Toward fast and scalable Bayesian Machine Learning

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Marius Hobbhahn, M. Sc.
aus Nürnberg

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation: 13.01.2025

Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Philipp Hennig
2. Berichterstatter:	Prof. Dr. Jakob Macke

Disclaimer: this thesis uses Federico Marotta's `kaobook` template based on Ken Arroyo Ohori's doctoral thesis. I am grateful to Felix Dangel for making his template public.

Acknowledgments

I want to express my deepest gratitude to Philipp Hennig for his mentorship before, throughout and after the PhD. When Philipp gave his first lecture on Probabilistic Machine Learning in 2018, I saw a professor who was incredibly competent and deeply cared about his students. I can wholeheartedly say that my first impression was accurate and I'm extremely grateful for everything Philipp has done for me, motivating me when I was hopeless and being supportive of my interests and ideas.

I want to thank Franziska Weiler (Methods of Machine Learning (MoML) group), Leila Masri & Sara Sorce (International Max Planck Research School for Intelligent Systems) for their effective administrative support.

I would like to thank all members of the Methods of Machine Learning (MoML) group. I continue to be impressed by how competent and friendly everyone at MoML is. I want to especially thank Agustinus for the patience and mentorship during my first paper and Nathanael, Hans, Jonathan, Marvin, Lukas, Emilia, Runa and others for great conversations.

I want to thank all of my friends for supporting me through the PhD journey. Knowing that others have encountered similar struggles and how they solved them made it easier for me.

I'd like to thank my parents for supporting me at all times.

Finally, I want to thank Maria Heitmeier for her support throughout this journey. Thank you for being a reasonable voice and cheering me up when I was hopeless after running into unforeseen problems.

Thank you!

Marius Hobbhahn
Tübingen, July 15, 2024

Abstract

Machine Learning (ML) involves training a model on a dataset and subsequently using that model to make predictions on previously unseen data. While predictive performance is a crucial aspect of ML, we also want ML methods to be able to quantify uncertainty and incorporate prior knowledge.

Bayesian Machine Learning approaches address these desiderata by explicitly or implicitly leveraging Bayes' theorem. Bayesian methods generate a posterior distribution over the model parameters, enabling principled uncertainty quantification. Furthermore, Bayesian ML allows for incorporating expert knowledge and additional information by specifying prior distributions.

The family of Bayesian ML techniques includes a wide range of methods, e.g. Monte Carlo sampling and Variational Inference. While Bayesian approaches offer advantages, they can be computationally expensive. *The primary objective of this thesis is to develop fast and efficient Bayesian ML methods* that maintain the benefits of Bayesian ML while mitigating the computational burden.

We achieve fast inference by combining Laplace approximations, change of variables, exponential families, and automatic differentiation. Laplace approximations can be analytically computed or efficiently approximated, while exponential family properties and change of variables allow for analytical transformations between distributions, avoiding costly iterative schemes.

First, we introduce Laplace Matching, which takes a non-Gaussian exponential family distribution, applies a transformation of variable such that its support is matched with a Gaussian, and then applies a Laplace approximation to the transformed variable. The result is a *closed-form* mapping between the parameters of the exponential family and a Gaussian. The variable transform is chosen such that one sufficient statistic of the exponential family matches the sufficient statistic of a Gaussian, yielding a much better approximation than on the original basis. Laplace Matching can be used to unlock the benefits of Gaussian inference for non-Gaussian exponential families while paying a small fixed approximation error. This allows, for instance, the straightforward application of Gaussian processes to various data formats. We demonstrate its feasibility by modeling the German election landscape and the trajectory of currency covariance matrices over time.

Second, we introduce the Laplace Bridge for Bayesian Deep Neural Networks, which is an application of Laplace Matching. Bayesian Neural Networks typically have a Gaussian distribution over the logits, then sample from this distribution and transform these samples using the softmax. The Laplace Bridge allows to analytically transform the Gaussian distribution into a Dirichlet over the outputs. The properties of the Dirichlet can be used to enable new applications. For example, marginals of Dirichlets are Dirichlets (or Betas in the one-dimensional case). We use this fact to create "uncertainty-aware top-k", a technique that uses the marginal Beta distributions of the model's predictions to individually determine how many classes it is uncertain between. We demonstrate this approach on the 1000 classes of ImageNet.

Third, we develop PIHAM, a generative model explicitly designed to perform Probabilistic Inference in directed and undirected Heterogeneous and Attributed Multilayer networks. PIHAM extends Laplace Matching from single variables to combinations of variables, including multiplication and addition. It transforms all latent variables to be Gaussian, then uses automatic differentiation to get a Laplace Approximation, and then uses Laplace Matching to transform them into their intended basis. We show its feasibility in the concrete use case of network inference, where we analyze the social support network of a rural Indian village. The flexibility of PIHAM allows us to incorporate various types of information, including categorical data and interactions between individuals.

In this thesis, we present the initial steps towards a broader paradigm of Bayesian approximate inference methods that are, first and foremost, fast. We outline extensions to perform fast Bayesian inference on neural networks and arbitrary probabilistic networks.

Zusammenfassung

Maschinelles Lernen (ML) beinhaltet das Training eines Modells auf einem Datensatz und die anschließende Verwendung dieses Modells, um Vorhersagen für bisher unbekannte Daten zu treffen. Während die Vorhersageakkuratheit ein entscheidender Aspekt von ML ist, wollen wir auch, dass ML-Methoden in der Lage sind, Unsicherheiten zu quantifizieren und Vorwissen zu integrieren.

Ansätze des Bayesianischen Maschinellen Lernens adressieren diese Desiderata, indem sie explizit oder implizit das Theorem von Bayes nutzen. Bayesianische Methoden erzeugen eine posteriore Verteilung über die Modellparameter, was eine prinzipielle Quantifizierung der Unsicherheit ermöglicht. Darüber hinaus erlaubt Bayesianisches ML die Einbeziehung von Expertenwissen und zusätzlichen Informationen durch die Spezifikation von A-priori-Verteilungen.

Die Familie der Bayesianischen ML-Techniken umfasst eine Vielzahl von Methoden, z.B. Monte-Carlo-Sampling und Variational Inference. Während Bayesianische Ansätze Vorteile bieten, können sie rechenintensiv sein. *Das primäre Ziel dieser Arbeit ist die Entwicklung schneller und effizienter Bayesianischer ML-Methoden*, die die Vorteile des Bayesianischen ML beibehalten und gleichzeitig den Rechenaufwand reduzieren.

Wir erreichen eine schnelle Inferenz durch die Kombination von Laplace-Approximationen, Variablentransformation, Exponentialfamilien und automatischer Differentiation. Laplace-Approximationen können analytisch berechnet oder effizient approximiert werden, während die Eigenschaften der Exponentialfamilie und der Variablentransformation analytische Transformationen zwischen Verteilungen ermöglichen und so aufwändige iterative Schemata vermeiden.

Zunächst führen wir das Laplace Matching ein, das eine nicht-Gaußsche Exponentialfamilienverteilung nimmt, eine Variablentransformation anwendet, so dass der Träger mit einer Gaußverteilung übereinstimmt, und dann eine Laplace-Approximation auf die transformierte Variable anwendet. Das Ergebnis ist eine *analytische* Abbildung zwischen den Parametern der Exponentialfamilie und einer Gaußverteilung. Die Variablentransformation wird so gewählt, dass die kanonische Statistik der Exponentialfamilie mit der kanonische Statistik der Gaußverteilung übereinstimmt, was zu einer wesentlich besseren Approximation als auf der ursprünglichen Basis führt. Laplace Matching kann verwendet werden, um die Vorteile der Gaußschen Inferenz für nicht-Gaußsche Exponentialfamilien zu nutzen. Dies ermöglicht beispielsweise die unkomplizierte Anwendung von Gauß-Prozessen auf verschiedene Datenformate. Wir demonstrieren das, indem wir die deutsche Wahllandschaft und die Entwicklung von Währungskovarianzmatrizen im Laufe der Zeit modellieren.

Zweitens führen wir die Laplace-Bridge for Bayesian Neuronal Networks ein, die eine Anwendung des Laplace Matching ist. Bayesianische Neuronale Netze haben typischerweise eine Gaußsche Verteilung über die Logits, dann werden aus dieser Verteilung Stichproben gezogen und diese Stichproben mit der Softmax-Funktion transformiert. Die Laplace-Bridge ermöglicht es, die Gaußsche Verteilung analytisch in eine Dirichlet-Verteilung über die Ausgabewerte zu transformieren. Die Eigenschaften der Dirichlet-Verteilung können genutzt werden, um neue Anwendungen zu ermöglichen. Zum Beispiel sind Marginalverteilungen von Dirichlet-Verteilungen wieder Dirichlet-Verteilungen (oder Beta-Verteilungen im eindimensionalen Fall). Wir nutzen diese Tatsache, um "unsicherheitsbewusste Top-k" zu erstellen, eine Technik, die die marginalen Beta-Verteilungen der Modellvorhersagen verwendet, um individuell zu bestimmen, zwischen wie vielen Klassen das Modell unsicher ist. Wir demonstrieren diesen Ansatz auf den 1000 Klassen von ImageNet.

Drittens entwickeln wir PIHAM, ein generatives Modell, das explizit für die probabilistische Inferenz in gerichteten und ungerichteten heterogenen und attributierten Multilayer-Netzwerken entwickelt wurde. PIHAM erweitert das Laplace Matching von einzelnen Variablen auf Kombinationen von Variablen, einschließlich Multiplikation und Addition. Es transformiert alle latenten Variablen in Gaußsche Variablen, verwendet dann automatische Differentiation, um eine Laplace-Approximation zu erhalten, und verwendet dann Laplace Matching, um sie in ihre beabsichtigte Basis zu transformieren.

Wir zeigen die Durchführbarkeit am konkreten Anwendungsfall der Netzwerkinferenz, bei der wir das soziale Unterstützungsnetzwerk eines ländlichen indischen Dorfes analysieren. Die Flexibilität von PIHAM ermöglicht es uns, verschiedene Arten von Informationen, einschließlich kategorischer Daten und Interaktionen zwischen Individuen, zu berücksichtigen.

In dieser Arbeit stellen wir die ersten Schritte zu einem breiteren Paradigma von Bayesianischen approximativen Inferenzmethoden vor, die in erster Linie schnell sind. Wir skizzieren Erweiterungen, um schnelle Bayesianische Inferenz auf neuronalen Netzen und beliebigen probabilistischen Netzen durchzuführen.

Table of Contents

1. Overview

1.1 Introduction

Machine Learning (ML) has supported human decision-making for many decades. Although methods and techniques have significantly advanced, the fundamental principle of learning from data and generalizing to unseen data points remains unchanged. However, many ML methods face challenges in providing well-calibrated uncertainty estimates for their predictions and incorporating prior knowledge. While Bayesian methods tackle these issues, they often suffer from computational inefficiency. In this thesis, we aim to develop fast approximations for Bayesian ML methods to address these limitations.

The origins of Machine Learning can be traced back to Linear Regression, which was introduced over two centuries ago by Legendre (Stigler, 1981). Building upon this foundation, researchers have developed increasingly sophisticated, adaptable, and potent learning algorithms.

Remark 1.1 (Origin of regression) There are some disputes over who invented Linear Regression. Legendre published the method in 1805 but Gauss claimed in 1809 that he had been using Linear Regression since 1795 and thus should be seen as the true inventor. For more details see Stigler (1981).

The family of regression techniques was expanded to accommodate various data and function types, such as logistic and polynomial regression. Additionally, kernel-based methods, including Support Vector Machines (Cortes et al., 1995), Kernel PCA (Schölkopf et al., 1997), and others (see e.g. Hofmann et al., 2008), were developed to further enhance the capabilities of machine learning algorithms.

Neural networks have emerged as the dominant Machine Learning (ML) technique over the past decade. Although early neural networks were introduced in 1986 (Rumelhart et al., 1986), it took the wider community more than twenty years to recognize and harness their potential. The introduction of AlexNet (Krizhevsky, Sutskever, et al., 2012) in 2012 marked a turning point in ML, setting a new state of the art for image classification on ImageNet (Russakovsky et al., 2014). Since then, the amount of compute, data, and model size have grown exponentially (Sevilla et al., 2022; Villalobos, Sevilla, Besiroglu, et al., 2022; Villalobos, Sevilla, Heim, et al., 2022). Deep Learning has surpassed human performance in various games, including Go, Starcraft, and most Atari games (Mnih et al., 2013; Vinyals et al., 2019; Silver et al., 2017). Beyond its success in gaming, Deep Learning has also begun to impact the lives of average citizens. Large language models are used for translation, as spellcheckers, and in search engines like BingGPT or general-purpose conversation tools such as ChatGPT (Brown et al., 2020) and Claude (Anthropic, 2024).

Despite the increasing diversity of ML tools, the significant advancements in their capabilities, and the broadening of their applications, the fundamental concept remains consistent with linear regression. Namely, the developers define a model, train it on data, and then use it to make predictions on unseen data. Specifically, in most ML methods, the ML model will get an input and predict a point estimate of the output. For example, an ML model might be trained on historical house prices. Then,

a user can enter the data for a new house, and the model will predict a price for it.

However, many ML methods struggle to address two specific desiderata:

1. **Quantified uncertainty:** Point estimates are often sufficient as initial guesses, but in some cases, it is crucial for the model to quantify the uncertainty associated with its predictions. For instance, when estimating house prices, a plausible range of estimates may be more informative than a single value. Well-calibrated uncertainty estimates are particularly vital in safety-critical applications, such as medical diagnosis or autonomous vehicles, to enable better risk assessment and decision-making.
2. **Incorporating prior knowledge:** ML models typically learn solely from the training data and generalize patterns to unseen datapoints. However, human experts may possess domain-specific knowledge or informed hypotheses that are not captured in the training data but should be incorporated into the model's predictions. For example, a potential home buyer who has personally visited a property may wish to integrate their preferences into the price estimation. Most ML methods find it challenging to incorporate such prior knowledge effectively.

Bayesian Machine Learning aims to incorporate both of these desiderata.

Remark 1.2 (What does Bayesian mean?) There are theoretical and practical disagreements regarding the precise definition of Bayesian ML. For instance, there is debate about whether Deep Ensembles can be considered a Bayesian method (A. G. Wilson and Izmailov, 2021). Moreover, methods previously seen as "non-Bayesian" often have Bayesian interpretations. Ridge regression, for example, can be viewed as Bayesian linear regression with independent Gaussian priors of equal variance on the regression parameters β_i. Similarly, Korbak et al. (2022) demonstrate that RL with KL penalties, a common approach for fine-tuning language models, can be interpreted as Bayesian inference. Different communities have also developed varying understandings of Bayesian ML. Some consider it an umbrella term for all ML methods that quantify uncertainty, while others restrict it to methods that explicitly use Bayes' theorem in their computation. This thesis adopts a pragmatic stance on the "true" meaning of Bayesian ML, focusing on the goal of accelerating ML approaches that quantify uncertainty and allow for the incorporation of prior knowledge, regardless of the exact definition of Bayesian.

1.1.1 Bayesian ML

Bayesian Machine Learning leverages Bayes' theorem, either implicitly or explicitly, to generate a posterior distribution over the parameters and/or outputs:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.1}$$

$$P(B) = \int P(B|A)P(A)dA \tag{1.2}$$

In theory, the Bayesian formalism enables the incorporation of additional knowledge through the prior distribution and yields a distribution over the parameters, which can be used to obtain a distribution over outputs, in contrast to point estimates.

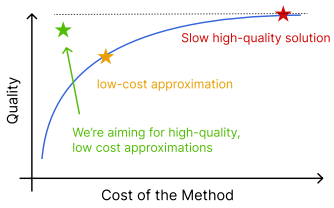
In practice, Bayesian ML often faces challenges due to the intractability of the evidence term $P(B)$, which is the integral in the denominator. As a result, approximations to the posterior are commonly employed. For instance, Laplace Approximations (D. J. C. MacKay, 1992) approximate the distribution of interest as a Gaussian, while Variational Inference methods introduce a parameterized approximate distribution and minimize its KL divergence to the true posterior (Blei et al., 2017a). Additionally, Markov Chain Monte Carlo methods use samples from the posterior as an approximation (Gilks et al., 1995).

However, these approximations often remain computationally expensive, leading practitioners to be either unable or reluctant to employ them.

Consequently, a conventional narrative has emerged, suggesting that "Bayesian methods are slower than non-Bayesian methods."

The primary objective of this thesis is to identify and develop approximate Bayesian Inference methods that prioritize speed, even if they compromise approximation quality.

1.1.2 Bayesian methods are not inherently slower than non-Bayesian approaches



The goal of the thesis is to find fast high-quality approximate Bayesian Inference methods.

Intuitively, it makes sense that Bayesian methods should be more computationally expensive than non-Bayesian methods. After all, they do provide a distribution instead of a point estimate. However, in practice, this comes with a myriad of clarifications, exceptions, and caveats. In the following, we will discuss three examples to highlight the complexity of this discussion.

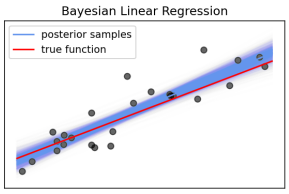
Example 1: Linear Regression

In linear regression, the goal is to obtain a parameter vector β for the equation

$$Y = \beta X + b \tag{1.3}$$

where the bias term b is commonly incorporated into β by appending a row of constant 1-terms to the data matrix X. To minimize the mean squared error between the true labels Y and the predicted labels $\hat{Y}$, the parameters β are computed using

$$\beta = \left(X^\top X\right)^{-1} X^\top Y \quad . \tag{1.4}$$



Visualization of Bayesian Regression.

The data matrix X consists of N datapoints, each with D features. Consequently, $(X^\top X)$ is a $D \times D$ matrix. Since D is typically much smaller than N, computing β is generally feasible despite the $\mathcal{O}(N^3)$ complexity of the matrix inversion.

In Bayesian linear regression, the goal is to find a distribution over the parameters β (including the bias term) that maximizes the posterior probability for the model:

$$Y = \beta X + \epsilon \tag{1.5}$$

where ϵ is an i.i.d. noise term. Unlike standard linear regression, Bayesian linear regression does not generally have a closed-form solution. Instead, it is typically computed using Markov chain Monte Carlo (MCMC) sampling methods. Although modern sampling techniques like No U-turn sampling (NUTS) (Hoffman et al., 2011) are more efficient than the first MCMC methods such as Metropolis-Hastings (Chib et al., 1995), they are still significantly slower than the closed-form solution of standard linear regression. This would suggest that the Bayesian approach to linear regression is computationally more expensive than the non-Bayesian counterpart.

However, even this case is not straightforward. A common choice is to use a normal distribution for β conditioned on σ^2, and an inverse-gamma distribution for σ^2. The conjugacy of these distributions allows

the posterior to be computed in closed form (Gelman et al., 2013).

$$\beta|\sigma^2 \sim \mathcal{N}(\mu_0, \sigma^2 V_0) \tag{1.6}$$
$$\sigma^2 \sim \text{InvGamma}(a, b) \tag{1.7}$$
$$\beta|y \sim \mathcal{N}(\mu_n, \sigma^2 V_n) \tag{1.8}$$
$$\sigma^2|y \sim \text{InvGamma}(a_n, b_n) \tag{1.9}$$

where the updated parameters are given by:

$$V_n = (V_0^{-1} + X^T X)^{-1} \tag{1.10}$$
$$\mu_n = V_n(V_0^{-1}\mu_0 + X^T y) \tag{1.11}$$
$$a_n = a + \frac{n}{2} \tag{1.12}$$
$$b_n = b + \frac{1}{2}[(y - X\mu_0)^T(I + XV_0X^T)^{-1}(y - X\mu_0)] \tag{1.13}$$

Consequently, in the conjugate case, the computational costs of the Bayesian and non-Bayesian approaches are nearly equivalent. Conjugate inference techniques can be applied to many ML techniques beyond linear regression. They pose a prominent counter-example to the narrative that Bayesian methods are always much slower than non-Bayesian methods.

Example 2: Bayesian Neural Networks

Obtaining high-fidelity approximations to the posterior distribution of neural network parameters can be computationally expensive. Izmailov et al. (2021) demonstrate this by using full-batch Hamiltonian Monte Carlo (HMC) to sample from the posterior of a ResNet (He et al., 2016) architecture trained on CIFAR10 (Krizhevsky, Nair, et al., 2014). While conventional methods like SGD and mini-batch training can typically train the parameters of a standard neural network for this task within a few hours on most laptops, the HMC approximation to the full posterior required the authors to use 512 TPUv3 in parallel, which far exceeds a typical academic compute budget.

Therefore, it is fair to say that for some choices of approximation techniques, the Bayesian solution is much more computationally expensive than the non-Bayesian alternative.

