensions $D \in \{2, 10, 90\}$, and the number of covariates $Q \in \{2, 5, 10\}$. These parameter choices cover a wide range of possibilities, spanning from challenging scenarios with a small number of samples and a high-dimensional feature space ($N = 50$, $Q = 10$, $D = 90$) to simpler settings with larger sample sizes and lower dimensionality ($N = 1000$, $Q = 2$, $D = 2$). By exploring this diverse range of parameter values, we aimed to comprehensively assess the performance of our model across different data complexities. For our model, we use a shallow neural network with $L = 2$ layers having $H = 6$ neurons each for learning the variational parameters and choose $K = D$ to compare our results fairly with the full rank approximation as done by *PLNmodels*. For each parameter setting, we generated synthetic datasets by first creating the covariate matrix $\mathbf{X}$, where the entries were sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$. Subsequently, we generated a collection of $R = 50$ count matrices $\left(\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \cdots, \mathbf{Y}^{(R)}\right)$, representing replicate datasets, to ensure robustness and reliability in our evaluations.

The design of our simulation study was motivated by the work of *Chiquet et al.*, as we aimed to provide a direct comparison to their method and replicate their framework. By adopting a similar simulation setup, we could effectively demonstrate the superiority of our model while

ensuring consistency in the evaluation process. Moreover, the chosen parameter settings in our simulation study align with those commonly observed in real datasets, enabling us to assess the performance of our method under realistic conditions.

5.3.1 Results

In our evaluation, we assessed the accuracy of parameter estimates for both the mean vector $\boldsymbol{\mu}$, the dispersion matrix $\boldsymbol{\Sigma}$, as well as for the regression coefficients $\mathbf{B}$ for our method and *PLNmodels* on a set of simulated datasets. To quantify the differences between the estimated and true parameter values, we computed the total absolute element-wise errors as $\sum_{r=1}^{R} |\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}|$, where R represents the number of replicates.

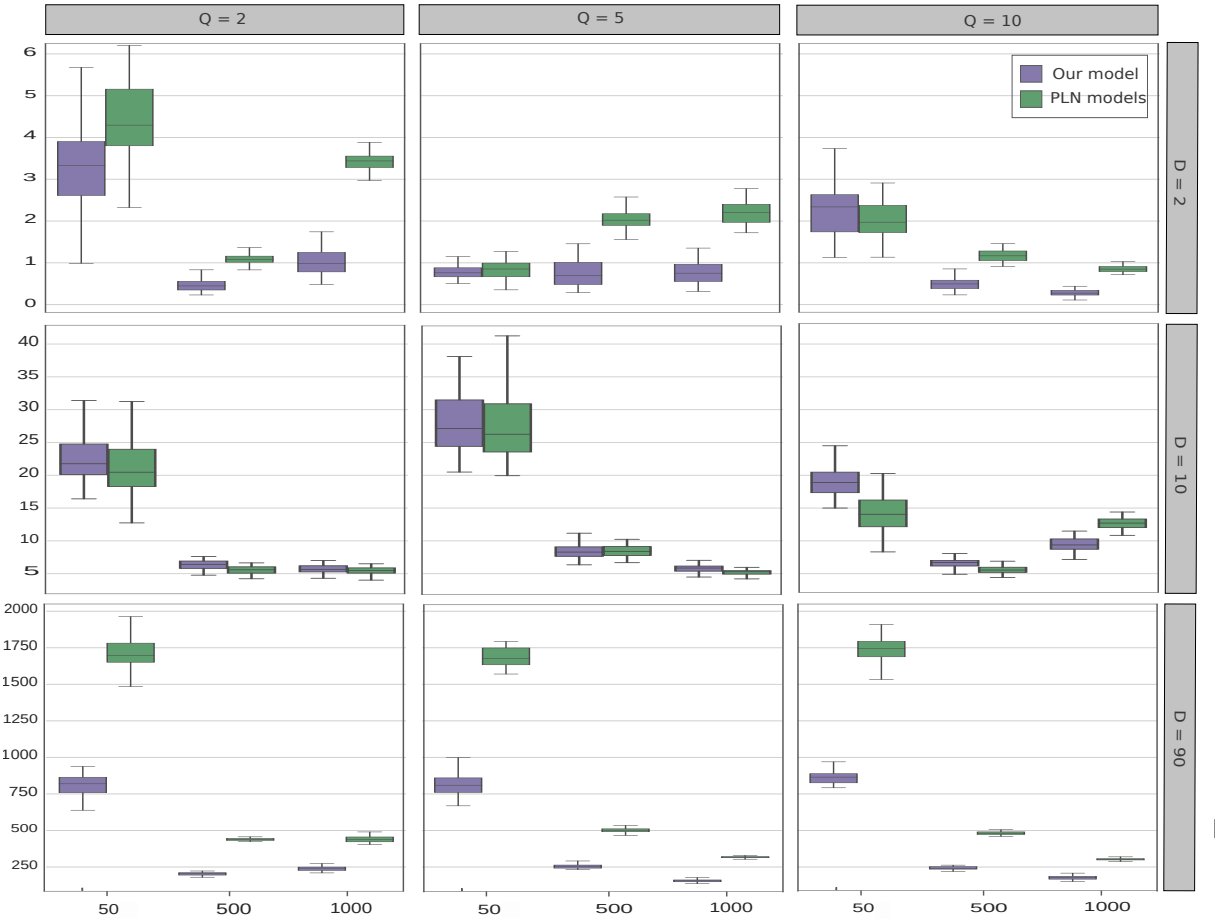


FIGURE 5.3: Comparison of our method with PLNmodels with different parameter settings- $D = \{2, 10, 90\}$, $Q = \{2, 5, 10\}$ and $N = \{50, 500, 1000\}$: Box plots of total absolute difference of the parameter estimates and true parameter values computed over $R = 50$ replicates.

We observe that our method outperforms *PLNmodels* consistently across all by a large margin. except two where the performance is practically same, setups, as evidenced by the lower error values. The impact of our model's performance is majorly noticeable when the dimensionality of the counts are very high, as can be seen in the last row of Fig. (5.3).

5.4 Real Datasets

In the following sections, we illustrate the application of our method on a few real world datasets. The selection of these datasets was based on the need to demonstrate the superior performance of our method over *PLNmodels*. A concise overview of each dataset is provided, highlighting its inherent characteristics and complexities. We then conduct a brief comparative analysis of the results obtained from our method with those achieved by *PLNmodels*, particularly by studying correlation matrices. However, comparing performance on real datasets poses challenges as we do not have the knowledge of the true parameter values a priori.

5.4.1 Barents Fish Dataset

The Barents fish dataset is a study describing the assemblages and distributions of fish species in the southwestern and central part of the Barents Sea [33]. The dataset has $N = 89$ observations, primarily contains information for $D = 30$ fish species and shrimp identified in the period 1997-1999 along with the $Q = 4$ covariates: latitude, longitude, temperature and depth.

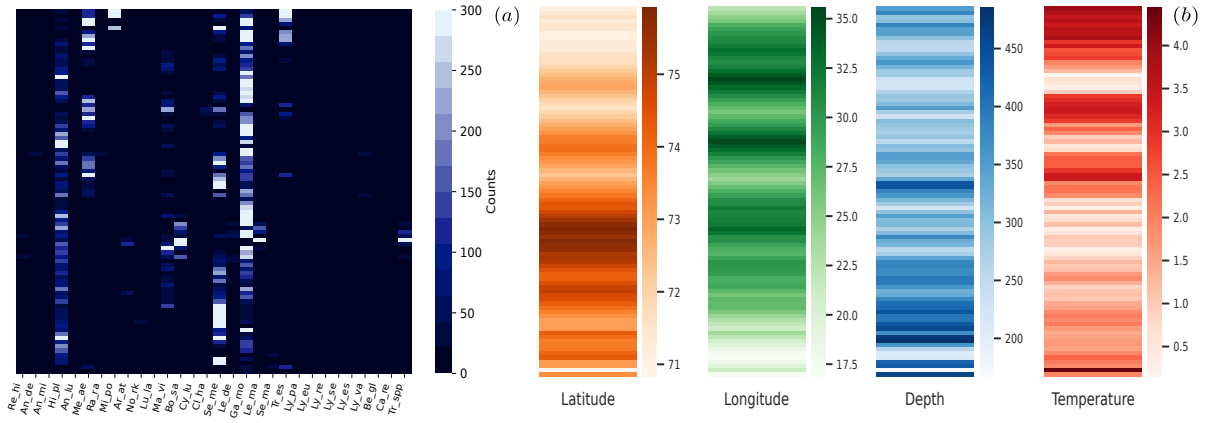


FIGURE 5.4: Barents dataset: (a) Count heatmap (b) Covariate heatmap.