However, there are cheap approximations to the posterior of neural network weights that do not meaningfully increase the training cost. For instance, several methods based on Laplace approximations (e.g. Kristiadi et al., 2020; Daxberger et al., 2021) only necessitate one additional training epoch to approximate the uncertainty. Posterior networks (Charpentier et al., 2020) are even more computationally efficient and provide a better-calibrated estimate of the posterior.

In such cases, the computational cost of the Bayesian method is not substantially higher than that of the non-Bayesian alternative.

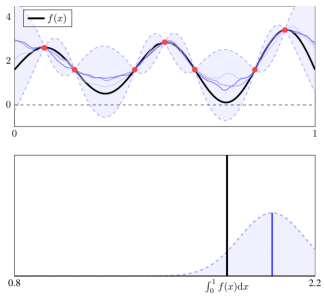
Example 3: Bayesian Quadrature

Lastly, we will examine a case in which the Bayesian approach is faster or more efficient than the corresponding non-Bayesian technique.

A prominent example of such a case is the computation of integrals. Bayesian Quadrature (Diaconis, 1988; O'Hagan, 1991; Ghahramani et al., 2002; Briol et al., 2017; Hennig, Osborne, et al., 2022) is a probabilistic numerical method to approximate intractable integration problems. Commonly, Bayesian Quadrature uses a Gaussian Process as a prior distribution to enable conjugate inference. While traditional quadrature methods, such as Newton-Cotes or Gaussian quadrature are highly accurate for low-dimensional integration problems, their performance degrades for high-dimensional tasks due to the curse of dimensionality.

Since Bayesian Quadrature treats integration as a Bayesian inference problem, it can incorporate prior knowledge about the function (e.g. smoothness) and can use the quantified uncertainty to focus evaluations on regions with the highest uncertainty.

Thus, Bayesian Quadrature scales better to high-dimensional problems than traditional Quadrature and is thus *more effective* (see e.g. Gunter et al., 2014; Kanagawa et al., 2019). This reverses the conventional narrative and thus poses a clear counter-example to it.



Visualization of Bayesian Quadrature. Taken from (Tskarvon, 2024).

1.1.3 Motivation

We established that the narrative that "Bayesian methods are inherently slower than non-Bayesian methods" is not true in all cases. Therefore, it is possible to build algorithms that provide quantified uncertainty and allow for the incorporation of prior knowledge that are not prohibitively expensive to run, at least for certain use cases. The goal of this thesis is to expand the class of these algorithms and build fast Bayesian ML algorithms for a wide range of use cases.

There are three concrete motivations for developing fast Bayesian approximate inference schemes, even if they may occasionally yield large approximation errors:

1. **Enabling new applications:** Fast Bayesian techniques would enable the application of Bayesian inference to large-scale problems that were previously intractable. This could significantly impact fields operating with big data, such as molecular biology or material sciences, where uncertainty quantification and prior knowledge integration are crucial.
2. **Providing default baselines:** Fast Bayesian inference schemes could serve as go-to baselines for practitioners. Researchers could initially use the fast approach and switch to more computationally intensive but accurate methods if necessary. This would offer a quick and efficient way to begin Bayesian analysis and determine the need for more advanced techniques.
3. **Informing priors for accurate methods:** The posteriors obtained from fast Bayesian inference approaches could be used as informative priors for slower, higher-fidelity methods. By leveraging the parameters gained from the fast approach, this two-stage approach could significantly reduce the computational cost of the more accurate methods.

1.2 Other work

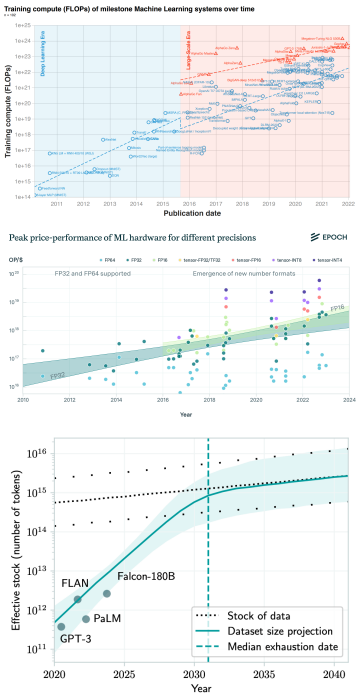
This thesis is focused on three papers that share a common narrative centered on efficient Bayesian approximate inference. Beyond these papers, I have worked on multiple other research projects unrelated to Bayesian ML that will be briefly presented in this section.

1.2.1 AI forecasting

The broader field of AI forecasting aims to understand and predict AI-related trends. Our work at EpochAI specifically focused on quantifying trends that enable the prediction of the most powerful models' capabilities. To this end, we collected data and estimated scaling laws for three* key components of frontier model training: *compute*, *data*, and *parameters*.

In Sevilla et al. (2022), we analyze the historical compute usage of the largest ML training runs and estimate that the compute requirements of the biggest models grow by a factor of 6 annually. We also investigate the potential limitations posed by the availability of training data in Villalobos, Sevilla, Heim, et al. (2022), assessing whether it could become a realistic bottleneck for training frontier models. Moreover, in Villalobos, Sevilla, Besiroglu, et al. (2022), we discover that since 2018, the number of parameters in frontier models has been increasing at a rate of approximately 9 times per year.

In light of the crucial role that computational resources play in training state-of-the-art models, we investigate the question of whether Moore's law is likely to persist and for what duration in Hobbhahn and Besiroglu (2022). Furthermore, in Hobbhahn, Heim, et al. (2023), we conduct a comprehensive analysis of various compute-related trends, encompassing price-performance ratios, precision, memory, interconnect technologies, and specialized hardware components.



Key figures from my AI forecasting work. Typically, years are on the x-axis and a log-transformed AI-related quantity on the y-axis.

1.2.2 AI safety

The field of AI safety aims to guarantee that artificial intelligence systems remain aligned with human values, even as they become more capable than humans in a large range of tasks.

In Hobbhahn, Lieberum, et al. (2022), we assess the causal understanding of state-of-the-art large language models. A model without a solid grasp of causality may fail to generalize to out-of-distribution scenarios, potentially compromising its safety. The concept of "aligning AIs to human values" is frequently used without clearly defining these "human values." In Hobbhahn, Landgrebe, et al. (2022), we survey around 1,000 people to gather their views on morality and AI, aiming to identify possible alignment targets.

Moreover, in Scheurer et al. (2023), we red-team cutting-edge language models and show that they can deceive users without being explicitly instructed to do so. This work can be attributed to the wider field of AI

Remark 1.3 (Current work) You can find most of my current work here: Apollo Research

* The three main factors for ML progress are compute, data, and algorithmic progress. Parameters are coupled to compute and data through scaling laws. Nevertheless, trends in parameter scaling are still worth investigating.

evaluations that try to establish how capable AI systems are and assess their safety.

1.3 Outline

Chapter 2 presents the key statistical techniques that form the foundation of our approaches: Laplace Approximations, Change of Variable, Exponential Families, and Automatic Differentiation.

In Chapter 3, we present Laplace Matching (LM), an approximate inference scheme that establishes a mapping between the parameters of different exponential families. The central concept of LM is to transform the variable of any exponential family into a domain where it closely resembles a Gaussian distribution. Subsequently, a Laplace approximation is applied, and the resulting analytic mapping is used to efficiently translate between exponential families defined on different domains.

In Chapter 4, we explore the Laplace Bridge for Bayesian Neural Networks, an important application of Laplace Matching. We demonstrate that the Laplace Bridge, which is the specific instance of Laplace Matching for the Dirichlet distribution, can be used to substitute the sampling process at the end of Gaussian-based neural network approximations.

In Chapter 5, we introduce PIHAM, a generative model explicitly designed to perform Probabilistic Inference in directed and undirected Heterogeneous and Attributed Multilayer networks. PIHAM extends Laplace Matching from single variables to combinations of variables, including multiplication and addition. It transforms all latent variables to be approximately Gaussian and then uses automatic differentiation to efficiently get a Laplace Approximation. Finally, it then uses Laplace Matching to transform the resulting Gaussians into their intended basis.

In Chapter 6, we analyze the advantages and limitations of each approach and outline potential avenues for future research.

Every main chapter includes a section titled “Practical Advice & Honest Thoughts”, offering guidance for practitioners and my honest opinion of the strengths and drawbacks of each method.

Part I.

Background

2. Background

This section covers the core concepts and techniques to understand the methods presented in the subsequent chapters. First, we introduce Laplace approximations (Section 2.1), a powerful tool for approximating distributions in Bayesian inference. Next, we discuss the change of variable technique for probability density functions (Section 2.2), which plays a crucial role in transforming distributions and facilitating efficient computation. We then explore exponential families (Section 2.3), a broad class of probability distributions that exhibit desirable properties and are widely used in various statistical models. Finally, we introduce automatic differentiation (Section 2.4), a technique that enables the efficient computation of derivatives, which is instrumental in implementing gradient-based optimization algorithms and Laplace approximations.

2.1 Laplace Approximations

The Laplace approximation fits a normal distribution to a given function, in our case a probability density function (pdf). If $\hat{\theta}$ denotes the mode of a pdf $h(\theta)$, then it is also the mode of the log-pdf $q(\theta) = \log h(\theta)$ since the logarithm is a monotonic transformation. The 2nd-order Taylor expansion of $q(\theta)$ is

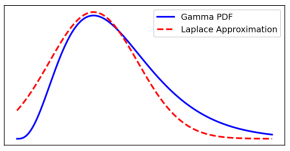
$$q(\theta) \approx q(\hat{\theta}) + \nabla q(\hat{\theta})(\theta - \hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})\nabla\nabla^\top q(\hat{\theta})(\theta - \hat{\theta}) \tag{2.1}$$

$$= q(\hat{\theta}) + 0 + \frac{1}{2}(\theta - \hat{\theta})\nabla\nabla^\top q(\hat{\theta})(\theta - \hat{\theta}) \qquad [\text{since } \nabla q(\theta) = 0] \tag{2.2}$$

$$= c - \frac{1}{2}(\theta - \mu)^\top \Sigma^{-1}(\theta - \mu)\,, \tag{2.3}$$

where c is a constant, $\mu = \hat{\theta}$ and $\Sigma = \{-\nabla\nabla^\top q(\hat{\theta})\}^{-1}$. Since the Gaussian is fit at the mode, the gradient term is zero. The right-hand side of the last line matches the log-pdf of a Gaussian. Therefore, the pdf $h(\theta)$ is approximated by the pdf of the normal distribution $\mathcal{N}(\mu, \Sigma)$ where $\mu = \hat{\theta}$ and $\Sigma = \{-\nabla\nabla^\top q(\hat{\theta})\}^{-1}$.

Since the Laplace approximation is based on the 2nd-order Taylor expansion, it introduces a fixed approximation error. On the other hand, since it is computed analytically, the Laplace approximation yields a closed-form solution and doesn't require any iterative approximation, making it very fast. Thus, a core goal of this thesis is to transform the distribution that is being approximated such that the Laplace approximation is a much better fit to leverage its computational efficiency.



Visualization of a Laplace Approximation. The Gamma distribution is approximated by a Gaussian.

2.2 Change of variable for pdfs

Let X have a continuous density f_X.* Let $g : \mathbb{R} \to \mathbb{R}$ be piecewise strictly monotonic and continuously differentiable, i.e. there exist intervals $I_1, I_2, \ldots, I_n$ that partition $\mathbb{R}$ such that g is strictly monotonic and continuously differentiable on the interior of each I_i. For each interval i, $g : I_i \to \mathbb{R}$ is invertible on $g(I_i)$; let g_i^{-1} be its inverse function. Let $Y = g(X)$ and $r(y) = \{y \mid y = g(x), x \in \mathbb{R}\}$ be the range of g. Then, the density function f_Y of Y exists and is given by

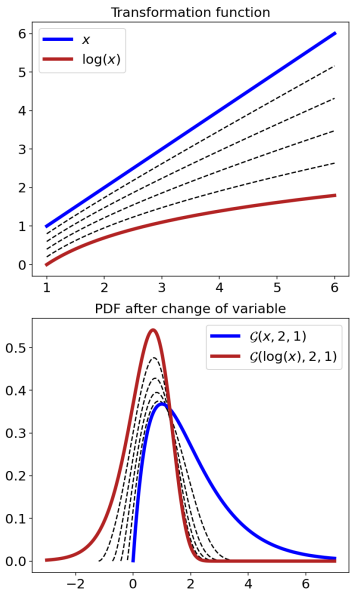
$$f_Y(y) = \sum_{i=1}^{n} f_X(g_i^{-1}(y)) \left| \frac{\partial g_i^{-1}(y)}{\partial y} \right| \mathbf{1}_{r(y)} \,. \tag{2.4}$$

This is also true for the multi-dimensional case

$$f_Y(\mathbf{y}) = \sum_{i=1}^{n} f_{\mathbf{x}}\left(g^{-1}(\mathbf{y})\right) \left| \det \left[\frac{dg^{-1}(\mathbf{x})}{d\mathbf{x}} \Big|_{\mathbf{x}=\mathbf{y}} \right] \right| \mathbf{1}_{r(y)} \,. \tag{2.5}$$

where the function in each interval is multiplied with the determinant of the Jacobian of g^{-1} evaluated at $\mathbf{y}$.

Laplace approximations (see Section 2.1) can be applied to probability distributions represented in different bases, resulting in varying levels of approximation quality. To minimize the approximation error introduced by such an approximation, our objective is to transform the variable of non-Gaussian distributions in a way that most closely resembles a Gaussian distribution.



Visualization of a change of variable. The Gamma distribution is being log-transformed with a linear interpolation between x and $\log(x)$. For more illustrations see (Hobbhahn, 2021a).

* This definition follows "Probability Essentials" by Jacod et al. (2004)

2.3 Exponential Families

A pdf that can be written in the form

$$p(x) = h(x) \cdot \exp\left(w^\top \phi(x) - \log Z(w)\right) \tag{2.6}$$

where

$$Z(w) := \int_{\mathbb{X}} h(x) \exp\left(w^\top \phi(x)\right)\, dx \tag{2.7}$$

is called an exponential family. $\phi(x) : \mathbb{X} \to \mathbb{R}^d$ are the sufficient statistics, $w \in \mathcal{D} \subseteq \mathbb{R}^d$ the natural parameters with domain $\mathcal{D}$, $\log Z(w) : \mathbb{R}^d \to \mathbb{R}$ is the (log) partition function (normalization constant), and $h(x) : \mathbb{X} \to \mathbb{R}_+$ the base measure.

The product of two pdfs of different instances of the same exponential family is proportional to another instance of this exponential family. Assume we have two instances of the same exponential family

$$p(x; w_i) = h(x) \cdot \exp(\phi(x)^\top w_i - \log Z(w_i)) \tag{2.8}$$

where $i \in \{1, 2\}$ and $w_i \in \mathcal{D}$. Then, the product of their pdfs is

$$p(x; w_1) \cdot p(x; w_2) \propto p(x; w_1 + w_2) \tag{2.9}$$

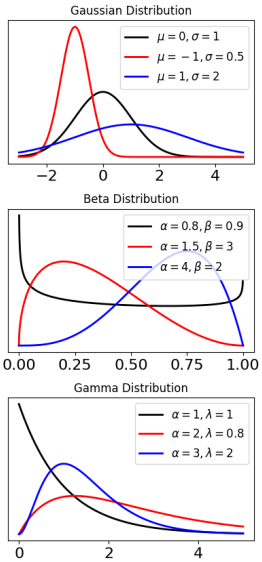
if $w_1 + w_2 \in \mathcal{D}$. Thus, the product of two pdfs can be computed by adding their parameters.

Another important property of exponential families is that their expected value is given by the differential of the normalizing constant

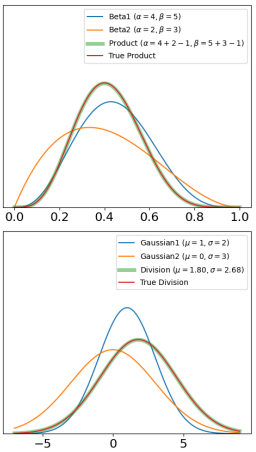
$$\mathbb{E}_{p_w}(\phi(x)) = \nabla_w \log Z(w)\,. \tag{2.10}$$

This makes the computation of moments easier since it replaces integration with differentiation.

The thesis aims to leverage the speed of Laplace approximations, which yield Gaussian distributions, for distributions with limited support, such as those confined to the positive real numbers or the probability simplex. To achieve this, we apply a change of variable (see Section 2.2) to transform these distributions to a new basis where their support aligns with that of a Gaussian, prior to performing the Laplace approximation.



Visualization of three exponential family distributions.



Products of two pdfs of an exponential families are proportional to an exponential family with the sum of their parameters. For more illustrations see (Hobbhahn, 2021b).

2.4 Automatic Differentiation

Automatic differentiation (AD) is a technique used to compute derivatives of functions efficiently and automatically. AD makes use of the chain rule of partial derivatives of composite functions, i.e. it exploits the fact that partial derivatives of composite functions, no matter how complicated, are the product of local partial derivatives. For example, consider the composition of functions

$$y = h(g(f(x))) = h(g(f(z_0))) = h(g(z_1)) = h(z_2) \quad . \tag{2.11}$$

Then the partial derivative of y w.r.t x is

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial z_0} = \frac{\partial h(z_2)}{\partial z_2}\frac{\partial g(z_1)}{\partial z_1}\frac{\partial f(x)}{\partial x} \quad . \tag{2.12}$$

The atomic partial derivatives, e.g. $\frac{\partial f(x)}{\partial x}$, are known for most simple functions. Therefore, if a computational graph consists of compositions of functions with known partial derivatives, the partial derivative of the entire graph can be computed automatically. This is fundamental to modern ML since neural networks and many other ML methods are compositions of simple functions with known derivatives. This allows us to compute arbitrary gradients and use gradient-based optimization such as SGD.

There are many important theoretical considerations going into automatic differentiation such as whether to use forward or backward accumulation. We refer interested readers to Griewank et al. (2008) for details. For the purpose of this thesis, these choices have been made for us by the creators of modern software packages. Therefore, for the rest of the thesis, we can think of AD as a black-box tool that takes a composition of functions and returns arbitrary gradients along the graph.

Importantly, AD can also compute higher-order partial derivatives. For example, if the function $f(x)$ is twice differentiable, AD can compute the Hessian matrix

$$H_{(i,j)} = \frac{\partial^2 f}{\partial x_i \partial x_j} \quad . \tag{2.13}$$

When possible, we prefer to compute analytical approximations for our distributions due to their efficiency. However, this approach is not always feasible, particularly when dealing with compositions and combinations of functions, such as addition and multiplication. In such cases, we resort to Automatic Differentiation, which remains highly efficient in practice. This will become evident in Chapter 5, where we approximate models that involve such combinations of variables.

Part II.

Toward Fast and Scalable Bayesian Machine Learning

3. Laplace Matching

Remark 3.1 The contents of this chapter are primarily based on:

Marius Hobbhahn, and Philipp Hennig. "*Laplace Matching for fast Approximate Inference in Latent Gaussian Models*". arXiv preprint arXiv:2105.03109. 2022.

	Idea	Analysis	Exp.	Code	Writing
Marius Hobbhahn	80%	90%	90%	100%	90%
Philipp Hennig	20%	10%	10%	0%	10%

Abstract

Bayesian inference on non-Gaussian data is often non-analytic and requires computationally expensive approximations such as sampling or variational inference. We propose an approximate inference framework primarily designed to be computationally cheap while still achieving high approximation quality. The concept, which we call *Laplace Matching*, involves closed-form, approximate, bi-directional transformations between the parameter spaces of exponential families. These are constructed from Laplace approximations under custom-designed basis transformations. The mappings can then be leveraged to effectively turn a latent Gaussian distribution into an approximate conjugate prior to a rich class of observable variables. This allows us to train latent Gaussian models such as Gaussian Processes on non-Gaussian data at nearly no additional cost. The method can be thought of as a pre-processing step which can be implemented in <5 lines of code and runs in less than a second. Furthermore, Laplace Matching yields a simple way to group similar data points together, e.g. to produce inducing points for GPs. We empirically evaluate the method with experiments for four different exponential distributions, namely the Beta, Gamma, Dirichlet and inverse Wishart, showing approximation quality comparable to state-of-the-art approximate inference techniques at a drastic reduction in computational cost.