We focus on the parameter estimate of $\mathbf{\Sigma}$ for both the methods and look at the correlation matrix computed in Fig. 5.5 using the remark 5.1. We notice that the correlation matrix estimated

by PLNmodels are largely overestimated as compared to our model. This demonstrates the superiority of our modelling in accurately estimating correlations among the species.

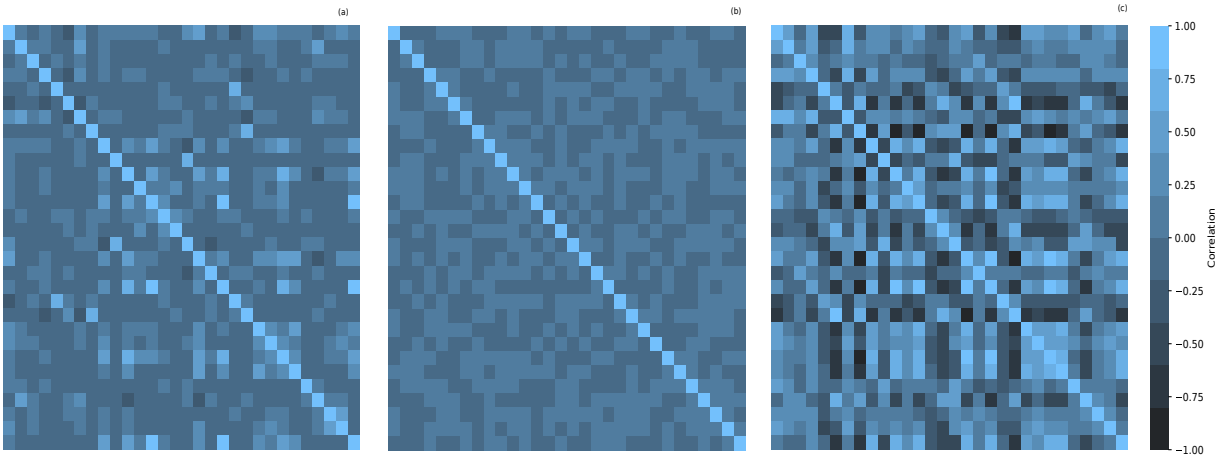


FIGURE 5.5: Correlation heatmaps of Barents dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels.

5.4.2 Trichoptera Dataset

The ecological data set trichoptera [34] describes the counts of 17 Trichoptera species, accompanied with some covariates (meteorological factors) which that may influence their presence during the sampling. The dataset has 49 observations and 7 covariates: temperature, wind, pressure, humidity, cloudiness, precipitation and the group. It is worth noting that group is a factor and not a continuous covariate.

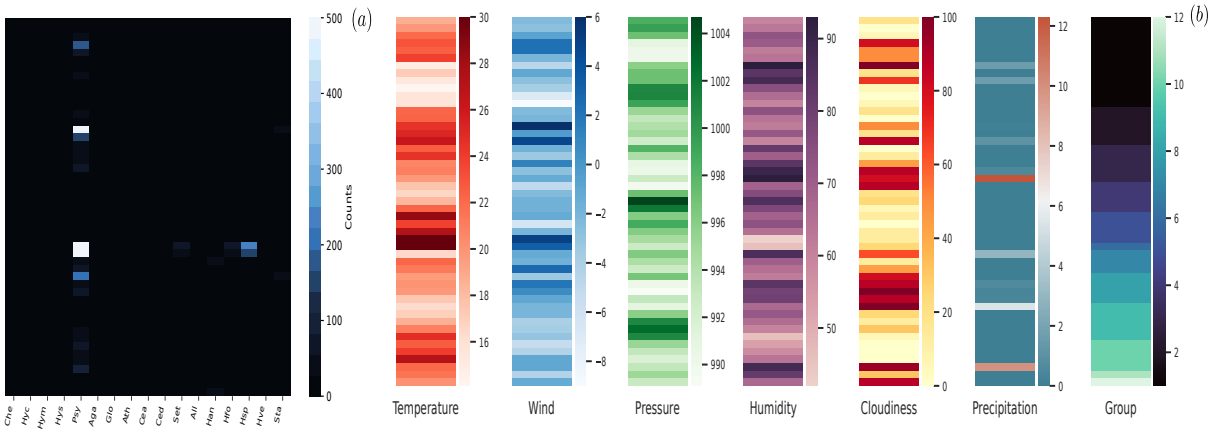


FIGURE 5.6: Trichoptera Dataset: (a) Count Heatmap (b) Covariate Heatmap.

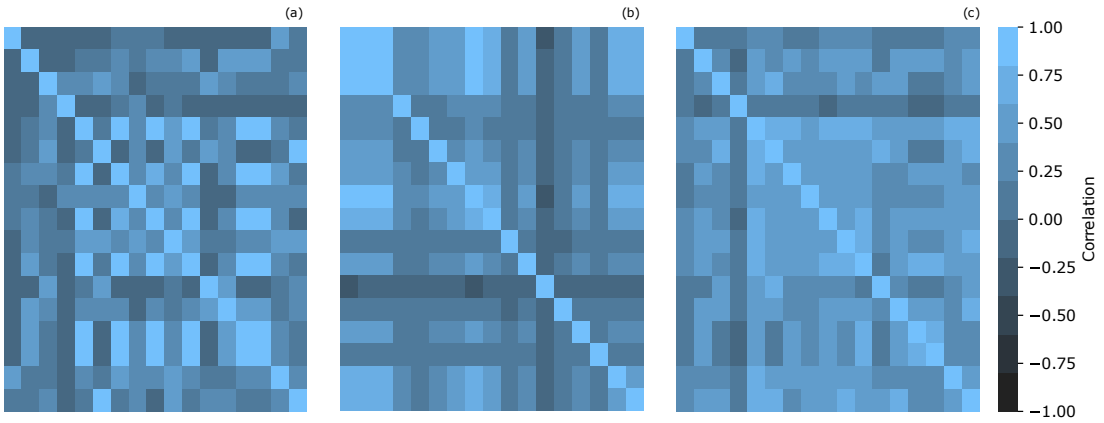


FIGURE 5.7: Correlation heatmaps of Trichoptera dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels.

We observe the correlation matrices estimated by both the models in Fig. (5.7).

5.4.3 Oaks Amplicon Dataset

The Oaks Amplicon dataset [35] includes information on the abundance of 114 taxa, comprising of 66 bacterial OTUs (Operational Taxonomic Unit) and 48 fungal OTUs, across a total of 116 samples. In addition to this, for each sample, there is information available on 11 additional covariates.

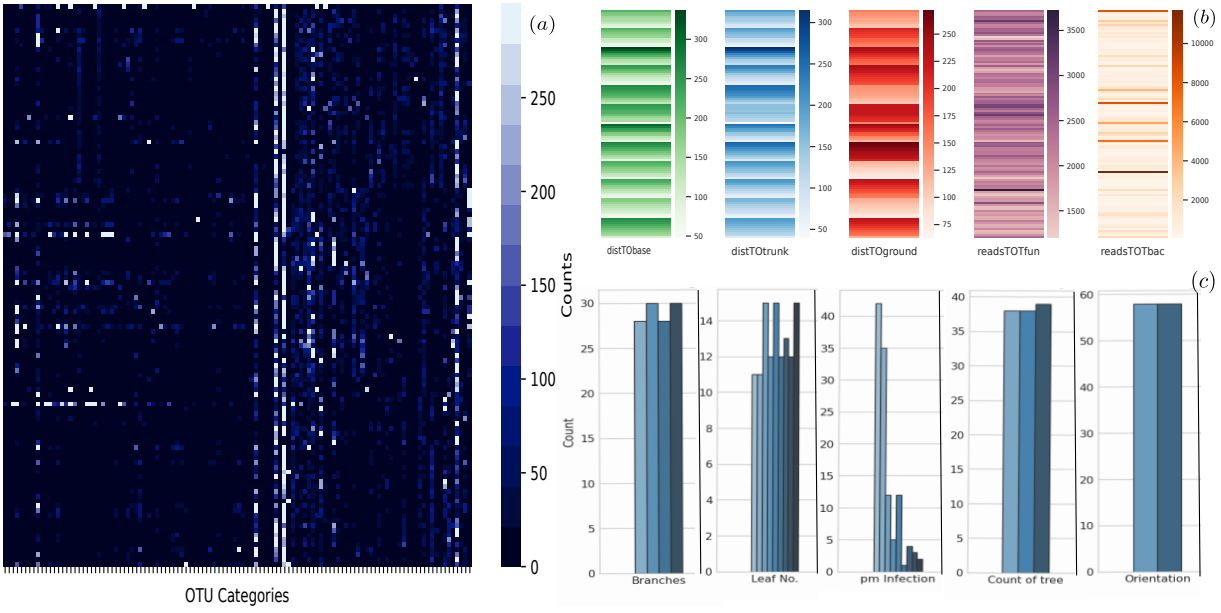


FIGURE 5.8: Oaks Amplicon Dataset: (a) Count Heatmap (b) Covariate Heatmap (c) Factor Histogram.