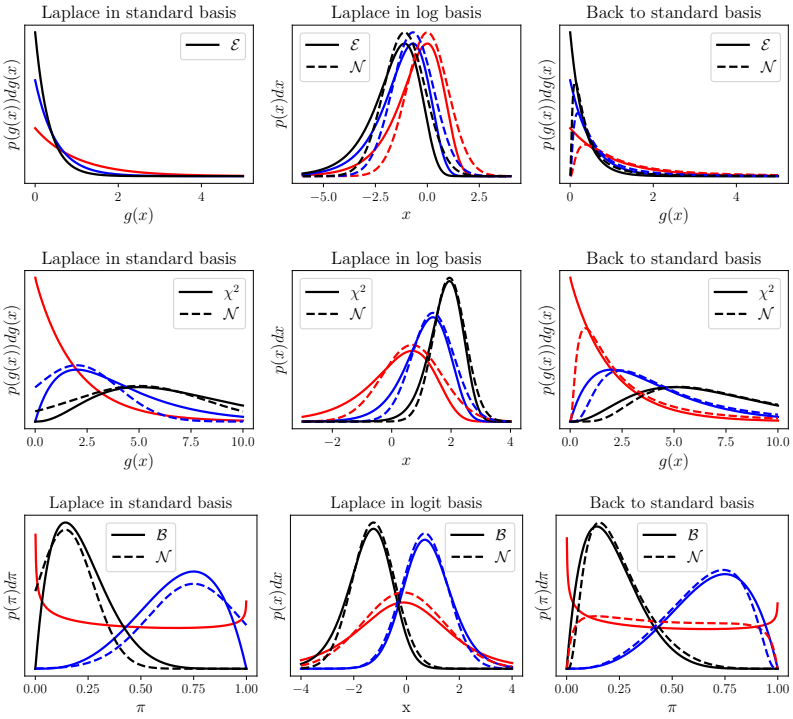


Figure 3.1: Laplace approximations for the exponential distribution (top), the Chi-squared distribution (middle row), and the Beta distribution (bottom) in the standard basis (left) and a more suitable basis (middle column) for three different sets of parameters (——, —— and ——). The Laplace approximations are then transformed back to the standard base for comparison (right). The parameter for the red line does not have a valid Laplace approximation in the standard base, since the Hessian is not positive definite. In the new basis, however, this problem does not occur. Similar figures for all other exponential families can be found in the appendix.

3.1 Introduction

Probabilistic inference often involves latent and observable variables of different types, lying in different domains. Apart from the most basic cases, one cannot expect to find a joint conjugate prior for all latent quantities. For instance, when observing discrete samples from a latent categorical distribution, the Dirichlet exponential family provides a conjugate prior, enabling analytic and computationally efficient Bayesian inference. However, if we observe several such discrete samples from several latent categoricals that must be assumed to relate to each other somehow, the Dirichlet is the wrong model. A common approach in such scenarios is to construct a latent Gaussian model with a latent multi-output Gaussian process prior, which is connected to the individual categorical distributions via a softmax link function. However, the softmax of a Gaussian random variable has no useful analytic form, and inference is intractable. Approximate inference methods constructed from Laplace approximations (cf. §3 in Rasmussen, 2006) or even custom-built Markov Chain Monte Carlo (MCMC) methods (Murray et al., 2009) are available, but they have significantly higher computational cost than conjugate prior inference.

We introduce *Laplace Matching* (LM), a principled framework for constructing a fast, closed-form, bi-directional approximate transformation between the parameters of any exponential family distribution and a Gaussian. While Laplace approximations are well-studied, the key novel aspect of our approach lies in first transforming the variable of the original distribution to enable a better approximation by a Laplace approximation in the new base (see Figure 3.1 for a visual illustration). Our work extends and generalizes an insight by D. J. MacKay (1998). LM enables efficient Gaussian inference on likelihoods with exponential family conjugate

priors at minimal additional computational cost. Although this paper primarily focuses on Gaussian Process (GP) inference, LM has broader applicability and can be used in various settings to seamlessly translate between exponential families and Gaussians.

Our contributions include

1. setting up the general framework for Laplace Matching (LM),
2. providing derivations for most commonly used exponential families,
3. showcasing how LM leads to a natural and simple way to choose inducing points in GP inference,
4. showing feasibility by applying LM to GPs on multiple non-Gaussian, commonly used data types (binary, counts, categorical and covariances) with a few lines of code, and
5. comparing LM to multiple alternatives w.r.t. approximation quality and speed.

3.2 Laplace Matching

Algorithm 1 Laplace Matching

Require: Exponential family pdf $p(x|\theta)$ with parameters θ, transformation $g(x)$

$p_y(y;\theta) = p_x(g^{-1}(y);\theta)$ ▷ Apply transformation of variable to pdf

$\mu = \arg\max_y p_y(y;\theta) \Leftrightarrow \frac{\partial p_y(y;\theta)}{\partial \theta} = 0$ ▷ Compute mode

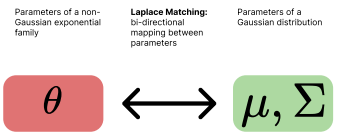
$\Sigma = -\left(\frac{\partial^2 p_y(y;\mu)}{\partial^2 y}\right)^{-1}$ ▷ Compute Hessian and invert

results in $f : \theta \to (\mu, \Sigma)$

$f^{-1} : (\mu, \Sigma) \to \theta$ ▷ Invert mapping

return $f : \theta \leftrightarrow (\mu, \Sigma)$ ▷ Return mapping between parameters

The goal of Laplace Matching is to fix the approximation of the likelihood as a Laplace approximation and then to reduce the KL-divergence between the true distribution and the Gaussian approximation by representing the likelihood in a new basis. Most other approximation schemes (e.g. Murray et al., 2009; Nickisch, 2012; Minka, 2001; Neal, 1993; Seeger et al., 2009; Williams et al., 1998; A. Wilson et al., 2010) are based on an internal step of numerical optimisation. They define a loss for the approximation quality, then define an approximation scheme and iteratively adapt the approximation to improve the quality. Each iteration has some cost. In contrast, Laplace Matching takes a different approach. It starts by proposing an approximation with a fixed cost, such as the Laplace approximation, and then transforms the representation of the data to minimize the approximation loss. This transformation is performed in closed form, eliminating the need for iterative updates. As a result, the approximation error is fixed, and the cost of finding a good approximation is externalized because it is chosen beforehand. The primary advantage of Laplace Matching lies in its computational efficiency. While other methods invest computational resources to find a particularly good approximation, Laplace Matching adopts a more simplistic approach by proposing a specific approximation with known, very low computational cost.



High-level sketch of Laplace Matching

While we thus specifically do not aim to numerically minimize some loss, we instead *motivate* the choice of a particular analytic approximation by the KL-divergence between an exponential family distribution and its Gaussian approximation, under a variable transformation. This works as follows. Let $p(x)$ be an exponential family distribution in the 'standard basis' x:

$$p_x(x) = h(x) \exp\left(w^T \phi(x) - \log Z(w)\right) . \tag{3.1}$$

Then, the random variable x can be changed to another basis with transformation $g(x)$ via Equation 2.4 (see Section 2.2). Let $x(y) = g^{-1}(x)$ be the inverse transform of $g(x)$. The resulting distribution $p_y(x(y))$ in basis y is again an exponential family

$$p_y(x(y)) = h(x(y)) \exp\left(w^T \phi(x(y)) - \log Z(w)\right) \cdot \left|\frac{\partial x(y)}{\partial y}\right| \tag{3.2}$$

$$= \exp\left(w^T \phi(x(y)) + \log(h(x(y))) + \log\left|\frac{\partial x(y)}{\partial y}\right| - \log Z(w)\right) . \tag{3.3}$$

In the new basis y a Laplace approximation is performed to yield a Gaussian $q(y) = \mathcal{N}(y; \mu, \Sigma)$, which can be written as

$$q(y) = \exp\left(\underbrace{(\Sigma_g^{-1}\mu_g)^\top y - \frac{1}{2}\operatorname{Tr}\left(\Sigma_g^{-1} y y^\top\right)}_{w_q^T \phi_q(y)} \underbrace{- \frac{k}{2}\log(2\pi)}_{\log h_q(y)} + \underbrace{\frac{1}{2}\mu_g^\top \Sigma_g^{-1}\mu_g - \frac{1}{2}\log|\Sigma_g|}_{-\log Z_q(w_q)} \right), \tag{3.4}$$

where the mean and covariance

$$\mu_g = \arg\max_y p(x(y)) \Leftrightarrow \frac{\partial p_y(x(y))}{\partial y} = 0 \tag{3.5}$$

$$\Sigma_g = -\left\{\frac{\partial^2 p(x(y); \mu)}{\partial^2 y}\right\}^{-1} \tag{3.6}$$

depend on the choice of parameters w of $p(x(y))$, which in turn depends on $g(x)$. We then define Laplace Matching as the procedure of finding a transformation $g^{-1}(x) = x(y)$ with accompanying basis y for distribution $p(x)$ such that $\mathrm{KL}\left(p(x(y))||q(y)\right)$ is small. Thus, LM describes a map from the parameters θ of $p(x)$ to the parameters (μ, Σ) of the Gaussian approximation $q(y)$. The aim is to invert this map to also have a function from (μ, Σ) to θ. However, since not all variable transformations are bijective (e.g. the softmax), the inverse map has to be approximated in some cases. Therefore, Laplace Matching ultimately yields a bi-directional, closed-form mapping between the parameters θ and (μ, Σ). One way to define an *optimal* transformation g would be that which minimizes

$\mathrm{KL}\left(p(x(y))||q(y)\right)$.

$$\mathrm{KL}(p_y(x(y))||q(y)) = \int p(y)\log\left(\frac{p(y)}{q(y)}\right)dy \tag{3.7}$$

$$= \int p(y)\log(p(y))dy - \int p(y)\log(q(y))dy \tag{3.8}$$

$$= \int p(y)\left(\log h_p(y) + w_p^\top \phi_p(y) - \log Z(w_p)\right)dy \tag{3.9}$$

$$- \int p(y)\left(\log h_q(y) + w_q^\top \phi_q(y) - \log Z(w_q)\right)dy$$

$$= \mathbb{E}_p\left[\log h_p(y)\right] + w_p^\top \mathbb{E}_p\left[\phi_p(y)\right] - \log Z(w_p) \tag{3.10}$$

$$- \mathbb{E}_p\left[\log h_q(y)\right] - w_q^T \mathbb{E}_p\left[\phi_q(y)\right] + \log Z(w_q)$$

The objective $\mathrm{KL}\left(p(x(y))||q(y)\right)$ is in general not differentiable w.r.t. $g(x)$, due to the arg max-function in the mode (see Equation 3.5). So even numerical optimization is challenging, let alone finding a global analytic extremum. Thus, other strategies have to be used to find an optimal transformation $g(x)$.

It is generally known (see, e.g. (cf. §10.7 in Bishop, 2006)) that the KL divergence between a distribution $p(y)$ and its Gaussian approximation $q(y)$ is minimized when the first two *moments match*. This comes from optimizing $\mathrm{KL}\left(p_y(x(y))||q(y)\right)$ w.r.t. w_q. Writing Equation 3.10 as a function of w_q yields

$$\text{const.} - w_q^T \mathbb{E}_p\left[\phi_q(y)\right] + \log Z(w_q) \tag{3.11}$$

which, if differentiated w.r.t w_q, yields

$$-\mathbb{E}_p\left[\phi_q(y)\right] + \nabla_{w_q}\log Z(w_q) \stackrel{!}{=} 0 \tag{3.12}$$

$$\Leftrightarrow \mathbb{E}_p\left[\phi_q(y)\right] = \nabla_{w_q}\log Z(w_q) \tag{3.13}$$

$$\Leftrightarrow \mathbb{E}_p\left[\phi_q(y)\right] = \mathbb{E}_q\left[\phi_q(y)\right] \tag{3.14}$$

due to the properties of exponential families described in Section 2.3. Consequently, the optimal choice for the transformation $g(x)$ is one that matches both sufficient statistics of the Gaussian, namely $[x, xx^\top]^\top$. However, it is not possible to simultaneously match both moments with a single transformation g, as this would imply that our distribution is already Gaussian. Therefore, we propose two alternative approaches: The first approach is to choose g such that the mode μ of $p_y(x(y))$ matches the mean, i.e., $\bar{\mu} = \arg\max_w \log p(x;w) \stackrel{!}{=} g^{-1}\left(\mathbb{E}_p(x;w)\right)$. The second approach is to choose g such that the second moment matches, i.e. $\mathbb{E}(xx^\top) = -\left[\nabla\nabla^\top \log p(x;\bar{\mu})\right]^{-1}$.

Note that while neither of these two approaches necessarily provides a minimum of the KL divergence, both are analytically tractable! Without a clear formal motivation to prefer one option over the other, Section 3.5.6 provides an empirical comparison.

Furthermore, as a second constraint for the transformation used by Laplace Matching, the *domain* of $p_y(x(y))$ should be $\mathbb{R}^d$ since this is the domain of the d-dimensional Gaussian $q(y)$, which is given by the Laplace approximation. If this is not the case, the support of $p_y(x(y))$ is smaller than the support of $q(y)$ and therefore $\mathrm{KL}\left(p_y(x(y))||q(y)\right)$ is

Table 3.1: Overview of the transformations and the sufficient statistics and domains of the exponential families in the standard and new basis. Bold notation for the new sufficient statistics denotes which moment has been matched by the transformation. Beta and Dirichlet only have one transformation each, as they have only one kind of sufficient statistic, namely $\log(x_i)$. $\mathbb{R}_{++}^{d\times d}$ and $\mathbb{R}_{S}^{d\times d}$ describe the spaces of positive definite and symmetric matrices respectively.

Distribution	$g(x)$	$g^{-1}(x)$	standard $\phi(x)$	new $\phi(y)$	Standard domain	New domain
Exponential	log	exp	x	$(\mathbf{y}, \exp(y))$	$\mathbb{R}_+$	$\mathbb{R}$
	sqrt	sqr	x	$(\log(y), \mathbf{y}^2)$	$\mathbb{R}_+$	$\mathbb{R}$
Gamma	log	exp	$(\log(x), x)$	$(\mathbf{y}, \exp(y))$	$\mathbb{R}_+$	$\mathbb{R}$
	sqrt	sqr	$(\log(x), x)$	$(\log(y), \mathbf{y}^2)$	$\mathbb{R}_+$	$\mathbb{R}$
Inv. Gamma	log	exp	$(\log(x), x)$	$(\mathbf{y}, \exp(y))$	$\mathbb{R}_+$	$\mathbb{R}$
	sqrt	sqr	$(\log(x), x)$	$(\log(y), \mathbf{y}^2)$	$\mathbb{R}_+$	$\mathbb{R}$
Chi-squared	log	exp	$(\log(x), x)$	$(\mathbf{y}, \exp(y))$	$\mathbb{R}_+$	$\mathbb{R}$
	sqrt	sqr	$(\log(x), x)$	$(\log(y), \mathbf{y}^2)$	$\mathbb{R}_+$	$\mathbb{R}$
Beta	logit	logistic	$(\log(x), \log(1-x))$	$(\log(\sigma(\mathbf{y})), (1-\log(\sigma(y)))$	$\mathbb{P}$	$\mathbb{R}$
Dirichlet	-	softmax	$(\log(x_i))$	$\log(\pi_i(\mathbf{y}))$	$\mathbb{P}^d$	$\mathbb{R}^d$
Wishart	logm	expm	$(\text{logm}(X), X)$	$(\mathbf{Y}, \text{expm}(Y))$	$\mathbb{R}_{++}^{d\times d}$	$\mathbb{R}_{S}^{d\times d}$
	sqrtm	sqrm	$(\text{logm}(X), X)$	$(\text{logm}(Y), \mathbf{Y}^2)$	$\mathbb{R}_{++}^{d\times d}$	$\mathbb{R}_{S}^{d\times d}$
Inv. Wishart	logm	expm	$(\text{logm}(X), X)$	$(\mathbf{Y}, \text{expm}(Y))$	$\mathbb{R}_{++}^{d\times d}$	$\mathbb{R}_{S}^{d\times d}$
	sqrtm	sqrm	$(\text{logm}(X), X)$	$(\text{logm}(Y), \mathbf{Y}^2)$	$\mathbb{R}_{++}^{d\times d}$	$\mathbb{R}_{S}^{d\times d}$

undefined. Matching the first or second moment provides the correct domain, which is an additional reason for moment matching.

Laplace Matching (LM), as presented in this thesis, is applicable to a wide range of latent Gaussian models, including Gaussian Process (GP) regression, Kalman filters, stochastic differential equations, and the last layer of Neural Networks (see Chapter 4). In essence, Laplace Matching allows any operation that can be applied to Gaussian random variables to be similarly applied to random variables with exponential family conjugate priors. While this transformation introduces a fixed approximation loss, it enables the application of Gaussian-based techniques to a broader class of distributions. This makes Laplace Matching a powerful tool for simplifying and unifying the treatment of various probabilistic models.

3.2.1 Bases and Transformations

The basis transformations presented in the following sections have been selected using the different approaches discussed earlier, ensuring that they always match either the first or second sufficient statistic. Table 3.1 provides a comprehensive overview of the distributions, their transformations, their standard and new sufficient statistics, and their respective domains. This table is complemented by Table 3.2, which presents the explicit transformations between the parameters of various exponential families and a Gaussian for a given basis. For a visual interpretation of these transformations, we refer the reader to Figure 3.1, which illustrates three different distributions along with their corresponding Laplace approximations.

Although Figure 3.1 does not include all the distributions listed in Table 3.1, the Beta distribution can be considered as a representative for the Dirichlet distribution, as it is its one-dimensional special case. Similarly, the Gamma distribution can be seen as a representative for the Chi-square, (inverse-) Gamma, and (inverse-) Wishart distributions, since the Chi-square and Gamma distributions are one-dimensional special cases of the (inverse-) Wishart distribution. Visualizations of all distributions can be found in Appendix A.

Distribution	Basis	$\theta \to \mathcal{N}$	$\mathcal{N} \to \theta$
Exponential	log	$\mu = \log(\frac{1}{\lambda})$ $\sigma^2 = 1$	$\lambda = \frac{1}{\exp(\mu)}$
	sqrt	$\mu = \sqrt{\frac{1}{2\lambda}}$ $\sigma^2 = \frac{1}{4\lambda}$	$\lambda = \frac{1}{2\mu^2}$
Gamma	log	$\mu = \log\left(\frac{\alpha}{\lambda}\right)$ $\sigma^2 = \frac{1}{\alpha}$	$\alpha = \frac{1}{\sigma^2}$ $\lambda = \frac{1}{\exp(\mu)\sigma^2}$
	sqrt	$\mu = \sqrt{\frac{\alpha-0.5}{\lambda}}$ $\sigma^2 = \frac{1}{4\lambda}$	$\alpha = \frac{\mu^2}{4\sigma^2} - 0.5$ $\lambda = \frac{4}{\sigma^2}$
Inv. Gamma	log	$\mu = \log\left(\frac{\lambda}{\alpha}\right)$ $\sigma^2 = \frac{1}{\alpha}$	$\alpha = \frac{1}{\sigma^2}$ $\lambda = \frac{\exp(\mu)}{\sigma^2}$
	sqrt	$\mu = \sqrt{\frac{\lambda}{\alpha}}$ $\sigma^2 = \frac{\lambda}{4\alpha^2}$	$\alpha = \frac{\mu^2}{4\sigma^2} - 0.5$ $\lambda = \frac{\mu^4}{4\sigma^2}$
Chi-squared	log	$\mu = \log(k)$ $\sigma^2 = 2/k$	$k = \exp(\mu)$
	sqrt	$\mu = \sqrt{k}$ $\sigma^2 = 1/2$	$k = \mu^2$
Beta	logit	$\mu = \log(\frac{\alpha}{\beta})$ $\sigma^2 = \frac{\alpha+\beta}{\alpha\beta}$	$\alpha = \frac{\exp(\mu)+1}{\sigma^2}$ $\beta = \frac{\exp(-\mu)+1}{\sigma^2}$
Dirichlet*	softmax^{-1}	$\mu_k = \log\alpha_k - \frac{1}{K}\sum_{l=1}^{K}\log\alpha_l$ $\Sigma_{k\ell} = \delta_{k\ell}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_\ell} - \frac{1}{K}\sum_{u=1}^{K}\frac{1}{\alpha_u}\right]$	$\alpha_k = \frac{1}{\Sigma_{kk}}\left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2}\sum_{l=1}^{K} e^{-\mu_l}\right)$
Wishart	logm	$\mu = \text{logm}((n-p+1)V)$ $\Sigma = \frac{2}{(n-p+1)}(I_p \otimes I_p)^{-1}$	$V = \frac{\text{expm}(\mu)}{(n-p+1)}$ $n = \frac{2p^2}{\text{tr}(\Sigma)} + p - 1$
	sqrtm	$\mu = \text{sqrtm}\left((n-p)V\right)$ $\Sigma = \left(V^{\frac{1}{2}} \otimes V^{\frac{1}{2}}\right)\left(I_{p^2} + V^{\frac{1}{2}} \otimes V^{-\frac{1}{2}}\right)^{-1}$	$V = **$ $n = **$
Inv. Wishart	logm	$\mu = \text{logm}\left(\frac{1}{(\nu+p-1)\Psi}\right)$ $\Sigma = 2(\nu+p-1)(I_p \otimes I_p)^{-1}$	$\Psi = (\nu+p-1)\,\text{expm}(\mu)$ $\nu = \frac{\text{tr}(\Sigma)}{2p^2} - p + 1$
	sqrtm	$\mu = \text{sqrtm}(\frac{1}{(\nu+p)}\Psi^{-1})$ $\Sigma = \frac{1}{(\nu+p)^2}\left(I_p \otimes \Psi\right)\left(\Psi^{\frac{1}{2}} \otimes \Psi^{\frac{1}{2}} + I_p\right)^{-1}$	$\Psi = **$ $\nu = **$

Table 3.2: Overview of all closed-form transformations for Laplace Matching. * indicates an approximate inversion since the transformation is not bijective. ** means that the solution has to be looked up in Appendix A, which also contains practical tips for the (inverse-) Wishart.