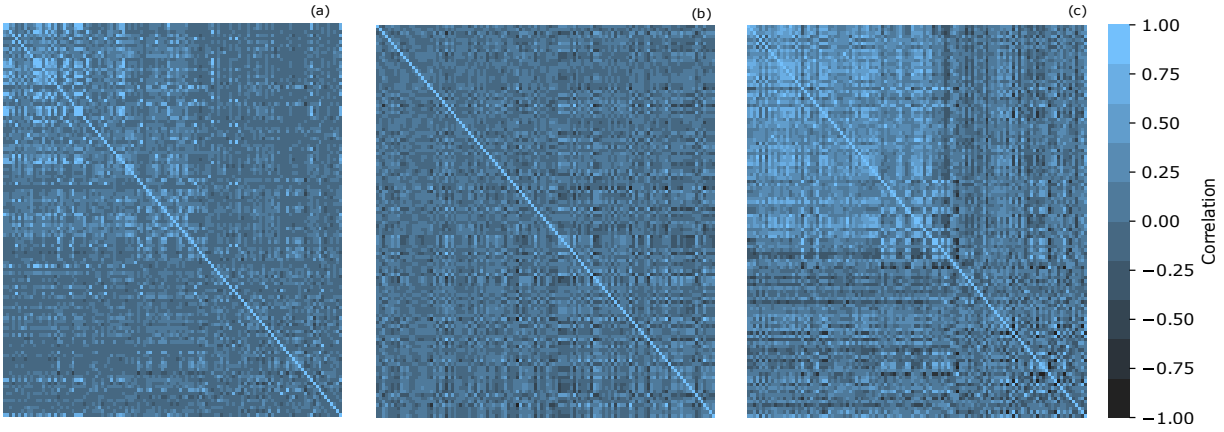


FIGURE 5.9: Correlation heatmaps of Oaks dataset: (a) Correlation matrix of the counts Y. (b) Correlation matrix of our method. (c) Correlation matrix of PLNmodels.

For both the figures Fig. (5.7) and Fig. (5.9), we again observe that the correlations estimated by *PLNmodels* is quite high as compared to our model.

5.5 Runtime Analysis

In addition to evaluating the accuracy of our method compared to *PLNmodels*, we also conducted a runtime analysis to assess the computational efficiency of our approach. Figure 5.10 illustrates the comparison of the average runtime between our method and *PLNmodels*. The results demonstrate the time difference between the two approaches and highlight potential areas for optimization.

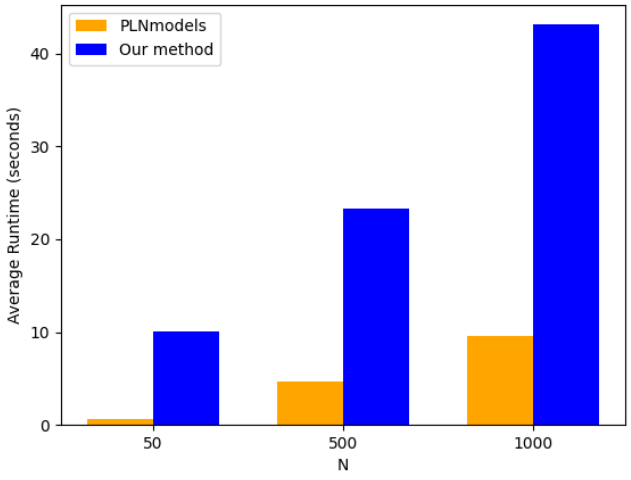


FIGURE 5.10: Comparison of average runtime of our method and PLNmodels.

Currently, our software takes an average of 40 seconds for inference with a dataset containing 1000 samples. This runtime is approximately 30 seconds longer than that of *PLNmodels*. Upon conducting profiling analysis, we identified that the increased runtime in our method is primarily due to repeated function calls to 'torch.einsum', which are not fully optimized for certain computations in PyTorch [36].

It is worth noting that while *PLNmodels* is implemented in R/C++, our software is developed in Python and utilizes the PyTorch framework. The difference in runtime between the two methods can be attributed to the underlying computational optimizations and language-specific efficiencies. However, this runtime discrepancy also presents an opportunity for further improvement and optimization in our method.

Chapter 6

Conclusion and Future Work

> "Statistics may give you a conclusion, but they'll never give you the joy of discovering your data for yourself...or the frustration of trying to make sense of it all."

6.1 Conclusion

In this study, we presented a novel approach for latent factor analysis and regression of multivariate correlated count data generated via the Poisson lognormal model. We compared our proposed method with the state-of-the-art method presented by *Chiquet et al* in their software *PLNmodels*, and found that our method significantly outperformed theirs in terms of accuracy. Specifically, our parameter estimates led to a 50% reduction in mean absolute error compared to the *Chiquet et al.* method. Furthermore, the simulation study included datasets with varying levels of difficulty, making our results particularly compelling for real-world applications. These findings demonstrate the superiority of our model for analyzing correlated count data.

We also acknowledge some shortcomings of our methodology. To model real datasets, we are yet to design a method to include factors in our model. Additionally, in practice, our method was found to be slower than the method proposed by *Chiquet et al.*, despite having better time complexity. Our computational time is more than PLNmodels, albeit by the order of seconds/few minutes which is optimizable according to our time complexity.

6.2 Future Work

Future work for our research includes enhancing the inference algorithm's robustness to bad initial values, which can hinder convergence. As we have observed, different initializations can lead to divergent outcomes during optimization. To address this, we will investigate alternative initialization techniques, such as adaptive step sizes and annealing schedules, to improve the algorithm's ability to recover from such situations and converge to the optimal solution. Once we have a robust initialization procedure, using LBFGS can speed up our optimization by orders of magnitude.

We also plan to do a comparative analysis of both these methods for synthetic datasets with highly correlated variables. For real datasets, we plan to compare the ELBO's of both these methods to observe which one is higher, and therefore better.

Furthermore, we plan to expand the scope of our software by implementing popular response variable distributions like the Bernoulli and Gumbel. This will provide users with a range of GLMMs to fit to their data and select the appropriate one for their needs.

Finally, we aim to implement a principal component analysis method using our framework, which will enable us to automatically determine the number of latent factors, denoted as K, in cases where it is not known a priori which will further increase the utility of our software.

Appendix A

Efficient Implementation and Time Complexity

A.1 Introduction

For matrices $\mathbf{A} \in \mathbb{R}^{K \times L}$ and $\mathbf{B} \in \mathbb{R}^{L \times M}$, matrix multiplication $\mathbf{AB}$ has time a complexity of $O(KLM)$. The time complexity to evaluate F$\mathbf{ABC}$ for $\mathbf{C} \in \mathbb{R}^{L \times M}$ depends on the order of evaluation. For $(\mathbf{AB})\mathbf{C}$ it is $O(KLM + KMN)$ and for $\mathbf{A}(\mathbf{BC})$ it is $O(LMN + KLN)$. These two can differ enormously. Suppose, for example, that $K = L = M$ is huge and $N = 1$, then the first evaluation has complexity $O(M^3)$ while the second one has $O(2M^2)$. Since we need to perform matrix manipulations for computing the ELBO in (4.16), it is important to perform these multiplications in the correct sequence for achieving an optimal time complexity.

A.1.1 Prerequisites

We give a note of some common time complexities that we encounter during this section. Assume that $\mathbf{L} \in \mathbb{R}^{D \times K}$, $\mathbf{P} \in \mathbb{R}^{K \times K}$, $\mathbf{D} \in \mathbb{R}^{D \times D}$ is a diagonal matrix, $\boldsymbol{v} \in \mathbb{R}^D$ is a vector.