3.3 Laplace Matching with Gaussian Processes

Algorithm 2 Laplace Matching + Gaussian Process (LM+GP) V1

Require: non-Gaussian data distribution d which is conjugate to an EF, Transformation

$\mathcal{EF}(x(f); \epsilon_\theta + \theta_d)$ ▷ Use EF to define pseudo-likelihoods in the data domain

$\mathcal{GP}(f; \mu_p, \Sigma_p) = LM^{-1}(\mathcal{EF})$ ▷ Use LM to map EF to Gaussian fit GP ▷ Apply standard GP toolbox on resulting Gaussian

$X' = x(y), y \sim \mathcal{GP}$ ▷ Draw samples from latent GP and apply transformation

return $\mathcal{GP}(f; \mu_p, \Sigma_p), X'$ ▷ Return GP and transformed samples

Remark 3.2 (LM+GP) We suggest two different ways to combine Laplace Matching and Gaussian Processes. For the rest of this thesis, we use the first definition, which we refer to as V1.

There are several approaches for performing latent Gaussian inference on non-Gaussian data (see Section 3.4 for a detailed comparison). Many of these methods, such as Variational Inference (VI), Expectation Propagation (EP), and Markov Chain Monte Carlo (MCMC), require multiple iterations of optimization or sampling, which can be computationally expensive. While there exist fast one-step approaches based on approximate inference, most of them are specialized for specific types of data. For example, Milios et al. (2018) focuses on classification, while Jia et al. (2021) is tailored for count data.

Algorithm 3 Laplace Matching + Gaussian Process (LM+GP) V2

Require: non-Gaussian data distribution d which is conjugate to an EF, Transformation
$\mathcal{GP}(f; \mu_0, \Sigma_0)$ ▷ Define prior GP in Gaussian latent space
$\mathcal{EF}(x(f); \theta_0) = LM(GP)$ ▷ Transform GP with LM
$\mathcal{EF}(x(f); \theta_0 + \theta_d)$ ▷ Update Exp. Fam. with data
$\mathcal{GP}(f; \mu_p, \Sigma_p) = LM^{-1}(\mathcal{EF})$ ▷ Transform back to latent Gaussian process
$X' = x(y), y \sim \mathcal{GP}(f; \mu_p, \Sigma_p)$ ▷ Draw samples from latent GP and apply transformation **return** $\mathcal{GP}(f; \mu_p, \Sigma_p), X'$ ▷ Return GP and transformed samples

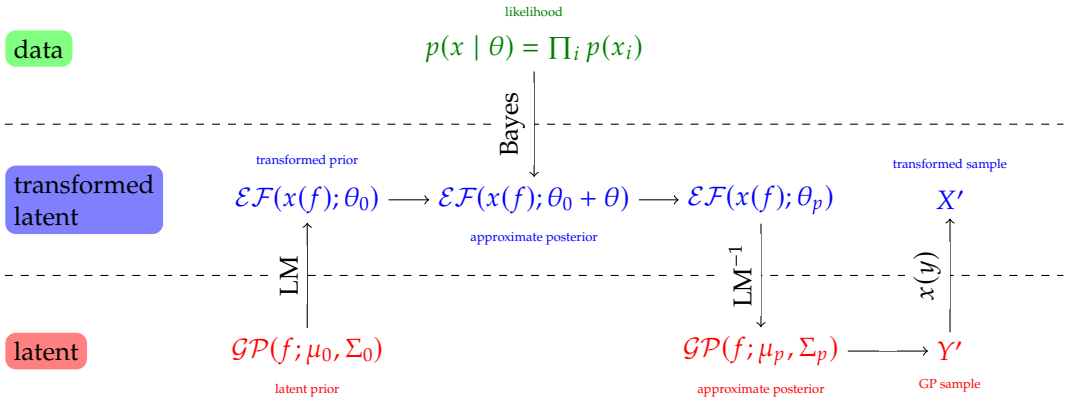


Figure 3.2: Visual description of Bayesian Inference with LM+GP. A latent prior GP is transformed to the latent exponential family of choice via Laplace Matching. It is then updated by the likelihood which is conjugate to the latent exponential family. This posterior distribution is then transformed back via LM and used to update the latent GP. From this approximate posterior GP we can draw samples and apply the original transformation $x(y) = g^{-1}(x)$ yielding a transformed sample in the desired space. Rather than using the full pipeline, we can skip the latent prior and replace it with a pseudo-likelihood (see Algorithm 2).

In contrast, Laplace Matching with Gaussian Processes (LM+GP) offers a more general framework that can be applied to any likelihood that is a conjugate prior to an exponential family, while maintaining the computational efficiency of specialized one-step approaches. The versatility of LM+GP lies in its ability to handle a wide range of data types without sacrificing speed. Moreover, Laplace Matching can be viewed as a fast pre-processing step that enables the application of conventional Gaussian inference tools to non-Gaussian data, thereby expanding the scope of these tools and facilitating their use in a broader range of scenarios.

Laplace Matching (LM) can be combined with Gaussian Processes (GPs) in two different ways. The first approach involves modeling the likelihood using an exponential family distribution and then applying LM to transform the likelihood into a Gaussian form. Subsequently, standard GP inference can be performed (see Algorithm 2). In this case, the exponential family is fitted by creating pseudo-observations for each data point or groups of data points. The choice of exponential family distributions is made such that their respective likelihoods are conjugate to them. Since LM is computationally efficient, the transformation cost is negligible, allowing GPs to be fitted on non-Gaussian data with minimal additional computational overhead (see Section 3.5.5 for further details).

The second approach involves defining a latent GP and transforming the GP prediction at the training data points using LM. The resulting

exponential family is then updated via conjugacy, and the updated exponential family is transformed back using the inverse LM map (see Algorithm 3 for details). This more elaborate approach may be preferable in cases where the prior GP is highly structured due to specific prior knowledge.

For the purposes of this thesis, we employ the first version of the algorithm, referred to as "LM+GP" for brevity, as it is simpler and faster to implement.

In both versions, the GP posterior in the latent domain can then be transformed into a posterior in the data domain either by applying $x(y)$ to samples from the latent Gaussian or by drawing latent sample functions and transforming them individually through LM^{-1}. Figure 3.2 shows a high-level summary of the concept for the entire inference pipeline.

By default, we employ pseudo-likelihoods to model the data points. For instance, in binary classification, a data point with label 1 would be modeled as $\mathcal{B}(1 + \epsilon_a, \epsilon_a)$. One of the benefits of selecting exponential families for the pseudo-likelihoods is the ease of grouping data points together through conjugacy. When multiple data points are close in time or belong to the same cluster after k-means clustering, they can be summarized using a single instance of the respective exponential family. This approach provides a natural way to group numerous data points into a smaller set of inducing points for Gaussian Process (GP) inference.

Remark 3.3 (Conjugacy and LM+GP) The conjugacy of exponential families allows us to already perform meaningful operations like grouping before transforming into the latent domain. This is a benefit compared to traditional GP approaches.

Since Laplace Matching (LM) provides both a mean estimate and a covariance estimate for each inducing point, we can employ a heteroskedastic noise model for our GP. Specifically, we can utilize the measured covariance obtained from LM as an approximation of the noise variable of the GP.

3.4 Related Work

The original idea to use basis changes to increase the quality of Laplace approximations goes back to D. J. MacKay (1998), who specifically studied Laplace approximations of Dirichlet distributions. This was extended to a bi-directional map by Hennig (2010). This work generalizes this idea to most of the popular exponential family distributions, provides explicit parameter mappings, and analyzes their costs and approximation quality.

There is a host of other approximate inference schemes for latent Gaussian models, including a Laplace approximation on the joint posterior (Williams et al., 1998), Expectation Propagation (Minka, 2001), Markov Chain Monte Carlo (MCMC) (Murray et al., 2009; Neal, 1993), variational approximations (Seeger et al., 2009; Challis et al., 2013), and Integrated Nested Laplace Approximations (INLA) (Rue et al., 2009; Martínez-Minaya et al., 2019).

Approximate inference schemes inevitably involve a trade-off between approximation quality and computational speed. Markov Chain Monte Carlo (MCMC) methods are asymptotically correct as the number of

samples approaches infinity. Variational approximations, on the other hand, converge to an approximation with a finite error within a finite amount of time. LM can be considered to lie at the extreme end of this spectrum, offering a closed-form approximation with a fixed quality.

Despite the fixed approximation error, the experiments presented in this paper demonstrate that the resulting approximation error of LM is often sufficiently small, making it a viable alternative to the aforementioned methods, especially when considering its very low computational cost (see Section 3.5.5). The computational efficiency of LM makes it an attractive first choice for various scenarios, such as testing purposes or low-level implementations where computational efficiency is of utmost importance.

3.5 Experiments

We first provide a number of experiments that explain the setup and provide quantitative results on four different non-Gaussian exponential families, namely the Beta (Section 3.5.1), Gamma (Section 3.5.2), Dirichlet (Section 3.5.3) and inverse Wishart (Section 3.5.4). Then we measure the time it took for LM to pre-process the data in all experiments in (Section 3.5.5) to showcase its speed. Finally, we investigate the quality of the proposed LM approximations by measuring the local KL divergence (Section 3.5.6). Code for all experiments is available.*

3.5.1 Binary Classification (Beta)

In this section, we demonstrate the application of Laplace Matching combined with Gaussian Processes (LM+GP) in binary classification. Binary classification predictions are probabilities, which poses a challenge when training a vanilla GP as it yields predictions in the wrong domain. To address this, we translate binary labels into Beta pseudo-likelihoods by adding a small value ϵ_a to both parameters. For example, a data point with label 1 would be modeled as $\mathcal{B}(1+\epsilon_a, \epsilon_a)$. Laplace Matching is then employed to map these pseudo-likelihoods to Gaussians, enabling the training of a vanilla GP on the resulting means and variances (see Figure 3.3).

The mapping between the Beta distribution parameters and the Gaussian parameters is given by:

$$\mu = \log\left(\frac{\alpha}{\beta}\right) \qquad \sigma^2 = \frac{\alpha+\beta}{\alpha\beta} \tag{3.15}$$

Since Laplace Matching provides estimates for both mean and variance, we can utilize the σ^2 to specify a heteroskedastic noise model. Furthermore, by leveraging the conjugacy of the Beta distribution, we can group multiple classes together, particularly when they are close in time (see bottom panel in Figure 3.3). This grouping allows us to use conjugacy to create a single "representative" for each cluster, which can then be used as inducing points.

* https://github.com/mariushobbhahn/Laplace_Matching_for_GLMs

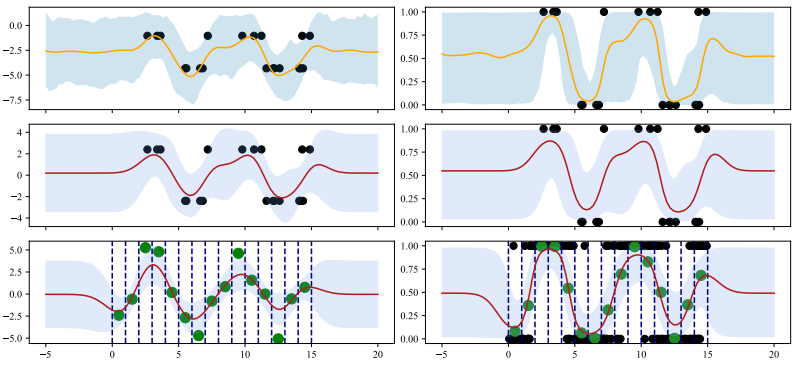


Figure 3.3: Conceptual figure for DirichletGPC and LM(Beta)+GP;
Left: latent GP in $\mathbb{R}$;
Right: transformed GP on the 1-simplex;
Top: DirichletGPC;
Middle: LM(Beta)+GP;
Bottom: LM(Beta)+GP using the conjugacy of the Beta to group datapoints that are close in time into one pseudo-likelihood

Table 3.3: Dataset specifications for classification experiments: Adapted from (Milios et al., 2018). The experiments with binary classification can be found in the Beta section and multi-class classification can be found in the Dirichlet section.

Dataset	Classes	Training instances	Test instances	Dimensionality	Inducing points
EEG	2	10980	4000	14	200
HTRU2	2	12898	5000	8	200
MAGIC	2	14020	5000	10	200
MINIBOO	2	120064	5000	50	400
LETTER	26	15000	2000	16	200
DRIVE	11	48509	2000	48	500
MOCAP	5	68095	2000	37	500

We have two options for making predictions: draw samples and transform them using the logistic function or translate the GP back via Laplace Matching. We use the toy example from Milios et al. (2018) to conceptually compare LM(Beta)+GP with DirichletGPC which is another fast way of doing GP classification. In Figure 3.3 we can see that LM(Beta)+GP produces estimates of similar visual quality to DirichletGPC but additionally allows for a simple way of aggregating inducing points.

Furthermore, we compare DirichletGPC with LM(Beta)+GP on four benchmarks from the UCI ML repository (Dua et al., 2017) that were already used in (Milios et al., 2018) (see Table 3.3). For all experiments classification experiments, we use an RBF kernel. All inducing points are the centers of k-means clustering applied to the training data set. We compare the results using the three common metrics of accuracy, mean negative log-likelihood (MNLL) and expected-calibration error (ECE). The results can be found in Figure 3.4. We find that LM(Beta)+DP is competitive with DirichletGPC and that LM(Beta)+GP using the conjugacy of the Beta to create inducing points often outperforms DirichletGPC. Since Milios et al. (2018) have extensively compared DirichletGPC with many other methods for binary GP classification, we argue that LM(Beta)+DP is also competitive with their comparisons.

3.5.2 Count data (Gamma)

In this section, we demonstrate the application of LM(Gamma)+GP on a toy dataset and compare its performance against various alternatives on a set of benchmarks.

We use a dataset from Bruin (2011) containing integer counts representing the number of awards and math scores of students belonging to three

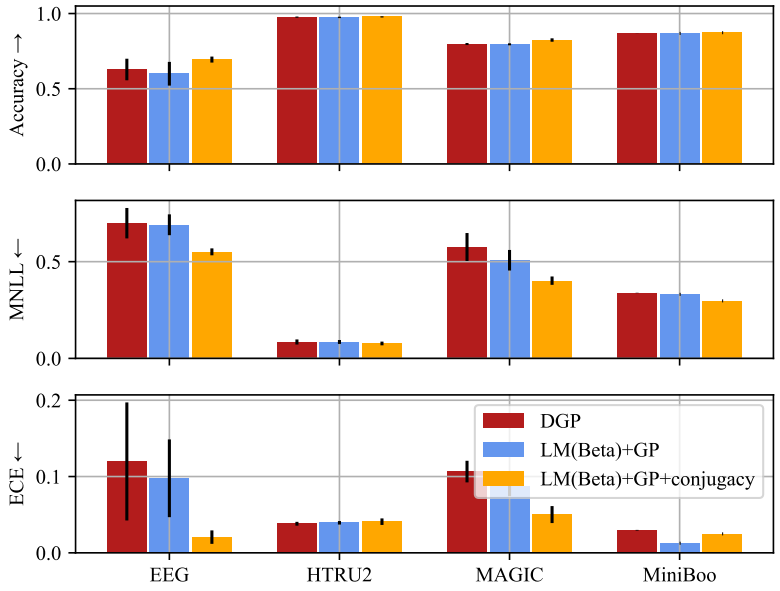


Figure 3.4: Results for binary classification experiment: LM(Beta)+GP is often competitive with DGPC, and outperforms both alternatives when the conjugacy of the Beta is leveraged to create inducing points.

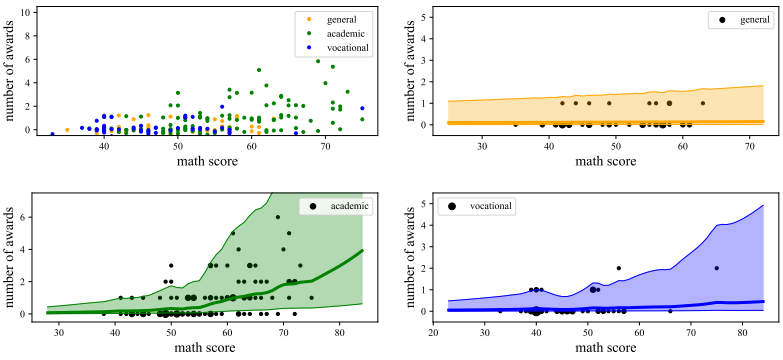


Figure 3.5: Probability of award as a function of math score. Top left: ground truth data (integers) with small noise added for visibility. The three other figures show LM+GP with count data and Gamma conjugate prior for the three classes respectively. The shadings in the figure show the *transformed* Gaussian measure on the domain $x = \exp(y)$ (hence the constraint to positive values).

groups: general, academic, and vocational. Given the nature of the data, where the number of awards are integers and reflect rare events, we assume a Poisson likelihood. The Gamma distribution serves as the conjugate prior for Poisson count data, allowing its parameters to be updated through simple addition. We construct a marginal Gamma pseudo-likelihood distribution $p(x(y))$ at each math score for all three classes. By applying LM in the log-base, we transform this Gamma distribution into a latent space $p(y)$ where a GP $q(y)$ is constructed:

$$\mu = \log\left(\frac{\alpha}{\lambda}\right) \qquad \sigma^2 = \frac{1}{\alpha} \tag{3.16}$$

As the kernel function, we employ a sum of a rational quadratic and a linear kernel. Through the inverse transformation, we represent the GP posterior back in the original space, $\mathbb{R}_+$, by transforming the mean function and two standard deviations using the exponential function.

The result is a GP $q(x(y))$ defined exclusively on the positive domain and fitted to Poisson likelihoods. The key advantage of this simple yet fast procedure is the ability to leverage the benefits of GPs on non-Gaussian data with minimal additional computational cost. Figure 3.5 illustrates the resulting GPs.

We evaluate the performance of LM(Gamma)+GP on five different

datasets from (Kleiber et al., 2008) where the target variable is a non-negative integer. LM(Gamma)+GP is well-suited for these variables since the Gamma distribution is defined on $\mathbb{R}^+$. We compare LM(Gamma)+GP against three alternative methods: a vanilla GP (ignoring the data domain), a GP trained on log-transformed data, and SVIGP on log-transformed data.

To assess the performance of each method, we employ three metrics: Root Mean Squared Error (RMSE), mean Negative Log-Likelihood (NLL) using Poisson likelihoods, and the number of test points within two standard deviations of the mean posterior prediction (in2std). Table 3.4 presents the results of our experiments.

The findings demonstrate that LM(Gamma)+GP achieves comparable or superior performance compared to the alternative methods in many cases. By leveraging the properties of the Gamma distribution and the flexibility of GPs, LM(Gamma)+GP effectively captures the characteristics of non-negative integer target variables.

Table 3.4: Quantitative results on count data. We find that GP+LM is either the best or comparable with alternative methods in many cases.

	vanilla GP			log GP			LM(Gamma)+GP			SVIGP		
Dataset	**RMSE ↓**	**MNLL ↓**	**in2std**	**RMSE ↓**	**MNLL ↓**	**in2std**	**RMSE ↓**	**MNLL ↓**	**in2std**	**RMSE ↓**	**MNLL ↓**	**in2std**
CreditCard	0.963	1.265	0.991	1.521	11.818	0.837	1.301	1.490	0.344	1.304	1.576	0.275
Doctor	0.851	1.247	0.982	0.910	4.618	0.265	0.780	0.704	0.169	0.811	0.724	0.008
Citations	57.996	46.393	0.256	66.491	104.156	0.047	41.556	15.038	0.972	51.950	26.337	0.888
GSS7402	1.255	1.628	0.949	2.249	2.921	0.253	1.262	1.633	0.428	1.312	1.656	0.207
Medicaid	3.662	2.665	0.948	4.120	22.337	0.748	3.590	2.580	0.489	3.831	3.367	0.132

3.5.3 Multi-class classification (Dirichlet)

To demonstrate the application of LM+GP to categorical data, we utilize the results from the German federal elections for the entire country between 1949 and 2017. The dataset contains T different election dates, P different political parties, and C different election counties. Each data point is represented as a tuple $(\mathsf{t}, \mathsf{p}, \mathsf{c})$ with an associated vote count v for party p in election year t and county c.

For every year t and county c, we construct a Dirichlet distribution given the p parties using $p(x(y)) = \mathcal{D}_{t,c}(v_p)$. This is possible because the votes form a categorical distribution, for which the Dirichlet distribution serves as a conjugate prior. To transform the Dirichlet distribution into a latent Gaussian distribution $q(y)$ and vice versa, we employ the LM equations:

$$\mu_k = \log \alpha_k - \frac{1}{K} \sum_{l=1}^{K} \log \alpha_l \tag{3.17a}$$

$$\Sigma_{k\ell} = \delta_{k\ell} \frac{1}{\alpha_k} - \frac{1}{K} \left[\frac{1}{\alpha_k} + \frac{1}{\alpha_\ell} - \frac{1}{K} \sum_{u=1}^{K} \frac{1}{\alpha_u} \right] \tag{3.17b}$$

$$\alpha_k = \frac{1}{\Sigma_{kk}} \left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2} \sum_{l=1}^{K} e^{-\mu_l} \right). \tag{3.17c}$$

to yield $\mu_{\mathsf{t},\mathsf{c}}$ and $\Sigma_{\mathsf{t},\mathsf{c}}$. Since some parties received 0 votes in some elections

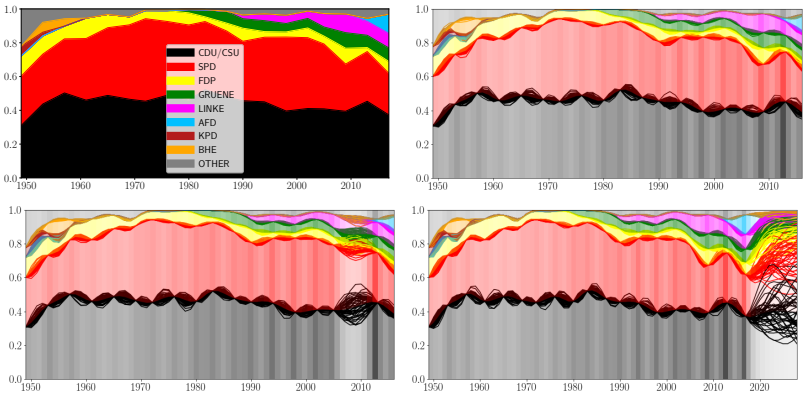


Figure 3.6: German elections from 1949 to 2017. Top left: Ground truth data points; Top right: LM+GP cumulative predictions for all parties with attached uncertainty (dark means low uncertainty); Bottom left: 2009 election has been left out of the training data; Bottom right: predictions for the next 3 elections (2021, 2025, 2029). The uncertainties are meaningful, e.g. the existence of data implies low uncertainty and vice versa.

we set the prior of each Dirichlet to $\alpha = 1$. The likelihood for all election data under the latent Gaussian is then given by

$$q(y) = LM^{-1}(p(v|\pi_d)) = \prod_d LM^{-1}(\mathcal{D}(v_d))) = \prod_d \mathcal{N}(f; \mu_d, \Sigma_d), \quad (3.18)$$

where f describes the latent Gaussian. Integrating all of the above into the general equation for GP regression (Rasmussen, 2006)

$$q(f|v) = \mathcal{GP}\left(f; m_k + k_{xX}(k_{XX} + \Sigma_X)^{-1}(\mu_X - m_d), k_{xx} - k_{xX}(k_{XX} + \Sigma_X)^{-1}k_{Xx}\right) \quad (3.19)$$

we can construct a $N = T \cdot P \cdot C$ dimensional GP over all combinations of elections, parties and counties. The kernel function k is defined as

$$k[(\mathsf{t}, \mathsf{p}, \mathsf{c}), (\mathsf{t}', \mathsf{p}', \mathsf{c}')] = k_T(\mathsf{t}, \mathsf{t}') \cdot k_P(\mathsf{p}, \mathsf{p}') \cdot k_C(\mathsf{c}, \mathsf{c}') \quad (3.20)$$

where the kernels k_T, k_P and k_C can be hand-crafted to their specific domain. The party kernel k_P could, for example, display the distance of parties on the political spectrum or the county kernel k_C could incorporate information about the relative differences in average income, education, crime, etc. as long as they correlate with voting behavior. The kernel matrix $k_{XX} \in \mathbb{R}^{N \times N}$ is created through Equation 3.20. The noise matrix $\Sigma_X \in \mathbb{R}^{N \times N}$ is a block diagonal matrix with $T \cdot C$ blocks of size $P \times P$ containing the respective $\Sigma_{\mathsf{t},\mathsf{c}}$ from the Laplace Matching.

In our experiment, we restrict ourselves to the $T = 19$ elections held between 1949 and 2017, to the $P = 9$ political parties that gained more than 5% of the vote at least once in these elections, and to the 9 federal states that continuously existed across this time-frame – plus Germany as a whole, such that $C = 10$.

After building the GP we can draw sample functions at different points in time and transform them back to probability space through the softmax function $\sigma(y) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$ to get our GP predictions. The resulting election landscape can be found in Figure 3.6.

Timing & Efficiency

To evaluate the speedup of LM+GP over alternatives, we compare LM+GP with elliptical slice sampling (Murray et al., 2009) (ESS). ESS is an efficient

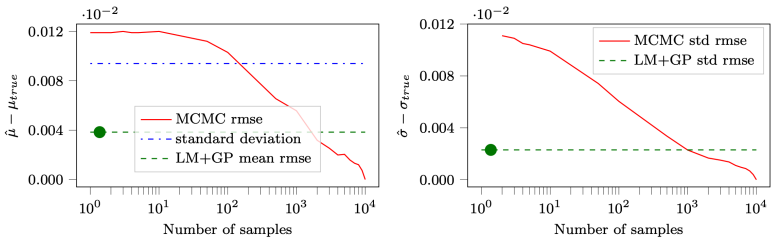


Figure 3.7: Comparison of approximation quality of LM vs MCMC-samples dependent on the number of samples. In both cases, the approximation quality of the samples meets that of LM after ca. 1000 samples. The green dot indicates the ratio of drawing one sample vs. applying LM. We see that LM costs slightly more than drawing one ESS sample to compute the latent Gaussian.

sampling method specifically designed for the combination of non-Gaussian likelihoods and GP priors. We calculate the log-likelihood for ESS with

$$l(\mathsf{t},\mathsf{c},\mathsf{p}) = \log\left(\prod_{\mathsf{t}}\prod_{\mathsf{c}}\prod_{\mathsf{p}} \sigma(f(\mathsf{t},\mathsf{c}))_p^{y_{\mathsf{t},\mathsf{c},\mathsf{p}}}\right) = \sum_{\mathsf{t}}\sum_{\mathsf{c}}\sum_{\mathsf{p}} y_{\mathsf{t},\mathsf{c},\mathsf{p}} \log(\sigma(f(\mathsf{t},\mathsf{c})) \tag{3.21}$$

The predictions with ESS are made by projecting the samples $\hat{f}$ to the desired space with

$$q(f_x|y,\hat{f}) = \sum_i \mathcal{N}\left(f_x; m_x + k_{xX}k_{XX}^{-1}(\hat{f}_i - m_x), k_{xx} - k_{xX}k_{XX}^{-1}k_{Xx}\right) \tag{3.22}$$

To evaluate the timing trade-off between a fast but fixed approximation strategy like Laplace Matching (LM) and a more computationally intensive but asymptotically exact iterative strategy such as Elliptical Slice Sampling (ESS), we compare the approximation quality on a significantly smaller subset of the data. Specifically, we focus on the local results (approximately 1000 votes per election) of a single voting precinct (the "Wanne" neighborhood in Tübingen) from 2002 to 2017. The true distribution is approximated by drawing 10000 samples from ESS. Figure 3.7 presents the results of this comparison. We observe that it requires around 1000 MCMC samples for ESS to achieve the same quality of mean and variance estimates as the static LM results. Moreover, the difference between the mean obtained from LM and the true mean is smaller than one standard deviation of the true distribution. Since the standard deviation serves as a measure of aleatoric uncertainty in our posterior, having the LM mean within one standard deviation of the true mean provides strong evidence for the quality of the LM approximation.

Quality of the Marginals

To evaluate the quality of the predictions provided by LM+GP in comparison to sampling-based methods, we utilize their marginal predictions on the smaller dataset. We compare three different prediction approaches: samples obtained through ESS, samples from LM+GP transformed via the softmax function, and Beta marginals from the Dirichlet distribution generated directly by the inverse LM transformation from the GP, since variable x_i from a Dirichlet distribution follows a Beta$(\alpha_i, \sum_{j\neq i}\alpha_j)$ distribution. Figure 3.8 illustrates the resulting marginals.

Our findings reveal two key observations. First, the transformed samples

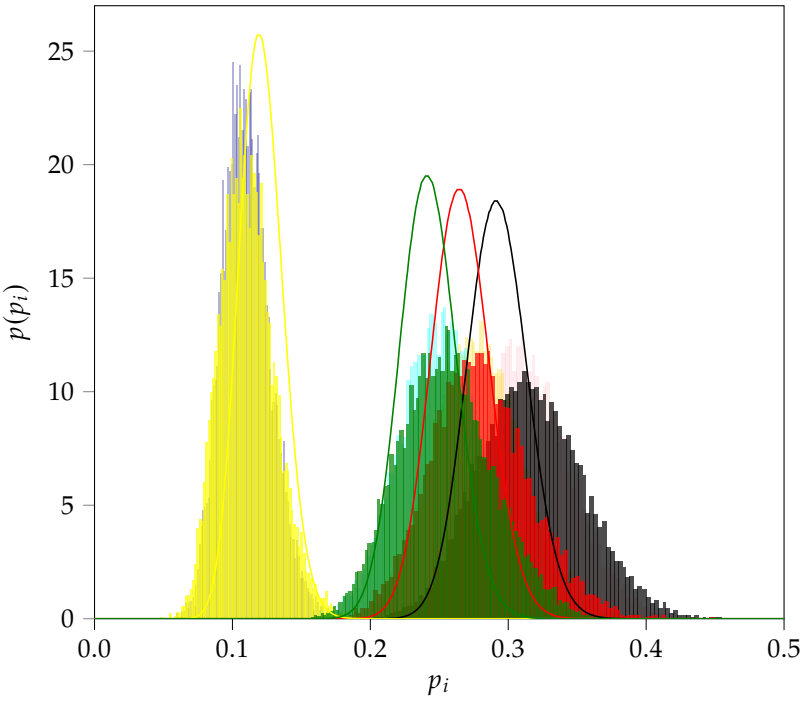


Figure 3.8: Marginal plot of the four largest parties for the prediction of a local election in 2004 (i.e. when no actual election took place). We compare predictions of ESS samples (dark histograms) with transformed samples from LM+GP (light histograms) and Beta marginals from the LM-Dirichlet transformation (solid lines). The reasons for their differences are discussed in the text.

from LM+GP exhibit slight differences compared to the Beta marginals obtained from the direct transformation. This discrepancy arises because the inverse transformation for the LM-Dirichlet is an approximation and not a bijective transformation. Second, the samples from ESS and the transformed samples from LM+GP show high similarity, with only minor differences. This can be attributed to the use of a small uniform prior for LM+GP, which is absent in ESS. Consequently, smaller probabilities are shifted to the right, while larger probabilities are shifted to the left, precisely as observed in Figure 3.8.

Although both approximations to ESS are not perfectly exact, they offer a trade-off in terms of computational speed. The LM-transformation has a computational complexity of $\mathcal{O}(n)$, sampling from LM+GP takes $\mathcal{O}(n^2)$, while sampling for ESS incurs a cost of $\mathcal{O}(n^3)$.

Quantitative Experiments

In addition to the conceptual and timing experiments, we compare LM(Dirichlet)+GP with DirichletGPC (Milios et al., 2018) on three of the benchmarks provided in their paper. The setup is similar to the Beta experiments detailed in Subsection 3.5.1 and the details of the datasets can be found in Table 3.3. The results of the experiments can be found in Figure 3.9. We find that LM(Dirichlet)+GP is competitive or better than DirichletGPC.

3.5.4 Currencies & Covariances (Inverse Wishart)

In this experiment, we aim to estimate the covariance of currency exchange rates over time. The inverse Wishart distribution is a conjugate prior for covariance matrices. However, modeling temporal variations within

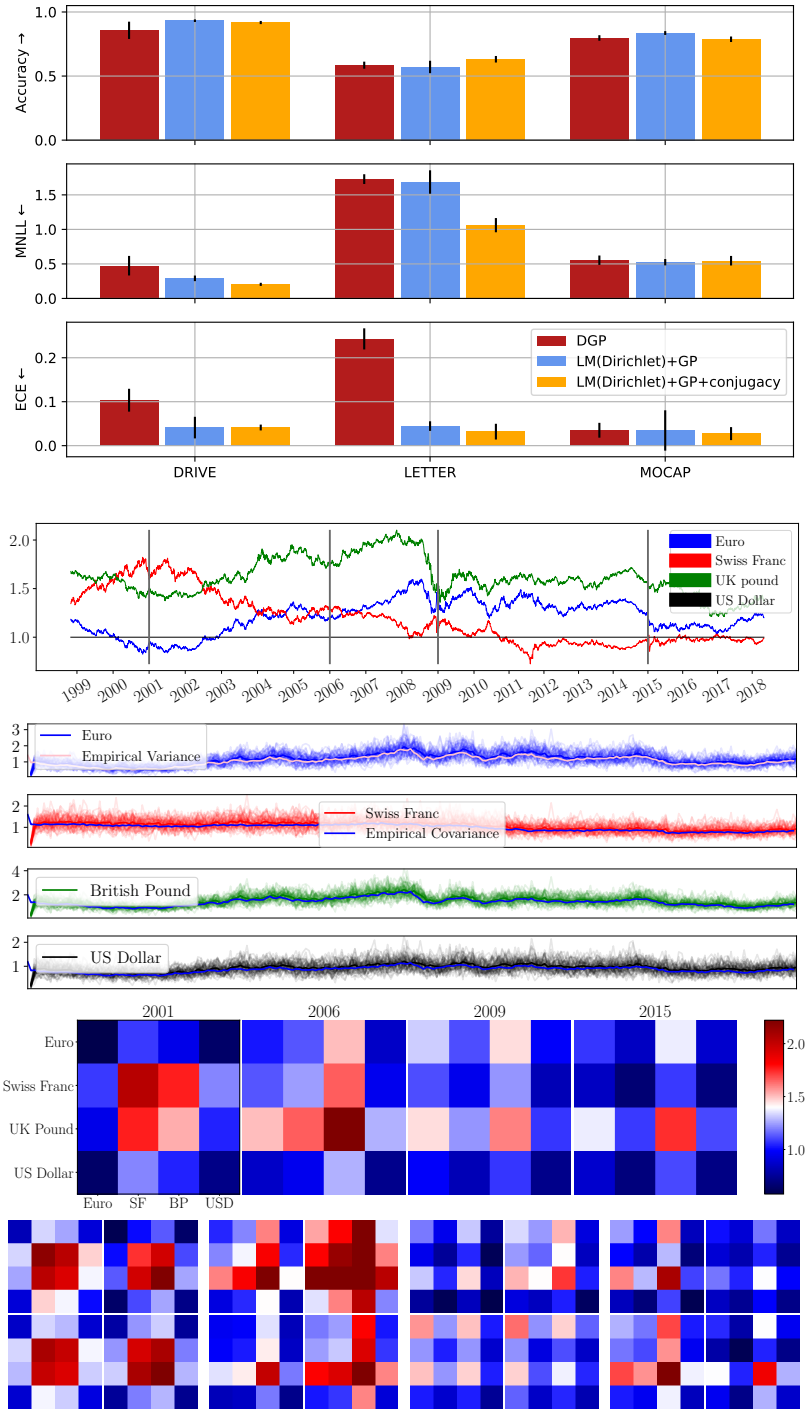


Figure 3.9: Results of the multiclass classification. We find that LM(Dirichlet)+GP is competitive or better than DirichletGPC.

Figure 3.10: LM+GP applied to the covariances of 4 currencies (Euro, Swiss Franc, UK Pound and US Dollar).
Top: the values of the currencies;
Middle: sample covariances from the posterior GP;
Bottom: selected covariance matrices from the Posterior - the grey lines (top) show the date of the samples.

the Wishart framework presents challenges (further discussed below). LM+GP offers a solution by transforming the cavity distribution at each datapoint $XX^\top$ of the inverse Wishart into a Gaussian. By applying LM and following a similar approach to the previous sections, we can create a GP that captures the temporal dynamics of the covariance structure. We use

$$\mu = \text{sqrtm}\left(\frac{1}{(\nu+p)}\Psi^{-1}\right) \tag{3.23a}$$

$$\Sigma = \frac{1}{(\nu+p)^2}\left(I_p \otimes \Psi\right)\left(\Psi^{\frac{1}{2}} \otimes \Psi^{\frac{1}{2}} + I_p\right)^{-1} \tag{3.23b}$$

to compute the mean and covariance function for the inverse Wishart (see Appendix A for derivation). This time we have $T = 235$ months between 1998 and 2018 with $C = 4$ currencies (Euro, Swiss Franc, UK Pound, US Dollar).

Designing a custom kernel enables the incorporation of information about real-world events, such as financial crises, or statistical correlations, like the relationship between GDP and currency value, into our model. This flexibility allows us to tailor the model to specific domain knowledge and capture relevant patterns in the data.

To illustrate the potential benefits of the additional uncertainty provided by LM+GP, we visualize the covariance of different sample functions drawn from the GP posterior and different samples of covariance matrices at four distinct points in time, as shown in Figure 3.10. The results demonstrate the high quality of the resulting posterior, particularly considering the simplicity of the LM+GP approach.

We also compare LM(inv. Wishart)+GP with the results of a vanilla GP fit on the covariances in Table 3.5. We find that the quality of the approximation is comparable on the currency dataset but the LM+GP variant has the advantage of actually reflecting covariance matrices, i.e. p.s.d matrices that correspond to the covariances of an underlying process rather than matrices that just roughly track values that happen to be covariances.

Table 3.5: Quantitative results on currency data. We find that both methods have comparable accuracy but LM+GP is slightly better calibrated (0.95 is optimal for in2std).

	vanilla GP		LM(inv. Wishart)+GP	
Dataset	**RMSE ↓**	**in2std**	**RMSE ↓**	**in2std**
Currencies	0.074	1.0	0.123	0.973

In general, modeling time-dependent covariance matrices is non-trivial. (A. Wilson et al., 2010) developed the Generalized Wishart Process (GWP), which uses a product of Gaussian processes in combination with MCMC to model time-dependent covariance matrices. However, we observed empirically that the associated MCMC method mixes very slowly, to the point where it did not yield usable results for us, even after 24h of wall-clock time on the currency dataset. In contrast, LM(inv. Wishart)+GP trained in a few seconds and provided good results. Table 3.6 provides a comparison of the timing and complexity.

Table 3.6: Timing and Complexity for Bayesian Inference on p.s.d. matrices: The LM+GP step is split into an inference and a transformation step. The complexity of the transformation step depends on the basis transformation. For our experiment we chose the sqrtm-transformation as it yielded better results. Since $D << n_X$ LM+GP is much faster than GWP. For both methods $n_X = 235$ and $D = 4$. For GWP we draw $n_s = 5000$ samples of which we only used 100 since they were very correlated.

	Inference	μ_X, Σ_X (LM)	GWP
Complexity	$\mathcal{O}(D^2 n_x n_X^2 + D^2 n_X^3)$	$\mathcal{O}(n_X D^{2.2})/\mathcal{O}(n_X D^2)$	$\mathcal{O}(n_s(D n_X^3 + \nu D^2))$
Time in s	1.59	0.09	82530

3.5.5 Timing

Table 3.7 provides time overhead for the application of LM across the datasets and settings presented in the experiments. We note that this

Table 3.7: Timings and complexity for LM in all experiments: In all cases, LM costs less than 0.1 seconds to apply and scales linearly in the number of datapoints.