- $\mathbf{L}^\top \mathbf{D}$ takes $O(KD)$ and $\mathbf{L}^\top \mathbf{D}\mathbf{L}$ takes $O(DK^2)$
- $\mathbf{D}_1\mathbf{D}_2$, $\mathbf{D}_1 + \mathbf{D}_2$, $\mathbf{D}\boldsymbol{v}$, $\mathbf{D}^{-1}$ and $|\mathbf{D}|$ takes $O(D)$
- $|\mathbf{P}|$, $\mathbf{P}^{-1}$ and $\mathbf{P}_1\mathbf{P}_2$ takes $O(K^3)$

A.1.2 Weinstein–Aronszajn identity.

Computing the determinant of an $N \times N$ matrix takes $O(N^3)$ and can be expensive when N is huge. To compute the determinants efficiently, we use the identity-

$$|\mathbf{I}_N + \mathbf{A}\mathbf{B}^\top| = |\mathbf{I}_K + \mathbf{A}^\top\mathbf{B}| \tag{A.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are $N \times M$ matrices. Now, we only need to find the determinant of a $K \times K$ matrix which can be done in $O(K^3)$.

A.1.3 Woodbury matrix identity.

Just like the determinant, computing the inverse of an $N \times N$ matrix takes $O(N^3)$. To compute inverses efficiently, we use the Woodbury matrix identity-

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1} \tag{A.2}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{U}, \mathbf{V}$ are conformable matrices. When $\mathbf{A}$ and $\mathbf{C}$ are matrices with trivial inverses (for instance diagonal matrices), we would only need to compute the $K \times K$ inverse $\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}$ which takes $O(K^3)$.

A.2 Complexity of Computing the ELBO

For computing the ELBO in (4.16), we need to calculate four expensive terms: $|\mathbf{S}_n|$, $|\mathbf{\Sigma}|$, $\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{S}_n\right)$ and $(\boldsymbol{\mu} - \boldsymbol{m}_n)^T \mathbf{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{m}_n)$ where $\mathbf{S}_n$ and $\mathbf{\Sigma}$ are $D \times D$ matrices and $\boldsymbol{m}_n$ and $\boldsymbol{\mu}$ are D-dimensional column vectors. Since D can be huge like 100, we need to optimize the time complexity for computing each of these four terms.

A.3 Computing $|\mathbf{S}_n|$

Computing $|\mathbf{S}_n|$ naively can take $O(D^3)$. Hence we simplify the expression of $|\mathbf{S}_n|$ for optimizing computation. We first use the Woodbury's identity (A.2) to compute $\mathbf{\Sigma}^{-1}$

$$\begin{aligned}\mathbf{\Sigma}^{-1} &= \big(\mathbf{D} + \mathbf{L}\mathbf{L}^\top\big)^{-1} \\ &= \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{L}\big(\mathbf{I}_K + \mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L}\big)^{-1}\mathbf{L}^\top\mathbf{D}^{-1} \\ &= \mathbf{D}^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top \end{aligned} \tag{A.3}$$

where $\mathbf{C} = \big(\mathbf{I}_K + \mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L}\big)^{-1}$ and $\mathbf{U} = \mathbf{D}^{-1}\mathbf{L}$.

We now use the above to expand $|\mathbf{S}_n|$-

$$\begin{aligned}|\mathbf{S}_n| &= \big|\big(\mathbf{\Lambda}_n + \mathbf{\Sigma}^{-1}\big)^{-1}\big| \\ &= \big|\big(\mathbf{\Lambda}_n + \mathbf{D}^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top\big)^{-1}\big| \\ &= \big|\mathbf{\Lambda}_n + \mathbf{D}^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top\big|^{-1} \\ &= \big|\mathbf{R}_n^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top\big|^{-1} \qquad \text{where } \mathbf{R}_n^{-1} = \mathbf{\Lambda}_n + \mathbf{D}^{-1} \\ &= |\mathbf{R}_n|\big|\mathbf{I}_D - \mathbf{R}_n\mathbf{U}\mathbf{C}\mathbf{U}^\top\big|^{-1} \\ &= |\mathbf{R}_n|\big|\mathbf{I}_K - \mathbf{C}^\top\mathbf{U}^\top\mathbf{R}_n^\top\mathbf{U}\big|^{-1} \\ &= |\mathbf{R}_n|\big|\mathbf{I}_K - \mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\big|^{-1} \end{aligned} \tag{A.4}$$

which takes $O(D)$ for $|\mathbf{R}_n|$, $O(KD)$ for $\mathbf{U}$, $O(DK)$ for $\mathbf{U}^\top\mathbf{R}_n$, $O(DK^2)$ for $\mathbf{U}^\top\mathbf{R}_n\mathbf{U}$ and $\mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L}$, $O(K^3)$ for computing $\mathbf{C}^{-1}$, $\mathbf{C}$, $\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}$ and $\big|\mathbf{I}_K - \mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\big|$.

A.4 Computing $(\mathbf{S}_n)_{ii}$

Applying the Woodbury matrix identity, we get

$$\begin{aligned}\mathbf{S}_n &= (\mathbf{\Lambda}_n + \mathbf{D}^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top)^{-1} \\ &= \mathbf{R}_n + \mathbf{R}_n\mathbf{U}\big(\mathbf{C}^{-1} - \mathbf{U}^\top\mathbf{R}_n\mathbf{U}\big)^{-1}\mathbf{U}^\top\mathbf{R}_n. \end{aligned} \tag{A.5}$$

We only need the diagonal elements of $\mathbf{S}_n$, and the second term can be written as product of two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D\times K}$, which can be computed in $O(DK^2)$,

$$(\mathbf{A}\mathbf{B}^\top)_{ii} = \sum_{j=1}^{K} A_{ij}B_{ij}, \tag{A.6}$$

so the total time complexity is $O(DK^2 + K^3)$.

A.5 Computing $|\boldsymbol{\Sigma}|$

This is a straightforward application of the Weinstein–Aronszajn identity (A.1) and reduces our computation of the $D \times D$ determinant to a $K \times K$ determinant.

$$\begin{aligned}
|\boldsymbol{\Sigma}| &= |\mathbf{D} + \mathbf{L}\mathbf{L}^\top| \\
&= |\mathbf{D}(\mathbf{I}_D + \mathbf{D}^{-1}\mathbf{L}\mathbf{L}^\top)| \\
&= |\mathbf{D}||\mathbf{I}_D + \mathbf{D}^{-1}\mathbf{L}\mathbf{L}^\top| \\
&= |\mathbf{D}||\mathbf{I}_K + \mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L}| = |\mathbf{D}||\mathbf{C}|^{-1}
\end{aligned} \tag{A.7}$$

which takes $O(D)$ for $|\mathbf{D}|$ and $O(K^3)$ for $|\mathbf{C}|$. Note that we have already computed $\mathbf{C}$ above.

A.6 Computing $\mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right)$

We expand $\mathbf{S}_n$ using Woodbury's identity (A.2)

$$\begin{aligned}
\mathbf{S}_n &= \left(\boldsymbol{\Lambda}_n + \boldsymbol{\Sigma}^{-1}\right)^{-1} \\
&= \left(\boldsymbol{\Lambda}_n + \mathbf{D}^{-1} - \mathbf{U}\mathbf{C}\mathbf{U}^\top\right)^{-1} \\
&= \mathbf{R}_n + \mathbf{R}_n\mathbf{U}\left(\mathbf{C}^{-1} - \mathbf{U}^\top\mathbf{R}_n\mathbf{U}\right)^{-1}\mathbf{U}^\top\mathbf{R}_n \\
&= \mathbf{R}_n + \mathbf{R}_n\mathbf{U}\left(\mathbf{I}_K + \mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L} - \mathbf{L}^\top\mathbf{D}^{-1}\mathbf{R}_n\mathbf{D}^{-1}\mathbf{L}\right)^{-1}\mathbf{U}^\top\mathbf{R}_n \\
&= \mathbf{R}_n + \mathbf{R}_n\mathbf{U}\left(\mathbf{I}_K + \mathbf{L}^\top(\mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{R}_n\mathbf{D}^{-1})\mathbf{L}\right)^{-1}\mathbf{U}^\top\mathbf{R}_n \\
&= \mathbf{R}_n + \mathbf{R}_n\mathbf{U}\left(\mathbf{I}_K + \mathbf{L}^\top\mathbf{Q}_n\mathbf{L}\right)^{-1}\mathbf{U}^\top\mathbf{R}_n
\end{aligned} \tag{A.8}$$

where $\mathbf{Q}_n = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{R}_n\mathbf{D}^{-1}$. We can now pre-multiply the above expression of $\mathbf{S}_n$ with $\boldsymbol{\Sigma}^{-1}$

$$\boldsymbol{\Sigma}^{-1}\mathbf{S}_n = \boldsymbol{\Sigma}^{-1}\mathbf{R}_n + \boldsymbol{\Sigma}^{-1}\mathbf{R}_n\mathbf{U}\left(\mathbf{I}_K + \mathbf{L}^\top\mathbf{Q}_n\mathbf{L}\right)^{-1}\mathbf{U}^\top\mathbf{R}_n$$