Dataset	Outputs	Training instances	Dimensionality	Time in s	Complexity
EEG	2	10980	14	0.0024	$\mathcal{O}(N)$
HTRU2	2	12898	8	0.0067	$\mathcal{O}(N)$
MAGIC	2	14020	10	0.0021	$\mathcal{O}(N)$
MINIBOO	2	120064	50	0.0037	$\mathcal{O}(N)$
CREDIDCARD	1	833	12	0.0018	$\mathcal{O}(N)$
DOCTOR	1	3477	11	0.0017	$\mathcal{O}(N)$
CITATIONS	1	434	12	0.0005	$\mathcal{O}(N)$
GSS7402	1	2219	9	0.0004	$\mathcal{O}(N)$
MEDICAID	1	667	13	0.0004	$\mathcal{O}(N)$
LETTER	26	15000	16	0.0149	$\mathcal{O}(NO)$
DRIVE	11	48509	48	0.0077	$\mathcal{O}(NO)$
MOCAP	5	68095	37	0.0082	$\mathcal{O}(NO)$
CURRENCY	16	235	4	0.0653	$\mathcal{O}(NO^2)$

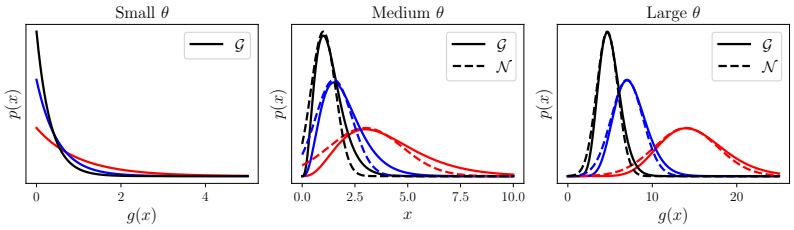


Figure 3.11: Visualization of small ($\alpha = 1, \lambda = 1$), medium ($\alpha = 4, \lambda = 2$) and large ($\alpha = 15, \lambda = 3$) θ for the Gamma distribution.

number never exceeded 0.1s, corroborating that LM can indeed be thought of as a fast pre-processing step for standard Gaussian inference.

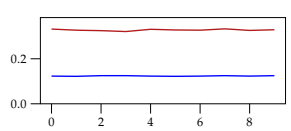
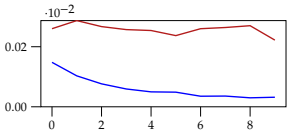
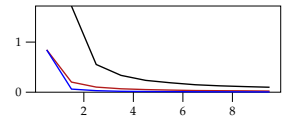
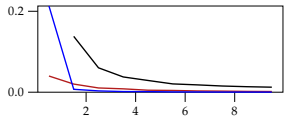
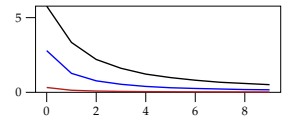
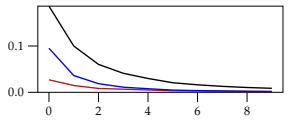
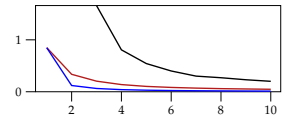
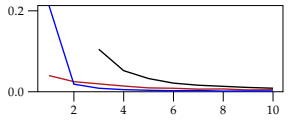
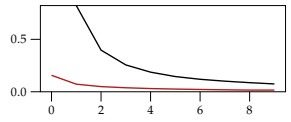
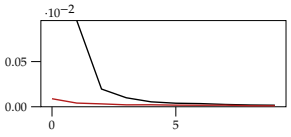
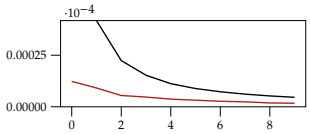
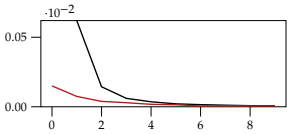
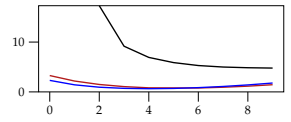
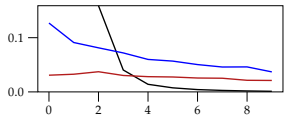
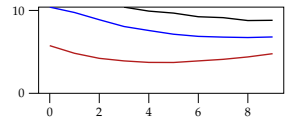
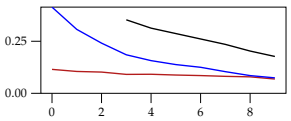
3.5.6 Distance Measures

Figure 3.1 suggests that the distributions are more Gaussian in other bases. We aim to quantify this similarity by comparing each approximation to the true distribution in every base for ten sets of parameters θ using two metrics. The ten parameter sets are selected to increase linearly, representing a cross-section of the entire parameter space. The parameter sets are chosen such that the first two sets yield no valid Laplace approximation in the standard base but do yield one in the transformed basis (e.g., $\alpha < 1$ for the Gamma distribution). This choice emphasizes the capability of LM to handle such cases.

For large parameter values, many exponential family distributions approach a Gaussian distribution (e.g., Gamma with $\alpha > 10$). The parameter sets are increased in a manner that ensures the last set reflects this property. To illustrate these scenarios, Figure 3.11 presents a Gamma distribution with $\alpha = 1, 4, 15$ and $\lambda = 1, 2, 3$ for small, medium, and large scenarios, respectively. By comparing the approximations to the true distribution across different bases and parameter sets, we can assess the effectiveness of LM in capturing the Gaussian behavior of the distributions.

The first metric to compare the distributions is KL-divergence and the second is *maximum mean discrepancy* (MMD) (Gretton et al., 2012). We calculate the KL-divergence by drawing samples from the distribution we

Table 3.8: Comparison of Distances. MMD and KL-divergence for increasing values of θ. —— shows distances in the standard base, —— when transformed by the logarithm, logit or inverse-softmax, and —— by the square root. In general, the approximation in the standard base is worse than in the transformed base, and all approximations improve towards larger θs. See Appendix A.3 for more detailed explanations and further discussion.

Distribution	KL-Divergence ↓	MMD ↓
Exponential		
Gamma		
Inv. Gamma		
Chi-squared		
Beta		
Dirichlet		
Wishart		
Inv. Wishart		

want to approximate, i.e. $x \sim p(x)$, and then compute $\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{p(x)}{q(x)}\right)$ where $q(x)$ is the Gaussian given by the Laplace approximation. A table of the exact parameters chosen can be found in Appendix A. The results can be found in Table 3.8.

We find that both the KL-divergence and MMD are are smaller in the base provided by LM compared to the standard base. This suggests that LM offers a more suitable representation for approximating exponential family distributions with Gaussians. Additionally, we observe that both metrics decrease for the standard base as the parameters increase, indicating that exponential families exhibit more Gaussian-like behavior when their parameters are large. It is worth noting that no single transformation consistently outperforms the others across all exponential families. Consequently, we recommend selecting the appropriate transformation based on the specific requirements and characteristics of the application at hand.

3.6 Conclusion

We introduce an approximate inference scheme designed to achieve fast computation while maintaining sufficient approximation quality. The proposed approach uses a change of basis for various exponential family distributions, followed by the application of a Laplace approximation and the derivation of an explicit closed-form transformation between the parameters of the resulting Gaussian and the original distribution. This process enables the use of closed-form Gaussian (process) inference on non-Gaussian data without requiring any iterative optimization scheme, provided that the data type has an exponential family conjugate prior. We present two closed-form transformations for each of the most common exponential families, showcasing the simplicity and functionality of Laplace Matching by applying a GP to non-trivial data types, including binary data, count data, categorical data, and positive semi-definite matrices. In all cases, we provide conceptual and empirical evidence supporting the approximation quality of the model.

3.7 Practical Advice & Honest Thoughts

Laplace Matching's primary strength lies in its speed and ease of implementation. There is a wide range of well-established Gaussian-based techniques available for various use-cases and applications. Laplace Matching simplifies the process of extending the Gaussian toolbox to arbitrary data types, provided they have an exponential family conjugate prior.

However, the fixed nature of the approximation in Laplace Matching can be limiting in certain scenarios. Unlike iterative methods, it does not improve with additional computational resources, which may be frustrating in cases where better results are desired but unattainable. Determining the quality of the approximation in advance can be challenging. It is often difficult to predict whether Laplace Matching will provide a good approximation for a specific problem.

In practice, I recommend using Laplace Matching as a baseline. It is cheap to set up and significantly faster than iterative approximate inference schemes. Throughout the development of this project, I was pleasantly surprised by how quickly and effectively Laplace Matching combined with Gaussian Processes (LM+GP) performed out of the box when integrated with existing GP libraries. In many of my experiments, alternative methods that were conceptually more sophisticated, more complex to implement, and more computationally demanding did not yield substantially better results compared to Laplace Matching.

4. Laplace Bridge for Bayesian Neural Networks

Remark 4.1 The contents of this chapter are primarily based on:

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. "*Fast Predictive Uncertainty for Classification with Bayesian Deep Networks*". Conference on Uncertainty in Artificial Intelligence (UAI). 2022.

	Idea	Analysis	Exp.	Code	Writing
Marius Hobbhahn	40%	80%	80%	100%	80%
Agustinus Kristiadi	20%	10%	10%	0%	10%
Philipp Hennig	40%	10%	10%	0%	10%

Abstract

In Bayesian Deep Learning, distributions over the output of classification neural networks are often approximated by first constructing a Gaussian distribution over the weights, then sampling from it to receive a distribution over the softmax outputs. This is costly. We reconsider old work (Laplace Bridge) to construct a Dirichlet approximation of this softmax output distribution, which yields an analytic map between Gaussian distributions in logit space and Dirichlet distributions (the conjugate prior to the Categorical distribution) in the output space. Importantly, the vanilla Laplace Bridge comes with certain limitations. We analyze those and suggest a simple solution that compares favorably to other commonly used estimates of the softmax-Gaussian integral. We demonstrate that the resulting Dirichlet distribution has multiple advantages, in particular, more efficient computation of the uncertainty estimate and scaling to large datasets and networks like ImageNet and DenseNet. We further demonstrate the usefulness of this Dirichlet approximation by using it to construct a lightweight uncertainty-aware output ranking for ImageNet.

4.1 Introduction

In Neural Network classification, a softmax function is applied after the final layer. Therefore, the output of classification networks is a vector of probabilities. In theory, these probabilities could be used to quantify the uncertainty of the Neural Network which matters in safety-critical applications such as self-driving cars (McAllister et al., 2017; Michelmore et al., 2018) or medical AI (Begoli et al., 2019). However, it has been shown that this vector of probabilities is overconfident (Nguyen et al., 2015; Hein et al., 2019). Many approaches attempt to provide more well-calibrated quantified uncertainty predictions (e.g. Blei et al., 2017a)

The predictive distribution from these approaches is typically non-analytic, i.e. they are approximated via Monte Carlo (MC) integration. MC integration is an unbiased estimator which makes it desirable from a theoretical perspective. In practice, on the other hand, sampling increases computational costs. The more classes the classification task has, the larger the distribution's dimensionality and therefore the sampling procedure's computational requirements.

Thus, practitioners face a trade-off between asymptotically exact but more expensive methods with MC sampling and fast but less exact approximations. Both ends of this spectrum have plausible use-cases. In the following, we describe an approach that falls on the fast-but-approximate end end of the spectrum: the Laplace Bridge for Bayesian Neural Networks.

The Laplace Bridge (LB) has been proposed in D. J. MacKay (1998) in a different setting (which is arguable the inverse of the Deep Learning setting). The LB applies a transformation of variable to a Dirichlet distribution using the inverse-softmax function. The shape of the transformed Dirichlet effectively approximates a Gaussian. This allows us to fit a Laplace approximation which yields an analytical map between the parameters of a Gaussian and a transformed Dirichlet.

As described by Hennig (2010) the original LB can be inverted and yield a mapping from Gaussian to Dirichlet. Approximate Gaussian inference in Bayesian NNs often use MC integration to translate Gaussian samples through the softmax into a distribution over probabilities. The Dirichlet is a distribution over probability vectors and we therefore propose to leverage the LB as a mapping from Gaussian to Dirichlet for very fast inference in Gaussian-based BNNs. Figure shows a sketch of the LB.

4.2 The Laplace Bridge

The LB used in this chapter is equivalent to Laplace Matching for the Dirichlet which was already described in Chapter 3. However, since it is the key feature of this chapter, we will briefly revisit the most important considerations.

In the standard form, the Dirichlet distribution has the density function

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1}, \tag{4.1}$$

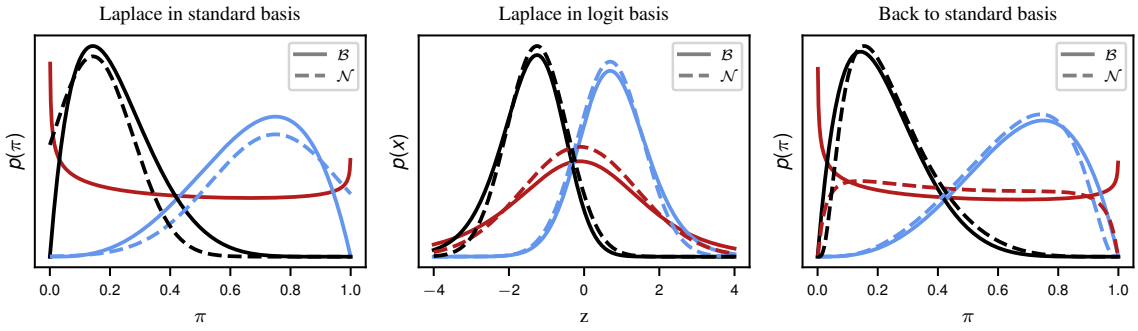


Figure 4.1: (Adapted from Hennig, Stern, et al. (2012)). Visualization of the Laplace Bridge for the Beta distribution (1D special case of the Dirichlet) for three sets of parameters. **Left:** "Generic" Laplace approximations of standard Beta distributions by Gaussians. Note that the Beta Distribution (red) does not have a valid approximation because its Hessian is not positive semi-definite. **Middle:** Laplace approximation to the same distributions after basis transformation through the softmax equation A.34a. The transformation makes the distributions "more Gaussian" (i.e. uni-modal, bell-shaped, with support on the real line), thus making the Laplace approximation more accurate. **Right:** The same Beta distributions, with the back-transformation of the Laplace approximations from the middle figure to the simplex, yielding an improved approximate distribution. In contrast to the left-most image, the dashed lines now actually are probability densities (they integrate to 1 on the simplex). The parameters are from left to right$\alpha, \beta = (0.8, 0.9), (4, 2,), (2, 7)$.

It is defined on the probability simplex and can be "multimodal" in the sense that the distribution diverges in the k-corner of the simplex when $\alpha_k < 1$. These two constraints preclude a Laplace approximation in the standard basis.

However, when we apply a variable transformation using the inverse-softmax function, we can elegantly fix both problems. Consider the K-dimensional variable $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$ defined as the softmax of $\mathbf{z} \in \mathbb{R}^K$:

$$\pi_k(\mathbf{z}) := \frac{\exp(z_k)}{\sum_{l=1}^K \exp(z_l)}\,, \tag{4.2}$$

for all $k = 1, \ldots, K$. We will call $\mathbf{z}$ the logit of $\boldsymbol{\pi}$. When expressed as a function of $\mathbf{z}$, the density of the Dirichlet in $\boldsymbol{\pi}$ has to be multiplied by the absolute value of the determinant of the Jacobian

$$\det \frac{\partial \boldsymbol{\pi}}{\partial \mathbf{z}} = \prod_k \pi_k(z_k), \tag{4.3}$$

thus removing the "-1" terms in the exponent:

$$\mathrm{Dir}_{\mathbf{z}}(\boldsymbol{\pi}(\mathbf{z})|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \pi_k(\mathbf{z})^{\alpha_k} \tag{4.4}$$

This density of $\mathbf{z}$, the Dirichlet distribution in the *softmax basis*, can now be accurately approximated by a Gaussian through a Laplace approximation (see Figure 4.1), yielding an analytic map from the parameter $\boldsymbol{\alpha} \in \mathbb{R}_+^K$ to the parameters of the Gaussian ($\boldsymbol{\mu} \in \mathbb{R}^K$ and symmetric positive definite $\boldsymbol{\Sigma} \in \mathbb{R}^{K \times K}$), given by

$$\mu_k = \log \alpha_k - \frac{1}{K} \sum_{l=1}^K \log \alpha_l\,, \tag{4.5}$$

$$\Sigma_{k\ell} = \delta_{k\ell} \frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_\ell} - \frac{1}{K} \sum_{u=1}^K \frac{1}{\alpha_u}\right]. \tag{4.6}$$

The corresponding derivations require care because the Gaussian parameter space (N, NxN) is evidently larger than that of the Dirichlet (N) and not fully identified by the transformation. A pseudo-inverse of this map was provided in Hennig, Stern, et al. (2012). It maps the Gaussian parameters to those of the Dirichlet as

$$\alpha_k = \frac{1}{\Sigma_{kk}} \left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2} \sum_{l=1}^{K} e^{-\mu_l} \right) \tag{4.7}$$

Together, Eqs. A.43a, A.43b and A.44 will be called the *Laplace Bridge*. For Bayesian Deep Learning, we only use Equation A.44 which maps from $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ to $\boldsymbol{\alpha}$. Due to the different sizes of the parameter spaces, the LB implies a reduction of the distribution's expressiveness. We investigate the effects of this reduction in Section 4.4 and 4.5.

4.3 The Laplace Bridge for BNNs

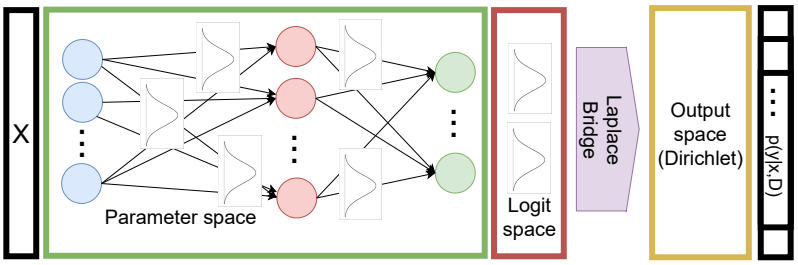


Figure 4.2: High-level sketch of the Laplace Bridge for BNNs. $p(y|x, D)$ denotes the marginalized softmax output, i.e. the mean of the Dirichlet.

Our goal is to provide an analytic transformation that is as fast as possible. Therefore, we use a last-layer Laplace approximation of the network. However, in theory, the Laplace Bridge can be applied to any NN setup that maps from a Gaussian to a distribution over probabilities by using the softmax including all-layer Laplace approximations or variational inference. The last-layer Laplace approximation, as e.g. used by Kristiadi et al. (2020) and Snoek et al. (2015), is given by

$$q(\mathbf{z}|\mathbf{x}) \approx \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{\mathbf{W}^{(L)}}\phi(\mathbf{x}), \phi(\mathbf{x})^T \boldsymbol{\Sigma}_{\mathbf{W}^{(L)}} \phi(\mathbf{x})) \,, \tag{4.8}$$

where $\phi(\mathbf{x})$ denotes the output of the first $L-1$ layers, $\boldsymbol{\mu}_{\mathbf{W}^{(l)}}$ is the maximum a posteriori (MAP) estimate for the weights of the last layer, and $\boldsymbol{\Sigma}_{\mathbf{W}^{(l)}}$ is the inverse of the negative loss Hessian w.r.t. $\mathbf{W}^{(l)}$, $\boldsymbol{\Sigma}_{\mathbf{W}^{(L)}} = -(\nabla^2_{\mathbf{W}^{(L)}} \mathcal{L})^{-1}$ around the MAP estimate $\mathbf{W}^{(L)}$. In all cases we use diagonal and Kronecker approximations to the Hessian.

The LB provides an *analytical* approximation to the density of the softmax-Gaussian output of the BNN. As shown in Eq. A.44, it requires $\mathcal{O}(K)$ computations to construct the K parameters α_k of the Dirichlet. In contrast, MC-integration has computational costs of $\mathcal{O}(MJ)$, where M is the number of samples and J is the cost of sampling from $q(\mathbf{z}|\mathbf{x})$ (typically J is of order K^2 after an initial $\mathcal{O}(K^3)$ operation for a matrix decomposition of the covariance). The MC approximation has the usual sampling error of $\mathcal{O}(1/\sqrt{M})$, while the LB has a fixed but small error (empirical comparison in Section 4.5.4). This means that computing the LB is faster than drawing a single MC sample while yielding a full distribution.

Furthermore, the Dirichlet exponential family comes with a range of convenient analytical properties. For example, a point estimate of the posterior predictive distribution is directly given by the Dirichlet's mean,

$$\mathbb{E}[\boldsymbol{\pi}] = \left(\frac{\alpha_1}{\sum_{l=1}^{K} \alpha_l}, \ldots, \frac{\alpha_K}{\sum_{l=1}^{K} \alpha_l} \right)^\top . \tag{4.9}$$

Thus, the MC integration is not necessary to compute the expected value. Additionally, Dirichlets have Dirichlet marginals: If $p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$, then

$$p\left(\pi_1, \ldots, \pi_j, \sum_{k>j} \pi_k\right) = \mathrm{Dir}\left(\alpha_1, \ldots, \alpha_j, \sum_{k>j} \alpha_k\right) . \tag{4.10}$$

Thus, marginal distributions of arbitrary subsets of outputs (including binary marginals) can be computed in closed form.