We define $\mathbf{P}_n = \left(\mathbf{I}_K + \mathbf{L}^\top\mathbf{Q}_n\mathbf{L}\right)^{-1}$ and expand $\boldsymbol{\Sigma}^{-1}$ using (A.3)

$$\boldsymbol{\Sigma}^{-1}\mathbf{S}_n = \mathbf{D}^{-1}\mathbf{R}_n + \mathbf{D}^{-1}\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n - \mathbf{U}\mathbf{C}\mathbf{U}^\top\mathbf{R}_n - \mathbf{U}\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n$$

Taking the trace on both sides, we get

$$\begin{aligned}\operatorname{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right) &= \operatorname{Tr}\left(\mathbf{D}^{-1}\mathbf{R}_n\right) + \operatorname{Tr}\left(\mathbf{D}^{-1}\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\right) \\ &\quad - \operatorname{Tr}\left(\mathbf{U}\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\right) - \operatorname{Tr}\left(\mathbf{U}\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\right)\end{aligned} \tag{A.9}$$

Now, the trick is to use the cyclicity of the trace to manipulate the order of matrix multiplications

$$\begin{aligned}\operatorname{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}_n\right) &= \operatorname{Tr}\left(\mathbf{D}^{-1}\mathbf{R}_n\right) + \operatorname{Tr}\left(\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\mathbf{D}^{-1}\mathbf{R}_n\mathbf{U}\right) \\ &\quad - \operatorname{Tr}\left(\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\right) - \operatorname{Tr}\left(\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\mathbf{U}\right).\end{aligned}$$

Now, we can compute $\operatorname{Tr}\left(\mathbf{D}^{-1}\mathbf{R}_n\right)$ in $O(D)$, $\mathbf{U}^\top\mathbf{R}_n\mathbf{D}^{-1}\mathbf{R}_n\mathbf{U}$ in $O(DK^2)$ and $\operatorname{Tr}\left(\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\mathbf{D}^{-1}\mathbf{R}_n\mathbf{U}\right)$ in $O(K^3)$ after computing $\mathbf{P}_n$. For the third term, we have already computed $\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}$ in (A.4) and so taking the trace takes $O(D)$. Finally, we can right multiply $\mathbf{C}\mathbf{U}^\top\mathbf{R}_n\mathbf{U}$ first with $\mathbf{P}_n$, then with $\mathbf{U}^\top\mathbf{R}_n\mathbf{U}$ in $O(K^3)$.

A.7 Computing $(\boldsymbol{\mu} - \boldsymbol{m}_n)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{m}_n)$

To compute this quadratic form, we first note the expression of $\mathbf{m}_n$-

$$\mathbf{m}_n = \mathbf{S}_n\big(\boldsymbol{\Lambda}_n\boldsymbol{\mu}_n + \boldsymbol{\Sigma}^{-1}\big(\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n\big)\big)$$

We start off by computing $\boldsymbol{\mu} + \mathbf{B}\mathbf{x}_n$ in $O(NDQ)$. Since matrix vector products are much cheaper to compute than matrix matrix products, the trick to evaluating this efficiently is to never compute $\boldsymbol{\Sigma}^{-1}$ by itself but the product $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$.

$$\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \mathbf{D}^{-1}\boldsymbol{\mu} - \mathbf{U}\mathbf{C}\mathbf{U}^\top\boldsymbol{\mu} \tag{A.10}$$

where we use (A.3) to expand $\boldsymbol{\Sigma}^{-1}$. The matrix vector product $\mathbf{D}^{-1}\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}_n\boldsymbol{\mu}_n$ takes $O(D)$. For the second term, we start by evaluating $\mathbf{U}^\top\boldsymbol{\mu}$ in $O(KD)$ which gives us a K-vector, then $\mathbf{C}(\mathbf{U}^\top\boldsymbol{\mu})$ in $O(K^2)$ and finally $\mathbf{U}(\mathbf{C}\mathbf{U}^\top\boldsymbol{\mu})$ in $O(KD)$. We can then compute $\boldsymbol{\nu}_n = \boldsymbol{\Lambda}_n\boldsymbol{\mu}_n + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ in $O(D)$.

Similarly, to compute $\boldsymbol{m}_n$ we directly find the product $\mathbf{S}_n\boldsymbol{\nu}_n$ by again expanding $\mathbf{S}_n$ with (A.8)

$$\mathbf{m}_n = \mathbf{S}_n\boldsymbol{\nu}_n = \mathbf{R}_n\boldsymbol{\nu}_n + \mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\boldsymbol{\nu}_n$$

We again use the same approach as above, and start by evaluating $\mathbf{R}_n\boldsymbol{\nu}_n$ in $O(D)$, $\mathbf{U}^\top\mathbf{R}_n\boldsymbol{\nu}_n$ in $O(KD)$, $\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\boldsymbol{\nu}_n$ in $O(K^2)$, $\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\boldsymbol{\nu}_n$ in $O(KD)$ and finally we can compute $\mathbf{R}_n\mathbf{U}\mathbf{P}_n\mathbf{U}^\top\mathbf{R}_n\boldsymbol{\nu}_n$

and hence $\mathbf{m}_n$ in $O(D)$. Now that we have $\mathbf{m}_n$, we can compute $\mathbf{\Sigma}^{-1}\boldsymbol{m}_n$ the same way as (A.10) by simply replacing $\boldsymbol{\mu}$ with $\boldsymbol{m}_n$ which will again take $O(KD)$. We can then compute $\mathbf{\Sigma}^{-1}(\boldsymbol{\mu} - \boldsymbol{m}_n)$ as well as $(\boldsymbol{\mu} - \boldsymbol{m}_n)^\top$ in $O(D)$ leading us to the desired quadratic form $(\boldsymbol{\mu} - \boldsymbol{m}_n)^T \mathbf{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{m}_n)$ again in $O(D)$.

A.8 Overall Time Complexity

The overall time complexity of computing the ELBO becomes the sum of computing each of these terms individually-

1. Computing $|\mathbf{S}_n| : \approx O(ND) + O(NKD) + O(NKD^2) + O(NK^3)$
2. Computing $(\mathbf{S}_n)_{ii} : \approx O(NK^3) + O(NDK^2)$
3. Computing $|\mathbf{\Sigma}| : \approx O(D) + O(K^3)$
4. Computing $\text{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{S}_n\right) : \approx O(ND) + O(NK^3) + O(NDK^2)$
5. Computing $(\boldsymbol{\mu} - \boldsymbol{m}_n)^T \mathbf{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{m}_n) : \approx O(NDQ) + O(ND) + O(NKD) + O(NK^2)$

which results in a time complexity of $O(NDK^2 + NDQ)$.

Appendix B

Laplace Approximations for Response Variable Distributions

Here we summarize the Laplace approximations of the likelihoods of z_n for various response variable distributions. The approximation for the Poisson is already discussed.

B.1 Bernoulli.

For the Bernoulli distribution with $\mathbb{E}[y] = \sigma(z)$ (inverse link function $\mu = g^{-1}(z) = \sigma(z)$), we can model any η and σ^2 for the two cases, $y = 0$ and $y = 1$, by simply setting

$$\eta_i(y, \boldsymbol{\phi}) = \phi_{i,0} + \phi_{i,1}\, y \tag{B.1}$$

$$\ln \lambda_i(y, \boldsymbol{\phi}) = \phi_{i,2} + \phi_{i,3}\, y\,. \tag{B.2}$$

B.2 Binomial.