An additional benefit of the LB for BNNs is that it is more flexible than an MC-integral. Concretely, the LB allows us to access quantities that are non-trivial to compute with the MC-integral. If we let $p(\boldsymbol{\pi})$ be the distribution over $\boldsymbol{\pi} := \mathrm{softmax}(\mathbf{z}) := [e^{z_1}/\sum_l e^{z_l}, \ldots, e^{z_K}/\sum_l e^{z_l}]^\top$, then the MC-integral can be seen as a "point-estimate" of this distribution since it approximates $\mathbb{E}[\boldsymbol{\pi}]$. In contrast, the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$ approximates the *distribution* $p(\boldsymbol{\pi})$. Thus, the LB enables tasks that can be done only with a distribution but not a point estimate. For instance, one could ask "what is the distribution of the softmax output of the first L classes?" when one is dealing with K-class ($L < K$) classification. Since the marginal distribution can be computed analytically with Eq. equation 4.10, the LB provides a convenient yet cheap way of answering this question.

4.4 Limitations of the Laplace Bridge

There are two limitations to applying the LB as presented in Equation A.44. First, the LB assumes that the random variable of the Gaussian sums to zero due to the difference in degrees of freedom between Dirichlet and Gaussian (see Appendix B.3). Thus, we have to add a correction that projects from any arbitrary Gaussian to one that fulfills this constraint. The resulting Gaussian (see Appendix B.1) is

$$\mathcal{N}\left(\mathbf{x}|\mu - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \mu}{\mathbf{1}^\top \Sigma \mathbf{1}}, \Sigma - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \Sigma}{\mathbf{1}^\top \Sigma \mathbf{1}}\right) \tag{4.11}$$

where $\mathbf{1}$ is the one-vector of size K.

Second, the softmax-Dirichlet distribution is asymmetric for extremely sparse cases (see Figure 4.4). These arise in regions where the logistic transform (the 1D special case of the softmax) is nearly flat (as indicated by its derivative in Figure 4.4). Therefore, the LA is suboptimal in these high-variance cases.

This limitation can also be explained by looking at Equation A.44. We observe that Σ contributes linearly to α with $\frac{1}{\Sigma_{kk}}$ while μ contributes exponentially with $\exp(\mu_k)$. For settings where Σ is small, this doesn't have a large effect. However, when Σ_{kk} and μ_k grow the LB results differ

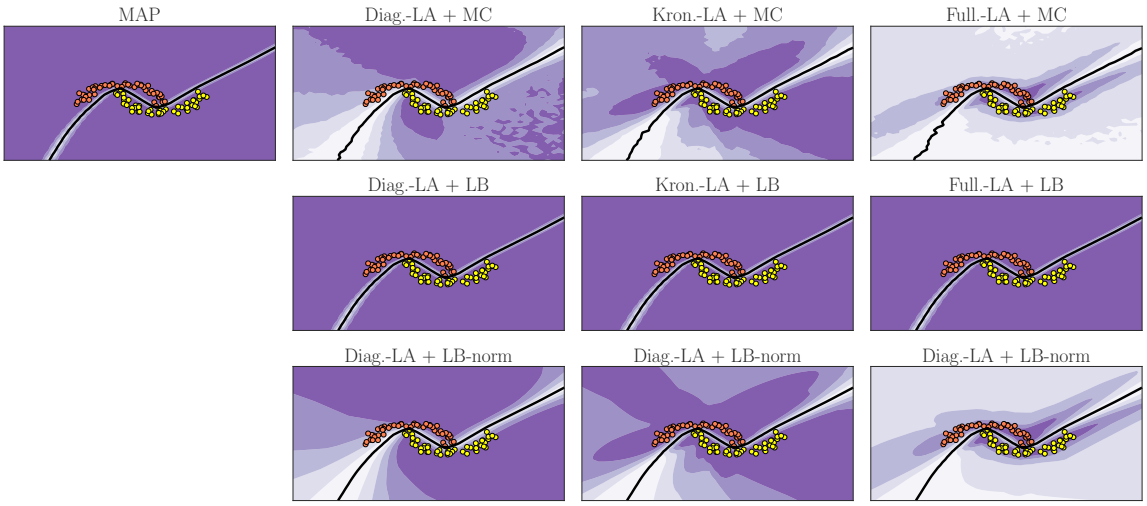


Figure 4.3: **Left column:** vanilla MAP estimate which is overconfident. **Top row:** mean of softmax applied to Gaussian samples. **Middle row:** mean of the vanilla LB. **Bottom row:** mean of the corrected LB. The vanilla LB yields overconfident prediction far from the data. Our proposed correction fixes this issue, making the LB's approximation close to MC.

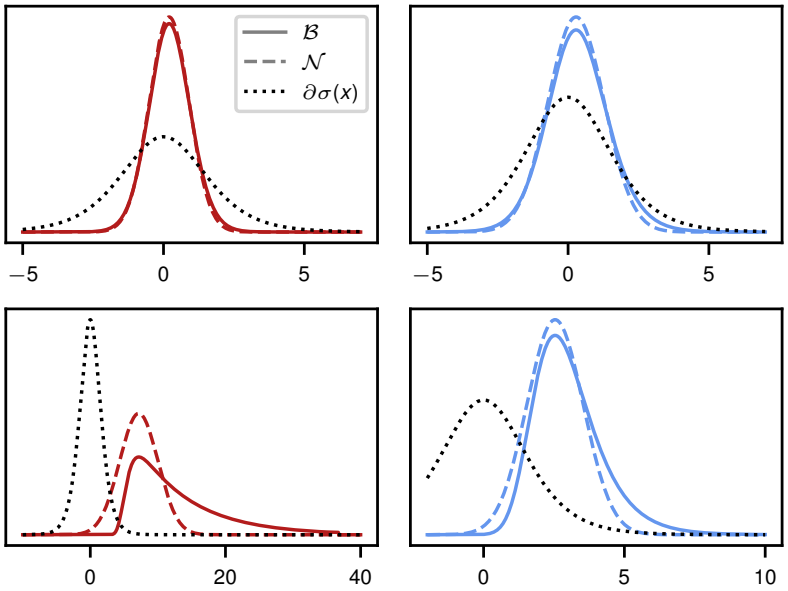


Figure 4.4: In most scenarios (upper row) the LB provides a good fit. However, in some high-variance scenarios (lower row) the softmax-Dirichlet becomes asymmetric and thus the Gaussian is a suboptimal fit. We propose a correction (right column) that projects the Gaussian into a lower-variance region before applying the LB. This can be understood as "pulling back" the Dirichlet to the dynamic of the logistic function (indicated here by its derivative $\partial\sigma$) and thus yields a better approximation.

from softmax Gaussian samples. In the LB, the resulting α is dominated by the mean and the linear influence of the variance cannot correct sufficiently. For MC sampling, on the other hand, the result is mostly determined by the large variance and then amplified through the softmax. Our proposed normalization to the LB reduces this effect (see Figure 4.3).

In BNNs, we often encounter such cases, especially far away from the data (see Figure 4.3 top). Therefore, we propose an additional correction

for practical purposes:

$$c = v_{\text{mean}}(\Sigma) \cdot \frac{1}{\sqrt{K/2}} \tag{4.12}$$

$$\mu' = \frac{\mu}{\sqrt{c}} \tag{4.13}$$

$$\Sigma' = \frac{\Sigma}{c} \tag{4.14}$$

where $v_{\text{mean}}(\Sigma)$ denotes the mean variance of Σ, $v_{\text{mean}}(\Sigma) = \sum_i \Sigma_{ii}$. The factor of $\frac{1}{\sqrt{K/2}}$ is added because we found that higher dimensionalities require less correction. Since our correction is just a rescaling, the zero-sum constrained is still fulfilled. This normalization that can be understood as "pulling back" the distribution into a space where it is symmetric has higher approximation quality. This correction is applied after the zero-sum constraint correction. We want to point out that our correction is motivated by experimentation and the theoretical insights detailed above. There is no theoretical derivation from first principles for the correction. We provide additional explanations and figures in Appendix B.1.

Remark 4.2 (Notes on the correction) While the correction term fixes some problems of the approximation, it doesn't solve all of them. Also, the fact that we need to add a correction term to an approximation makes the LB overall less appealing as a method.

Throughout the paper, we will call this normalizing correction *LB-norm* and explicitly state when we use it. Otherwise, we will use the vanilla version with zero-sum correction.

4.5 Experiments

Table 4.1: OOD detection results: In all scenarios, the Laplace Bridge (LB) or its normalized version yield comparable results to the MC estimate while being much faster. For MC experiments, we draw 100 samples.

		Diag.-LA + MC		Diag.-LA + LB		Diag.-LA + LB-norm		Kron.-LA + MC		Kron.-LA + LB		Kron.-LA + LB-norm	
Train	**Test**	**ECE ↓**	**AUROC ↑**	**ECE ↓**	**AUROC ↑**	**ECE ↓**	**AUROC ↑**	**ECE↓**	**AUROC ↑**	**ECE ↓**	**AUROC ↑**	**ECE ↓**	**AUROC ↑**
MNIST	FMNIST	**0.464**	0.975	0.478	**0.981**	0.498	0.951	0.390	0.987	0.553	0.977	**0.364**	**0.990**
MNIST	notMNIST	0.396	**0.965**	0.600	0.930	**0.360**	0.955	0.366	0.974	0.634	0.912	**0.294**	**0.986**
MNIST	KMNIST	0.429	**0.974**	0.617	0.949	**0.391**	0.970	0.374	0.985	0.619	0.956	**0.328**	**0.991**
CIFAR10	CIFAR100	0.379	**0.887**	0.691	0.859	**0.220**	0.883	0.577	**0.878**	0.670	0.855	**0.558**	0.866
CIFAR10	SVHN	0.309	**0.948**	0.652	0.928	**0.155**	**0.948**	0.447	0.955	0.635	0.924	**0.327**	**0.965**
SVHN	CIFAR100	**0.615**	0.957	0.667	**0.962**	0.679	0.944	0.583	**0.959**	0.659	0.962	**0.575**	0.953
SVHN	CIFAR10	**0.600**	0.958	0.659	**0.960**	0.662	0.947	0.567	**0.960**	0.651	0.959	**0.556**	0.955
CIFAR100	CIFAR10	0.474	0.788	**0.239**	**0.791**	0.834	0.757	0.479	0.787	**0.202**	**0.790**	0.855	0.749
CIFAR100	SVHN	0.470	0.795	**0.207**	**0.815**	0.842	0.748	0.469	0.798	**0.183**	**0.807**	0.849	0.761

Table 4.2: Comparison of the extended probit approximation with the normalized version of the LB norm. While the probit approximation performs well on in-dist problems, the LB norm is better on out-of-distribution tasks.

		Diag Probit					Diag LB norm				
Train	**Test**	**MMC ↓**	**AUROC ↑**	**NLL ↓**	**ECE ↓**	**Brier ↓**	**MMC ↓**	**AUROC ↑**	**NLL ↓**	**ECE ↓**	**Brier ↓**
MNIST	MNIST	**0.967**	-	0.050	0.024	0.002	0.944	-	0.078	0.045	0.003
MNIST	FMNIST	0.597	**0.971**	3.827	0.523	0.128	**0.589**	0.951	**3.538**	**0.498**	**0.124**
MNIST	notMNIST	0.616	**0.958**	3.839	0.488	0.123	**0.492**	0.955	**3.070**	**0.360**	**0.111**
MNIST	KMNIST	0.580	0.969	4.276	0.489	0.126	**0.484**	**0.970**	**3.288**	0.391	**0.115**
CIFAR10	CIFAR10	**0.869**	-	0.237	0.083	0.009	0.517	-	0.727	0.433	0.029
CIFAR10	CIFAR100	0.589	0.882	3.334	0.485	0.123	**0.319**	**0.883**	**2.590**	**0.220**	**0.099**
CIFAR10	SVHN	0.510	0.946	3.097	0.394	0.114	**0.273**	**0.948**	**2.457**	**0.155**	**0.094**

We conduct multiple experiments. Firstly, we compare the LB to the MC-integral on a 2D toy example (Section 4.5.1). Secondly, we apply

Table 4.3: Comparison of last-layer vs. full-layer Laplace approximation. Last-layer results are in the upper half and full-layer results are in the bottom half. We find that, as expected, full-layer results are slightly better than for the last-layer approximation.

Train	Test	Diag.-LA + MC		Diag.-LA + LB		Diag.-LA + LB-norm		Kron.-LA + MC		Kron.-LA + LB		Kron.-LA + LB-norm	
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
MNIST	FMNIST	0.464	0.975	0.478	0.981	0.498	0.951	0.390	0.987	0.553	0.977	0.364	0.990
MNIST	notMNIST	0.396	0.965	0.600	0.930	0.360	0.955	0.366	0.974	0.634	0.912	0.294	0.986
MNIST	KMNIST	0.429	0.974	0.617	0.949	0.391	0.970	0.374	0.985	0.619	0.956	0.328	0.991
MNIST	FMNIST	0.317	0.980	0.322	0.990	0.123	0.986	0.288	0.985	0.528	0.980	0.135	0.991
MNIST	notMNIST	0.280	0.960	0.566	0.924	0.126	0.952	0.282	0.958	0.629	0.915	0.171	0.973
MNIST	KMNIST	0.309	0.976	0.557	0.955	0.112	0.972	0.279	0.981	0.615	0.958	0.152	0.986

Table 4.4: Comparison of Prior Networks with the normalized version of the LB norm. PNs consistently outperform the LB. For discussion see main text.

Train	Test	Prior Network					Diag LB norm				
		MMC ↓	AUROC ↑	ECE ↓	NLL ↓	Brier ↓	MMC ↓	AUROC ↑	ECE ↓	NLL ↓	Brier ↓
MNIST	MNIST	0.802	-	0.184	0.246	0.008	**0.944**	-	**0.045**	**0.078**	**0.003**
MNIST	FMNIST	**0.273**	**0.995**	**0.212**	**2.659**	**0.098**	0.589	0.951	0.498	3.538	0.124
MNIST	notMNIST	**0.447**	0.938	**0.314**	**2.962**	**0.105**	0.492	**0.955**	0.360	3.070	0.111
MNIST	KMNIST	**0.372**	**0.976**	**0.261**	**3.142**	**0.104**	0.484	0.970	0.391	3.288	0.115

the same comparison to out-of-distribution (OOD) detection in many settings (Section 4.5.2). Thirdly, we compare the commonly used probit approximation to the LB in Section 4.5.3 Fourthly, we compare their computational cost and contextualize the speed-up for the prediction process in Section 4.5.4. Finally, in Section 4.5.5, we present analysis on ImageNet (Russakovsky et al., 2014) to demonstrate the scalability of the LB and the advantage of having a full Dirichlet distribution over softmax outputs. We extended Laplace torch (Daxberger et al., 2021) for the experiments. Code can be found in the accompanying GitHub repository.*

For all experiments, a last-layer Laplace approximation is applied. This scheme has been successfully used by Kristiadi et al. (2020) and Snoek et al. (2015). We use diagonal and Kronecker-factorized (KFAC)(Ritter et al., 2018; Martens et al., 2015) approximations of the Hessian, since inverting the exact Hessian is too costly. A detailed mathematical explanation and setup of the experiments can be found in Appendix B.4. While the LB could also be applied to different approximations of a Gaussian posterior predictive such as Variational Inference (Graves, 2011; Blundell et al., 2015), we used a Laplace approximation in our experiments to construct such an approximation. This is for two reasons: (i) it is one of the fastest ways to get a Gaussian posterior predictive and (ii) it can be applied to pre-trained networks which is especially useful for large problems such as ImageNet. Nevertheless, we want to emphasize again that the LB can be applied to any Gaussian over the outputs independent of the way it was generated.

4.5.1 2D Toy example

We train a simple ReLU network on the 2D half-moon problems from scikit-learn (Pedregosa et al., 2011). As can be seen in Figure 4.3 the MAP estimate and vanilla LB are overconfident for the reasons discussed in Section 4.4 but the normalized version yields a near-perfect fit.

* https://github.com/mariushobbhahn/LB_for_BNNs_official

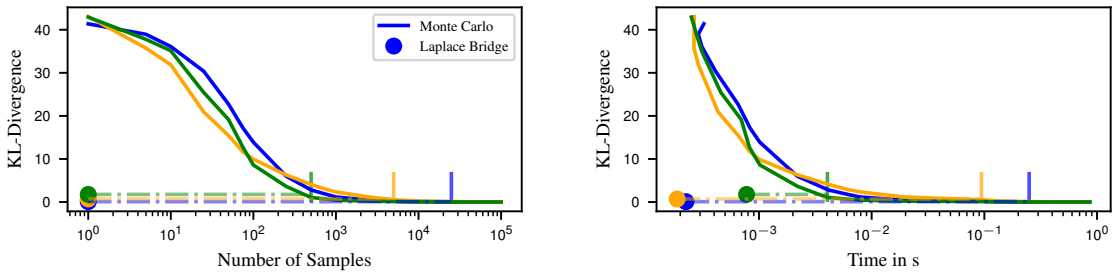


Figure 4.5: KL-divergence plotted against the number of samples (left) and wall-clock time (right). The Monte Carlo density estimation becomes as good as the LB after around 750 to 10k samples and takes at least 100 times longer. The three lines (blue, yellow, green) represent three different sets of parameters. The short vertical bars indicate where the KL divergence of the samples overtake that of the LB.

Table 4.5: Contextualization of the timings for the entire predictive process of a ResNet-18 on CIFAR-10. We see that with 1000 samples the forward pass only uses 6% of the time whereas the sampling uses 94%. In contrast the split for the LB is 96% and 4% respectively. We conclude that the LB provides a significant speed-up of the process as a whole.

# samples in brackets	Forward pass	+MC(1000)	+MC(100)	+MC(10)	+Laplace Bridge
Time in seconds	0.300 ± 0.003	4.712 ± 0.063	0.488 ± 0.009	0.059 ± 0.001	0.013 ± 0.000
Fraction of overall time	0.06/0.38/0.83/0.96	0.94	0.62	0.17	0.04

4.5.2 OOD detection

We compare the performance of the LB to the MC-integral (Diagonal and KFAC) on a standard OOD detection benchmark suite, to test whether the LB gives similar results to the MC sampling methods. Following prior literature, we use the standard expected calibration error (ECE) and area under the ROC-curve (AUROC) metrics (Hendrycks et al., 2016).

For the exact setup, we refer the reader to Appendix B.4. We use the mean of the Dirichlet to obtain a comparable approximation to the MC-integral. The results are presented in Table 4.1.

We find that the results of the LB or its normalized version are comparable throughout the entire benchmark suite. Since the LB is much faster it can be a good replacement for MC in time-sensitive applications.

Furthermore, we compare the LB to prior networks (PNs) in Table 4.4 since PNs also yield a Dirichlet distribution as an output on classification tasks. We find that PNs outperform the LB in most cases. However, we don't think this is a major problem since they have different aims and use cases. The LB creates a Dirichlet distribution on top of an already existing Gaussian model while PNs describe a training procedure and have to be trained from scratch. Thus, the primary comparison for the LB should be against sampling and other integral approximations like in Table 4.2.

Lastly, we compare the LB for a full-layer vs. last-layer Laplace approximation of the network in Table 4.3. We find that, as expected, the full-layer setting yield slightly better results. However, since the primary advantage of the LB is its speed, we think the natural fit for it is a last-layer approximation.

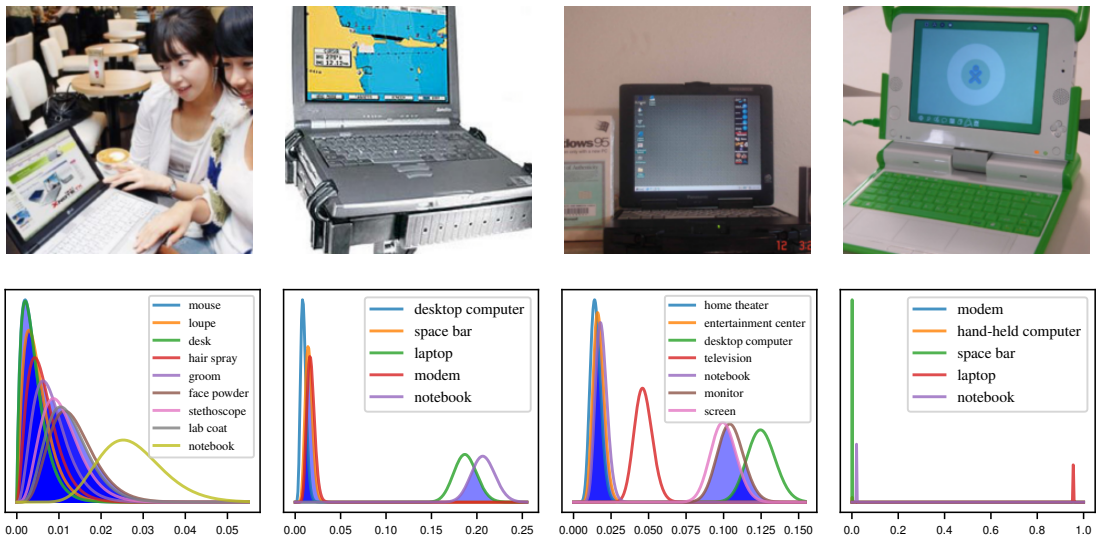


Figure 4.6: **Upper row:** images from the "laptop" class of ImageNet. **Bottom row:** Beta marginals of the top-k predictions for the respective image. In the first column, the overlap between the marginal of all classes is large, signifying high uncertainty, i.e. the prediction is "do not know". In the second column, "notebook" and "laptop" have confident, yet overlapping marginal densities and therefore yield a top-2 prediction: "either notebook or laptop". In the third column "desktop computer", "screen" and "monitor" have overlapping marginal densities, yielding a top-3 estimate. The last case shows a top-1 estimate: the network is confident that "laptop" is the only correct label.