For the binomial distribution with M_i draws with probability $p_i = \sigma(z_i)$, we have with $\mathbb{E}[y_i] = M_i\sigma(z)$ (inverse link function $\mu_i = g^{-1}(z_i) = M\sigma(z_i)$) and

$$\begin{aligned} f(z_i) &= \ln \mathrm{Binom}(y_i|p_i{=}e^{z_i}, M_i) \\ &= \ln \binom{M_i}{y_i} + y_i \ln p_i + (M_i - y_i)\ln(1-p_i) \\ &= y_i \ln \sigma(z_i) + (M_i - y_i)\ln\sigma(-z_i) + \text{const} \\ &= y_i z_i + M_i \ln\sigma(-z_i) + \text{const}. \end{aligned} \tag{B.3}$$

The derivative is $f'(z_i) = y_i - M_i\sigma(z_i)$. With a Binomial prior with a_i and b_i pseudocounts, the mode of the posterior is at $\sigma(z_i^*) = \frac{y_i+a_i}{M_i+a_i+b_i}$, or $z_i^* = \ln\frac{y_i+a_i}{M_i-y_i+a_i+b_i}$, and the second derivative at the mode is $-M_i\frac{y_i+a_i}{M_i+a_i+b_i}\frac{M_i-y_i+b_i}{M_i+a_i+b_i}$.

B.3 Gamma.

For the Gamma distribution with $\mathbb{E}[y] = e^z$ (inverse link function $\mu_i = g^{-1}(z_i) = e^{z_i}$) and fixed shape parameter $\psi_i = \alpha_i$ and $\beta_i = \alpha_i\,\mu_i$,

$$\begin{aligned} f(z_i) &= \ln \mathrm{Gamma}(y_i|\mu{=}e^{z_i}, \alpha_i) \\ &= \alpha_i \ln\beta_i - \beta_i y_i + (\alpha_i - 1)\ln y_i - \ln\Gamma(\alpha_i)) \\ &= \alpha_i z_i - \alpha_i y_i e^{z_i} + \text{const} \end{aligned} \tag{B.4}$$

has derivative $f'(z_i) = \alpha_i - \alpha_i y_i e^{z_i}$ and a mode at $z_i^* = \ln y_i$ and second derivative at the mode of $-\alpha_i y_i e^{z_i} = -\alpha_i$. The Laplace approximation is $f(z) \approx \ln\mathcal{N}(z|\ln y, 1/\alpha_i) + \text{const}$.

B.4 Natural exponential family.

To generalize, for a univariate exponential family distribution $p(y|z) = h(y)e^{yz-A(z)}$ with natural parameter z and normalization constant $e^{-A(z)}$, the expectation value can be obtained from

$$0 = \frac{\mathrm{d}}{\mathrm{d}z}\int p(y|z)\mathrm{d}y = \int p(y|z)\,\frac{\mathrm{d}\ln p(y|z)}{\mathrm{d}z}\mathrm{d}y = \mathbb{E}_{p(y|z)}\left[\frac{\mathrm{d}(yz - A(z))}{\mathrm{d}z}\right] = \mathbb{E}_{p(y|z)}[y] - A'(z), \tag{B.5}$$

and so the inverse link function is $g^{-1}(z) = \mathbb{E}_{p(y|z)}[y] = A'(z)$. The mode of $f(z) = \ln p(y|z) = yz - A(z) + \ln h(y)$ occurs when $f'(z) = y - A'(z) = 0$, so at $z = g(y)$, and the second derivative at the mode is $-A''(z) = -\frac{\mathrm{d}g^{-1}(z)}{\mathrm{d}z}$. The Laplace approximation is $f(z) \approx \ln \mathcal{N}(z|g(y), A''(g(y))) + \text{const}$.

B.5 Gumbel.

For the Gumbel (extreme-value) distribution with $\mathbb{E}[y] = z$ (identity link function, $\mu_i = z_i$) and fixed scale parameter β_i,

$$\begin{aligned} f(z_i) &= \ln \text{Gumbel}(y_i|\mu = z_i, \beta_i) \\ &= \ln\left(\frac{1}{\beta_i}\exp\left(-\frac{y_i - z_i}{\beta_i} - e^{-\frac{y_i - z_i}{\beta_i}}\right)\right) \\ &= -\frac{y_i - z_i}{\beta_i} - e^{-\frac{y_i - z_i}{\beta_i}} + \text{const} \end{aligned} \tag{B.6}$$

has derivative $f'(z_i) = -\frac{1}{\beta_i}\left(1 - e^{-\frac{y_i - z_i}{\beta_i}}\right)$, a mode at $z_i^* = y_i$, and second derivative at the mode of $-\frac{1}{\beta_i^2}$. The Laplace approximation is $f(z_i) \approx \ln \mathcal{N}(z_i|y_i, \beta_i^{-2}) + \text{const}$.

Appendix C

Expected Conditional Log Probability for Response Variables

We have already discussed this for the Poisson response and here we summarize the Expected conditional log probability

$$\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{u\sim\mathcal{N}(0,1)} \left[\ln p\big(y_{ni} | \mu_i^y = g^{-1}(m_{ni} + A_{nii}u), \boldsymbol{\theta}, \boldsymbol{\psi}\big) \right]$$

as discussed in Chapter 4 for various other response variable distributions.

C.1 Bernoulli.

For the case of Bernoulli distribution, the $g^{-1}(z) = \sigma(z)$ and

$$\begin{aligned}
\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) &= \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{u\sim\mathcal{N}(0,1)} \left[y_{ni}\sigma(m_{ni} + A_{nii}u) + (1 - y_{ni})(1 - \sigma(m_{ni} + A_{nii}u)) \right], \\
&= \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{u\sim\mathcal{N}(0,1)} \left[(2y_{ni} - 1)\sigma(m_{ni} + A_{nii}u) \right] + \text{const}, \\
&= \sum_{i=1}^{D} \sum_{n=1}^{N} \left[(2y_{ni} - 1)\sigma(m_{ni} + A_{nii}u) \right] + \text{const}, \\
&= \sum_{i=1}^{D} \sum_{n=1}^{N} \left[(2y_{ni} - 1)\Phi\left(\frac{\lambda m_{ni}}{\sqrt{1 + \lambda^2 \Sigma_{nii}}} \right) \right] + \text{const}, \qquad \text{(C.1)}
\end{aligned}$$

where we used

$$\mathbb{E}_{z\sim\mathcal{N}(\mu,\sigma^2)}[\sigma(z)] = \int \sigma(z)\mathcal{N}(z;\mu,\sigma^2)\mathrm{d}z \approx \int \Phi(\lambda z)\mathcal{N}(z;\mu,\sigma^2)\mathrm{d}z = \Phi\left(\frac{\lambda\mu}{\sqrt{1+\lambda^2\sigma^2}}\right)$$

using the probit approximation for the sigmoid $\sigma(a) \approx \Phi(\lambda a)$ with $\lambda^2 = \pi/8$.

C.2 Binomial.

For the binomial distribution with M_{ni} draws with probability $p_{ni} = \sigma(z_{ni})$, we have the inverse link function $m_{ni} = g^{-1}(z_{ni}) = M_{ni}\sigma(z_{ni})$.

$$\begin{aligned}\rho(\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\psi}) &= \sum_{i=1}^{D}\sum_{n=1}^{N}\mathbb{E}_{u\sim\mathcal{N}(0,1)}\left[y_{ni}(m_{ni}+A_{nii}u) + M_{ni}\ln\sigma(-(m_{ni}+A_{nii}u))\right] \\ &= \sum_{i=1}^{D}\sum_{n=1}^{N}\left[y_{ni}m_{ni} - M_{ni}\frac{1}{S}\sum_{s=1}^{S}\ln\sigma(-(m_{ni}+A_{nii}u^s))\right] \end{aligned} \tag{C.2}$$

where we use Monte-Carlo to calculate the expectation of the last term by sampling $u^s \sim \mathcal{N}(0,1)$.

C.3 Gamma.

For the Gamma distribution with $\mathbb{E}[y] = e^z$, hence the inverse link function is $\mu_i = g^{-1}(z_i) = e^{z_i}$ and fixed shape parameter $\psi_i = \alpha_i$ and $\beta_i = \alpha_i\,\mu_i$,

$$\begin{aligned}\rho(\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\psi}) &= \sum_{i=1}^{D}\sum_{n=1}^{N}\mathbb{E}_{u\sim\mathcal{N}(0,1)}\left[\alpha_{ni}(m_{ni}+A_{nii}u) - \alpha_{ni}y_{ni}e^{m_{ni}+A_{nii}u} + \text{const}\right] \\ &= \sum_{i=1}^{D}\sum_{n=1}^{N}\left[\alpha_{ni}m_{ni} - \alpha_{ni}y_{ni}e^{m_{ni}+\Sigma_{nii}/2}\right] + \text{const.}\end{aligned} \tag{C.3}$$

C.4 Natural Exponential Family.

For a univariate exponential family distribution $p(y|z) = h(y)e^{yz-D(z)}$ with natural parameter z and normalization constant $e^{-D(z)}$, the inverse link function can be found using (B.5), hence

$g^{-1}(z) = \mathbb{E}_{p(y|z)}[y] = D'(z)$.

$$\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{i=1}^{D} \sum_{n=1}^{N} \mathbb{E}_{u \sim \mathcal{N}(0,1)} \Big[y_{ni} D'(m_{ni} + A_{nii} u) - D(D'(m_{ni} + A_{nii} u)) \Big] + \text{const.} \tag{C.4}$$

C.5 Gumbel.

For the Gumbel (extreme-value) distribution with $\mathbb{E}[y] = z$ (identity link function, $\mu_i = z_i$) and fixed scale parameter β_i,

$$\rho(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{i=1}^{D} \sum_{n=1}^{N} \left[- \frac{y_{ni} - (m_{ni} + A_{nii} u)}{\beta_{ni}} - \exp\left(- \frac{y_{ni} - (m_{ni} + A_{nii} u)}{\beta_{ni}} \right) \right] + \text{const.} \tag{C.5}$$

Appendix D

Entropy and the Cross-Entropy of the Multivariate Gaussian

D.1 Entropy of the Multivariate Gaussian

Suppose that we have a random variable X following a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$. We can compute the entropy as follows:

$$\mathcal{H}(X) = -\mathbb{E}\left[\ln\left(\frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \cdot \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]\right)\right], \tag{D.1}$$

$$= -\mathbb{E}\left[-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right], \tag{D.2}$$

$$= \frac{D}{2}\ln(2\pi) + \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \frac{1}{2}\mathbb{E}\left[(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]. \tag{D.3}$$

The last term can be evaluated as

$$\mathbb{E}\left[(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] = \mathbb{E}\left[\mathrm{Tr}\left((\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)\right], \tag{D.4}$$

$$= \mathbb{E}\left[\mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\right)\right], \tag{D.5}$$

$$= \mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbb{E}\left[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\right]\right), \tag{D.6}$$

$$= \mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\right), \tag{D.7}$$

$$= \mathrm{Tr}\left(\boldsymbol{I}_d\right), \tag{D.8}$$

$$= D. \tag{D.9}$$

such that the differential entropy of a multivariate Gaussian is

$$\mathcal{H}(X) = \frac{D}{2}\ln(2\pi) + \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \frac{D}{2}. \tag{D.10}$$

D.2 Cross Entropy between Multivariate Gaussians

Suppose that we want to compute the cross entropy of $\mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ with respect to the Gaussian $\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, where $\boldsymbol{\mu}_p, \boldsymbol{\mu}_q \in \mathbb{R}^D$ and $\boldsymbol{\Sigma}_p, \boldsymbol{\Sigma}_q \in \mathbb{R}^{D\times D}$. To compute this, we will make use of the relations between the cross-entropy and the KL divergence

$$\mathcal{H}(q, p) = D_{KL}(q||p) + \mathcal{H}(q) \tag{D.11}$$

where $D_{KL}(q||p)$ is the KL-divergence between q and p. Hence, we can compute the entropy of two Gaussians as:

$$\mathcal{H}\Big(\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q), \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)\Big) = D_{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q), \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)\Big) + \mathcal{H}\Big(\mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)\Big). \tag{D.12}$$

We computed the entropy of a multivariate Gaussian above. For the KL-divergence between two multivariate Gaussians we use the following:

$$D_{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q), \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)\Big) = \mathbb{E}_q\Bigg[\frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_p|}{|\boldsymbol{\Sigma}_q|} - (\boldsymbol{x}-\boldsymbol{\mu}_q)^{\mathrm{T}}\boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_q) \tag{D.13}$$

$$+ (\boldsymbol{x}-\boldsymbol{\mu}_p)^{\mathrm{T}}\boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p)\Bigg]. \tag{D.14}$$

Now each of the expectations in the second and the third term can be evaluated using trace trick in the proof in (D.9). This gives us the expression of KL-divergence as

$$D_{KL}(q\|p) = \frac{1}{2}\left[\ln\frac{|\boldsymbol{\Sigma}_p|}{|\boldsymbol{\Sigma}_q|} - D + \big(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\big)^T\boldsymbol{\Sigma}_p^{-1}\big(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\big) + \mathrm{Tr}\big\{\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Sigma}_q\big\}\right]. \tag{D.15}$$

Hence final expression of cross entropy is

$$\mathcal{H}\Big(\mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q), \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)\Big) = \frac{1}{2}\Bigg[\ln\frac{|\boldsymbol{\Sigma}_p|}{|\boldsymbol{\Sigma}_q|} + \big(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\big)^T\boldsymbol{\Sigma}_p^{-1}\big(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\big)$$

$$+ \mathrm{Tr}\big\{\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Sigma}_q\big\} + \ln|\boldsymbol{\Sigma}_q| + D\big(\ln(2\pi) - 1\big)\Bigg]. \tag{D.16}$$

Appendix E

Derivation of ELBO for one-dimensional case

For the one dimensional case where $D = 1$, with the true model parameters as $\boldsymbol{\theta} = \big(\mu, \sigma\big)$, we can write the model as

$$\begin{aligned} z_n &\sim \mathcal{N}(\mu, \sigma^2) \\ y_n | z_n &\sim \text{Pois}\big(e^{z_n}\big). \end{aligned}$$

We calculate the posterior of the latent variables using the Bayes rule

$$\begin{aligned} p(z_n | y_n, \mu, \sigma) &\propto p(y_n | z_n)\, p(z_n | \mu, \sigma) \\ &= \frac{e^{-e^{z_n}} e^{y_n\, z_n}}{\ln y_n!} \times \mathcal{N}(z_n | \mu, \sigma^2). \end{aligned}$$

We approximate the first likelihood term using a normal likelihood where the mean $\eta(y_n, \phi_0, \phi_1)$ and precision $\lambda(y_n, \phi_2, \phi_3)$ are governed with some appropriate variational parameters $\boldsymbol{\phi} = (\phi_0, \phi_1, \phi_2, \phi_3)$

$$\begin{aligned} q_n(z_n | y_n, \mu, \sigma, \eta(y_n, \boldsymbol{\phi}), \lambda(y_n, \boldsymbol{\phi})) &= \mathcal{N}(z_n | \eta(y_n, \boldsymbol{\phi}), \lambda^{-1}(y_n, \boldsymbol{\phi})) \times \mathcal{N}(z_n | \mu, \sigma^2), \\ &= \mathcal{N}\bigg(z_n \bigg| \frac{\sigma^2 \lambda(y_n, \boldsymbol{\phi}) \eta(y_n, \boldsymbol{\phi}) + \mu}{1 + \lambda(y_n, \boldsymbol{\phi}) \sigma^2}, \frac{\sigma^2}{1 + \sigma^2 \lambda(y_n, \boldsymbol{\phi})}\bigg), \end{aligned} \tag{E.1}$$

where we use the formula

$$\mathcal{N}(\mu_1, \sigma_1^2) \times \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}\bigg(\frac{\sigma_1^2 \mu_2 + \sigma_2^2 \mu_1}{\sigma_1^2 + \sigma_2^2}, \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}\bigg).$$

We denote

$$\mu_{q_n}(y_n, \mu, \sigma, \boldsymbol{\phi}) = \frac{\sigma^2 \lambda(y_n, \boldsymbol{\phi}) \eta(y_n, \boldsymbol{\phi}) + \mu}{1 + \lambda(y_n, \boldsymbol{\phi}) \sigma^2},$$

and

$$\sigma^2_{q_n}(y_n, \mu, \sigma, \boldsymbol{\phi}) = \frac{\sigma^2}{1 + \sigma^2 \lambda(y_n, \boldsymbol{\phi})}.$$

The evidence lower bound can be written as:

$$\mathrm{ELBO}(\mu, \sigma, \boldsymbol{\phi}) = \sum_{n=1}^{N} \mathbb{E}_{q_n(z_n|y_n,\mu,\sigma,\boldsymbol{\phi})} \left[\ln \frac{p_n(y_n, z_n|\mu, \sigma)}{q_n(z_n|\mu_{q_n}, \sigma_{q_n})} \right]. \tag{E.2}$$

To simplify this, we simplify the expression of the conditional density in the numerator:

$$\begin{aligned} \ln p_n(y_n, z_n|\mu, \sigma) &= \ln \mathrm{Pois}(y_n|e^{z_n}) + \ln \mathcal{N}(z_n|\mu, \sigma), \\ &= y_n \, z_n - e^{z_n} - \ln \sigma - \frac{1}{2\sigma^2}(z_n - \mu)^2 + \mathrm{const.} \end{aligned} \tag{E.3}$$

Finally we explicitly write the denominator of E.2 using the fact that it is a Gaussian density:

$$\ln q_n(z_n|\mu_{q_n}, \sigma_{q_n}) = -\ln \sigma_{q_n} - \frac{1}{2\sigma^2_{q_n}}(z_n - \mu_{q_n})^2 + \mathrm{const} \tag{E.4}$$

We now use the reparametrization trick to sample from the variational distribution by drawing samples $\epsilon \sim \mathcal{N}(0, 1)$ and using

$$z_n(\epsilon) = \mu_{q_n} + \sigma_{q_n} \, \epsilon.$$

We note the following facts:

$$\mathbb{E}(z_n) = \mu_{q_n},$$

$$\mathbb{E}(z_n^2) = \mu_{q_n}^2 + \sigma_{q_n}^2,$$

$$\mathbb{E}((z_n - a)^2) = \mu_{q_n}^2 + \sigma_{q_n}^2 - 2a\mu_{q_n} + a^2,$$

$$\mathbb{E}(e^{z_n}) = e^{\mu_{q_n} + \sigma^2_{q_n}/2},$$

where a is a constant. Hence, we can write our ELBO as

$$
\begin{aligned}
&\text{ELBO}(\mu, \sigma, \boldsymbol{\phi}) \\
&\quad= \sum_{n=1}^{N} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)} \left[y_n \, z_n - e^{z_n} - \ln\sigma - \frac{1}{2\sigma^2}(z_n - \mu)^2 + \ln\sigma_{q_n} + \frac{1}{2\sigma_{q_n}^2}(z_n - \mu_{q_n})^2 \right], \\
&\quad= \sum_{n=1}^{N} \left[y_n \mu_{q_n} - e^{\mu_{q_n} + \sigma_{q_n}^2/2} - \ln\sigma - \frac{1}{2\sigma^2}(\mu_{q_n}^2 + \sigma_{q_n}^2 - 2\mu\mu_{q_n} + \mu^2) + \ln\sigma_{q_n} + \frac{1}{2} \right], \\
&\quad= \sum_{n=1}^{N} \left[y_n \mu_{q_n} - e^{\mu_{q_n} + \sigma_{q_n}^2/2} - \ln\sigma - \frac{1}{2\sigma^2}(\mu_{q_n}^2 + \sigma_{q_n}^2 - 2\mu\mu_{q_n} + \mu^2) + \ln\sigma_{q_n} \right], \qquad \text{(E.5)}
\end{aligned}
$$

where we omit the constant since it does not depend on $(\mu, \sigma, \boldsymbol{\phi})$. This is expression of the ELBO objective that we maximize for the one dimensional case.

Bibliography

[1] Yafang Zhang, Fanping Kong, Elizabeth J. Crofton, Steven N. Dragosljvich, M. P. Sinha, Dingge Li, Xiuzhen Fan, Shyny Koshy, Jonathan D. Hommel, Heidi M. Spratt, Bruce Arthur Luxon, and Thomas A. Green. Transcriptomics of environmental enrichment reveals a role for retinoic acid signaling in addiction. *Frontiers in Molecular Neuroscience*, 9, 2016.

[2] Sarah Aldridge and Sarah A Teichmann. Single cell transcriptomics comes of age. *Nature Commun*, 11(1):4307, 2020.

[3] Peter D Congdon. *Applied Bayesian hierarchical methods*. CRC Press, 2010.

[4] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2004.

[5] Diane Lambert. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14, 1992.

[6] Yen-Yi Chen, Tsung-I Chen, Chia-Chou Wu, and Chun-Chi Liu. Zero-inflated negative binomial mixed regression modeling of over-dispersed count data with extra zeros in transcriptomics data. *Scientific reports*, 6(1):1–10, 2016.

[7] Daryl Pregibon. Review: P. McCullagh, J. A. Nelder, Generalized Linear Models. *The Annals of Statistics*, 12(4):1589 – 1596, 1984.

[8] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome biology*, 11(10):R106, 2010.

[9] Martin J Lindstrom and Douglas M Bates. *Moments, likelihood, and approximations in mixed models*. CRC press, 2011.

[10] Xiaowei Wang and Jian Kang. Zero-inflated negative binomial mixed models for rna-seq data analysis. *Computational and mathematical methods in medicine*, 2020, 2020.

[11] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference, 2013.

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

[13] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[14] Mohadeseh Alsadat Farzammehr, Mohsen Mohammadzadeh, Mohammad Zadkarami, and G. Mclachlan. Bayesian analysis of generalized linear mixed models with spatial correlated and unrestricted skew normal errors. *Communications in Statistics - Theory and Methods*, 51:1–22, 04 2021.

[15] Martyn Plummer. Jags: A program for analysis of bayesian graphical models using gibbs sampling. *Distributed Statistical Computing*, 2003.

[16] Han Lin Shang. The bugs book: a practical introduction to bayesian analysis. *Journal of Applied Statistics*, 40, 07 2013.

[17] Christopher Nemeth and Paul Fearnhead. Stochastic gradient markov chain monte carlo, 2019.

[18] Robert H. Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986.

[19] W. D. Vousden, W. M. Farr, and I. Mandel. Dynamic temperature selection for parallel tempering in markov chain monte carlo simulations. *Monthly Notices of the Royal Astronomical Society*, 455(2):1919–1937, nov 2015.

[20] Radford M. Neal. *MCMC Using Hamiltonian Dynamics*, chapter chapter5. CRC Press, 2011.

[21] Matias Quiroz, Mattias Villani, Robert Kohn, Minh-Ngoc Tran, and Khue-Dung Dang. Subsampling mcmc - an introduction for the survey statistician, 2018.

[22] Ralitza Gueorguieva. A multivariate generalized linear mixed model for joint modelling of clustered outcomes in the exponential family. *Stat Modelling*, 1(3):177–193, 2001.

[23] Alan Agresti, James G Booth, James P Hobert, and Brian Caffo. Random-effects modeling of categorical response data. *Sociol Methodology*, 30(1):27–80, 2000.

[24] N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25, 1993.

[25] Stephen W. Raudenbush, Meng-Li Yang, and Matheos Yosef. Maximum likelihood for generalized linear models with nested random effects via high-order, multivariate laplace approximation. *Journal of Computational and Graphical Statistics*, 9(1):141–157, 2000.

[26] J. T. Ormerod and M. P. Wand. Gaussian variational approximate inference for generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 21(1):2–17, 2012.

[27] Julien Chiquet, Mahendra Mariadassou, and Stéphane Robin. Variational inference for probabilistic poisson PCA. *Ann Appl Stat*, 12(4):2674–2698, 2018.

[28] Julien Chiquet, Stephane Robin, and Mahendra Mariadassou. Variational inference for sparse network reconstruction from count data. In *Int Conf Mach Learning*, pages 1162–1171. PMLR, 2019.

[29] Julien Chiquet, Mahendra Mariadassou, and Stéphane Robin. The Poisson-lognormal model as a versatile framework for the joint analysis of species abundances. *Front Ecol Evol*, 9:188, 2021.

[30] Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of Statistical Physics*, 52:479–487, 07 1988.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[32] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(3):503–528, December 1989.

[33] Maria Fossheim, Einar Nilssen, and Michaela Aschan. Fish assemblages in the barents seapublished in collaboration with the university of bergen and the institute of marine research, norway, and the marine biological laboratory, university of copenhagen, denmark. *Marine Biology Research - MAR BIOL RES*, 2:260–269, 08 2006.

[34] Philippe Usseglio-Polatera and Yves Auda. Influence des facteurs météorologiques sur les résultats de piégeage lumineux. *http://dx.doi.org/10.1051/limn/1987005*, 23, 01 1987.

[35] Boris Jakuschkin, Virgil Fievet, Loïc Schwaller, Thomas Fort, Cécile Robin, and Corinne Vacher. Deciphering the pathobiome: Intra- and interkingdom interactions involving the pathogen erysiphe alphitoides. *Microbial ecology*, 72(4):870—880, November 2016.

[36] Jerome Richard. Einsum is slow for tensor multiplication. Stack Overflow, Aug 2022.

[37] Stan Development Team. *Stan: A C++ Library for Probability and Sampling*. https://mc-stan.org/, 2012.

[38] David B. Dunson. *Nonparametric Bayes*, pages 1–5. John Wiley and Sons Ltd, 2017.

[39] Robert Cornish, Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Scalable metropolis-hastings for exact bayesian inference with large datasets, 2019.