4.5.3 Comparison to the probit approximation

The multi-class probit approximation (Gibbs, 1997; Lu et al., 2020) is a commonly used approximation for the softmax-Gaussian integral. We compare it to the diagonal normalized LB in Table 4.2. We find that the LB norm outperforms the probit approximation in most OOD tasks. When we use a KFAC approximation of the Hessian, this trend still holds (see Table B.1 in Appendix B.4).

4.5.4 Time comparison

We compare the computational cost of the density-estimated p_{sample} distribution via sampling and the Dirichlet obtained from the LB p_{LB} for approximating the true p_{true} over MC-sampling. Different numbers of samples are drawn from the Gaussian, the softmax is applied and the KL-divergence between the histogram of the samples with the true distribution is computed. We use KL-divergences $D_{\text{KL}}(p_{\text{true}} \| p_{\text{sample}})$ and $D_{\text{KL}}(p_{\text{true}} \| p_{\text{LB}})$, respectively, to measure similarity between approximations and ground truth while the number of samples for p_{sample} is increased exponentially. The true distribution p_{true} is constructed via MC with 100k samples. The experiment is conducted for three different Gaussian distributions over $\mathbb{R}^3$. Since the softmax applied to a Gaussian does not have an analytic form, the algebraic calculation of the approximation error is not possible and an empirical evaluation via sampling is the best option. The fact that there is no analytic solution is part of the justification for using the LB in the first place.

Figure 4.5 suggests that the number of samples required such that the distribution p_{sample} approximates the true distribution p_{true} as good as the Dirichlet distribution obtained via the LB is large, i.e. somewhere

between 750 and 10k. This translates to a wall-clock time advantage of at least a factor of 100 before sampling becomes competitive in quality with the LB.

To further demonstrate the low compute cost of the LB, we timed different parts of the process for our setup. On our hardware and setup, training a ResNet-18 on CIFAR10 over 130 epochs takes 71 minutes and 30 seconds. Computing a Hessian for the network from the training data can be done with BackPACK (Dangel et al., 2020) at the cost of one backward pass over the training data or around 29 seconds. This one additional backward pass is the only change to the training procedure compared to conventional training. Since the LB only applies to the last step of the prediction pipeline, it is important to compare it to a forward pass through the rest of the network. Re-using the ResNet-18 and CIFAR10 setup we measure the time in seconds for a forward pass, for the application of the LB, and for the sampling procedure with 10, 100, and 1000 samples. The resulting sum total time for the entire test set is given in Table 4.5. We find that sampling takes up between 94% (for 1000 samples) and 17% (for 10 samples) of the entire prediction while the LB is only 4%. Thus, the acceleration through the LB is a significant improvement for the prediction process as a whole, not only for a part of the pipeline.

4.5.5 Uncertainty-aware output ranking on ImageNet

Due to the cost of sampling-based inference, classification on large datasets with many classes, like ImageNet, is rarely done in a Bayesian fashion. Instead, models for such tasks are often compared along a top-k metric (e.g. $k = 5$).

Although widely accepted, this metric has some pathologies: Depending on how close the point predictions are *relative to their uncertainty*, the total number of likely class labels should be allowed to vary from case to case. Figure 4.6 shows examples: In some cases (panel 2) the classifier is quite confident that the image in question belongs to one out of only two classes and all others are highly unlikely. In others (e.g. panel 1), a larger set of hypotheses are all nearly equally probable.

The Laplace Bridge, in conjunction with the last-layer Laplace approximations, can be used to address this issue. To this end, the analytic properties of its Dirichlet prediction are particularly useful: Recall that the marginal distribution $p(\pi_i, \sum_{j \neq i} \pi_j)$ over each component of a Dirichlet relative to all other components is $\text{Beta}(\alpha_i, \sum_{j \neq i} \alpha_j)$.

We leverage this property to propose a simple *uncertainty-aware top-k* decision rule inspired by statistical tests. Instead of keeping k fixed, it uses the model's confidence to adapt k (pseudo-code in Algorithm 4).

We begin by sorting the class predictions in order of their expected probability α_i. Then we compute the Beta marginal of the most likely class. Now, we compute the overlap of the next marginal and add that class to the list iff the overlap is more than some threshold (e.g. 0.05). Continuing in this fashion, the algorithm terminates with a finite value $k \leq K$ of "non-separated" top classes.

The intuition behind this rule is that, if any Beta density overlaps with the most likely one more than the threshold of, say, 5%, the classifier

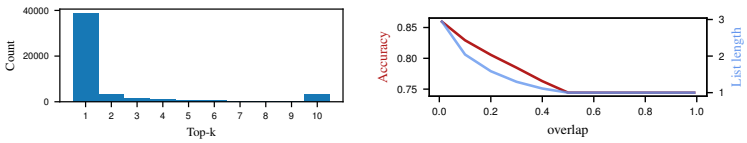


Figure 4.7: A histogram of ImageNet predictions' length using the proposed uncertainty-aware top-k. Results with more than 10 proposed classes have been put into the 10-bin for visibility.

cannot confidently predict one class over the other. Thus, all classes sufficiently overlapping with the top contender should be returned as the top estimates.

We evaluate this decision rule on the test set of ImageNet. The overlap is calculated through the inverse CDF[†] of the respective Beta marginals. The original top-1 accuracy of DenseNet on ImageNet is 0.744. In contrast, the uncertainty-aware top-k method yields accuracies of over 0.85 while average list lengths stay below 3 (see Figure 4.7). Furthermore, we find that most of the predictions given by the uncertainty-aware metric still yielded a top-1 prediction. This means that using uncertainty does not imply adding meaningless classes to the prediction. Furthermore, there are non-negligibly many cases where k equals to 2, 3, or 10 (all values larger than 10 are in the 10 bin).

Thus, using the uncertainty-aware prediction rule above, the classifier can use its uncertainty to adaptively return a longer or shorter list of predictions. This not only allows it to improve accuracy over a hard top-1 threshold. Arguably, the ability to vary the size of the predicted set of classes is a practically useful functionality in itself. As Figure 4.6 shows anecdotally, some of the labels (like "notebook" and "laptop") are semantically so similar to each other that it would seem only natural for the classifier to use them synonymously.

Algorithm 4 Uncertainty-aware top-k

Require: A Dirichlet parameter $\boldsymbol{\alpha} \in \mathbb{R}^K$ obtained by applying the LB to the Gaussian over the logit of an input, a percentile threshold T e.g. 0.05, a function class_of that returns the underlying class of a sorted index.

$\tilde{\boldsymbol{\alpha}} = \text{sort\_descending}(\boldsymbol{\alpha})$ ▷ start with the highest confidence
$\alpha_0 = \sum_i \alpha_i$
$\mathcal{C} = \{\text{class\_of}(1)\}$ ▷ initialize top-k, must include at least one class
$F_1 = \text{Beta}(\tilde{\alpha}_1, \alpha_0 - \tilde{\alpha}_1)$ ▷ the first marginal CDF
$l_1 = F_1^{-1}(T/2)$ ▷ left $\frac{T}{2}$ percentile of the first marginal **for** $i = 2, \ldots, K$ **do**
    $F_i = \text{Beta}(\tilde{\alpha}_i, \alpha_0 - \tilde{\alpha}_i)$ ▷ the current marginal CDF
    $r_i = F_i^{-1}(1 - T/2)$ ▷ right $\frac{T}{2}$ percentile of the current marginal **if** $r_i > l_1$ **then**
        $\mathcal{C} = \mathcal{C} \cup \{\text{class\_of}(i)\}$ ▷ overlap detected, add the current class **else**
        break ▷ No more overlap, end the algorithm
return $\mathcal{C}$ ▷ return the resulting top-k prediction

[†] Also known as the quantile function or percent point function

4.6 Related Work

In BNNs, analytic approximations of posterior predictive distributions have attracted a great deal of research. In the binary classification case, for example, the probit approximation (Gibbs, 1997; Lu et al., 2020) has been proposed already in the 1990s (Spiegelhalter et al., 1990; D. J. MacKay, 1992). However, while there exist some bounds (Titsias, 2016) and approximations of the expected log-sum-exponent function (Ahmed et al., 2007; Braun et al., 2010), in the multi-class case, obtaining a good analytic approximation of the expected softmax function under a Gaussian measure is an open problem. Our LB can be used to produce a close analytical approximation of this integral. It thus furthers the trend of sampling-free solutions within Bayesian Deep Learning (Wu et al., 2018; Haussmann et al., 2019, etc.). The crucial difference is that, unlike these methods, the LB approximates the full distribution over the softmax outputs of a deep network.

Previous approaches proposed to model the distribution of softmax outputs of a network directly. Similar to the LB, Malinin and Gales (2018), Malinin and Gales (2019), and Sensoy et al. (2018) proposed to use the Dirichlet distribution to model the posterior predictive for non-Bayesian networks. They further proposed novel training techniques in order to directly learn the Dirichlet. Additionally, different work on Distillation (Malinin, Mlodozeniec, et al., 2019; Vadera et al., 2020) takes larger models and distills them into a smaller one. The result of some distillation methods is a Dirichlet similar to the LB. We compare against prior nets in the experiments.

In contrast, the LB tackles the problem of approximating the distribution over the softmax outputs of the ubiquitous Gaussian-approximated BNNs (Graves, 2011; Blundell et al., 2015; Louizos et al., 2016; Sun et al., 2017, etc) without any additional training procedure. Therefore the LB can, for example, be used with pre-trained weights on large datasets while prior networks and distillation usually require training from scratch.

4.7 Conclusion

The Laplace Bridge for BNNs is a fast approximation scheme for Bayesian Deep Learning. For any Gaussian approximation to the weight-space posterior of an NN, and an input, the Laplace Bridge analytically maps the marginal Gaussian prediction on the logits onto a Dirichlet distribution over the softmax vectors. This map is linear in the number of classes and thus much faster than ML sampling.

In our experiments we found that the LB empirically preserves predictive uncertainty and is thus a low-cost alternative to MC sampling. Especially when combined with a low-cost, last-layer Gaussian approximation it is useful for real-time applications that require uncertainty. Furthermore, the LB can easily be scaled to larger networks and datasets like ResNet and ImageNet.

However, the vanilla LB has some limitations, i.e. it doesn't perfectly map points from one distribution to another. We proposed a simple correction

that outperforms alternative softmax-integral approximations such as the commonly used multi-class probit.

4.8 Practical Advice & Honest Thoughts

The main advantages of the LB for BNNs are a) that it is extremely fast, b) that it can scale to larger networks and datasets, and c) that it is very easy to apply (it should require less than 10 lines of code in most circumstances).

However, it also has some disadvantages. Most importantly, the LB is an imperfect approximation and, therefore, yields an imperfect prediction. Specifically, as shown in this chapter, the approximation error can be arbitrarily large for certain niche cases and the correction also makes the approximation more conceptually unclean in some sense. Therefore, the only practical use case of the LB for BNNs is when speed is of extreme importance. Otherwise, standard MC schemes or ensembles are preferable. We expect such situations to be very rare and thus think that the LB is almost never the best choice for most practitioners.

I recommend using other methods, such as those mentioned in the related work section or their successors, to obtain high-quality uncertainty approximations of the outputs.

Probabilistic Inference in Directed and Undirected Heterogeneous and Attributed Multilayer

5.

Remark 5.1 The contents of this chapter are primarily based on:

Martina Contisciani*, Marius Hobbhahn*, Eleanor A. Power, Philipp Hennig, and Caterina De Bacco. "*Flexible inference in heterogeneous and attributed multilayer networks*". https://arxiv.org/abs/2405.20918. 2024.

	Idea	Analysis	Exp.	Code	Writing
Martina Contisciani	20%	55%	60%	50%	65%
Marius Hobbhahn	70%	20%	25%	50%	10%
Eleanor A. Power	0%	10%	5%	0%	5%
Philipp Hennig	5%	0%	0%	0%	0%
Caterina De Bacco	5%	15%	10%	0%	20%

Remark 5.2 (Contributions) I started this project in 2021 and came up with the framework and set up a fast code library for all networks. Martina focused primarily on the analysis and experiments. My contributions were finished in late 2022. Martina and Eleanor worked part-time on the paper throughout 2023 and 2024.

Abstract

Networked datasets are often enriched by different types of information about individual nodes or edges. However, most existing methods for analyzing such datasets struggle to handle the complexity of heterogeneous data, often requiring substantial model-specific analysis. In this paper, we develop a probabilistic generative model to perform inference in multilayer networks with arbitrary types of information. Our approach employs a Bayesian framework combined with the Laplace matching technique to ease interpretation of inferred parameters. Furthermore, the algorithmic implementation relies on automatic differentiation, avoiding the need for explicit derivations. This makes our model scalable and flexible to adapt to any combination of input data. We demonstrate the effectiveness of our method in detecting overlapping community structures and performing various prediction tasks on heterogeneous multilayer data, where nodes and edges have different types of attributes. Additionally, we showcase its ability to unveil a variety of patterns in a social support network among villagers in rural India by effectively utilizing all input information in a meaningful way.

Motivation: Network Analysis

This section offers an intuitive introduction to provide context for readers less acquainted with the field rather than a formal presentation.

Network analysis is a domain of research that investigates the structure, dynamics, and behavior of intricate networks. These networks can represent diverse systems, including social networks, biological networks, transportation networks, and communication networks.

The approach we describe in the following sections is applicable to various types of networks. However, we demonstrate its implementation using the specific example of community detection in social networks.

A social scientist studying the social structure of a village might survey its N inhabitants using L questionnaires. These surveys would capture two types of information: relational data and individual attributes. Relational questions, such as friendships or financial obligations, would each generate an $N \times N$ adjacency matrix A, where A_{ij} represents individual i's response regarding individual j. Questions about personal characteristics like religion, age, or education would produce $N \times P$ matrices, with P denoting the number of categories for categorical data (e.g., different religions) or $P = 1$ for continuous variables (e.g., net worth). This comprehensive data collection approach enables a multifaceted analysis of the village's social dynamics and individual characteristics.

A researcher seeking to draw general conclusions about the village's social structure would want to aggregate the data in a principled way. In our case, we want to analyze the social communities in the village, i.e. which individuals belong to which community and how these communities interact with each other. Specifically, we're interested in creating an $N \times K$ community matrix U where each row U_i is a vector of probabilities denoting the community membership of individual i. Since community membership can be fuzzy and we want to explicitly incorporate our uncertainty, we use probabilistic inference. Thus, our goal is to create a probabilistic graphical model that finds the community matrix U and other latent parameters of interest that best explain the observed data A and X.

5.1 Introduction

Networks effectively represent real-world data from various fields, including social, biological, and informational systems. In this framework, nodes within the network correspond to individual components of the system, and their interactions are illustrated through network edges (M. Newman, 2018). With the advancement of data collection and representation techniques, networks have evolved to become more versatile and informative. Notably, attributed multilayer networks have emerged as a significant development, allowing the inclusion of additional information related to nodes and edges. This enriches the representation of real-world systems, where nodes naturally have specific characteristics and are connected through different types of interactions. For instance, in social networks, individuals can be described by attributes like age, gender, and

height, while engaging in various types of relationships like friendship, co-working, and kinship.

The analysis of attributed multilayer networks has primarily been tackled using techniques like matrix factorization (Liu et al., 2020; Xu et al., 2023), network embedding (Pei et al., 2018; Cen et al., 2019), and deep learning (Cao et al., 2018; Park et al., 2020; Han et al., 2022; Martirano et al., 2022). However, in this work, we focus on a less investigated methodology that involves probabilistic generative models (Goldenberg et al., 2010). Unlike the aforementioned approaches, these methods provide a principled and flexible framework to incorporate prior knowledge and specific assumptions, while also accounting for the inherent uncertainty present in real-world data (Peel et al., 2022). Moreover, they can be applied to perform inference on networks, including tasks such as predictions or the detection of statistically meaningful network structures. The latter task is commonly referred to as community detection problem, and is relevant in many applications (Fortunato, 2010).

Our goal is to develop a probabilistic generative model that can flexibly adapt to any attributed multilayer network, regardless of the type of information encoded in the data. Acting as a "black box", our method can enable practitioners to automatically analyze various datasets, without the need to deal with mathematical details or new derivations. This approach aligns with some practices in the machine learning community, where principled black box methodologies have been introduced to simplify the inference of latent variables in arbitrary models (Ranganath et al., 2014; Tran et al., 2016). In this context, more specific probabilistic methods have been developed to address the challenge of performing inference on heterogeneous data (Valera et al., 2020; Nazabal et al., 2020). However, these techniques are tailored for tabular data and do not provide a general solution to adapt them to network data.

Probabilistic generative models specifically designed for attributed networks aim to combine node attributes effectively with network interactions. Existing methods (Tallberg, 2004; Yang et al., 2013; Hric et al., 2016; M. E. Newman and Clauset, 2016; White et al., 2016; Stanley et al., 2019; Fajardo-Fontiveros et al., 2022; Contisciani et al., 2020) have highlighted the importance of incorporating extra information to enhance network inference, resulting in improved prediction performance and deeper insights on the interplay between edge structure and node metadata. However, these models mainly focus on single-layer networks, assume the same generative process for all interactions, and consider only one type of attribute – typically categorical. These limitations restrict their capability to represent complex scenarios characterized by heterogeneous information. As a consequence, addressing the challenge of effectively incorporating various sources of information and evaluating their collective impact on downstream network inference tasks remains an open issue.

We address this gap by introducing PIHAM, a generative model explicitly designed to perform Probabilistic Inference in directed and undirected Heterogeneous and Attributed Multilayer networks. Our approach differs from previous studies in that PIHAM flexibly adapts to any combination of input data, while standard probabilistic methods rely on model-specific analytic derivations that highly depend on the data types given in input. This can dramatically hinder the flexibility of a model, as any small

change in the data, e.g., adding a new node attribute or a new type of interaction, usually requires new derivations. As a result, the vast majority of these models work only with one type of edge weight for all layers, and one type of attribute. In contrast, PIHAM takes in input any number of layers and attributes, regardless of their data types.

At its core, PIHAM assumes the existence of a mixed-membership community structure that drives the generation of both interactions and node attributes. In addition, the inference of the parameters is performed within a Bayesian framework, where both prior and posterior distributions are modeled with Gaussian distributions. Importantly, PIHAM employs the Laplace matching technique (Hobbhahn and Hennig, 2021) and conveniently maps the posterior distributions to various desired domains, to ease interpretation. For instance, to provide a probabilistic interpretation of the inferred communities, our method properly maps the parameters of a Gaussian distribution into those of a Dirichlet distribution. The latter operates within a positive domain and enforces normalization on a simplex, making it a valuable tool for this purpose. Notably, the inference process is flexible and scalable, relying on automatic differentiation and avoiding the need for explicit derivations. As a result, PIHAM can be considered a "black box" method, as practitioners only need to select the desired probabilistic model and a set of variable transformation functions, while the remaining calculations and inference are performed automatically. This versatility enables our model to be flexibly applied to new modeling scenarios.

We apply our method to a diverse range of synthetic and real-world data, showcasing how PIHAM effectively leverages the heterogeneous information contained in the data to enhance prediction performance and provide richer interpretations of the inferred results.

5.2 Methods

We introduce PIHAM, a versatile and scalable probabilistic generative model designed to perform inference in attributed multilayer networks. Our method flexibly adapts to any combination of input data, regardless of their data types. For simplicity, in what follows, we present examples with Bernoulli, Poisson, Gaussian, and categorical distributions, which collectively cover the majority of real-world examples. Nevertheless, our model can be easily extended to include new distributions, as well as applied to single-layer networks with or without attributes.

General framework

Attributed multilayer networks provide an efficient representation of complex systems in which the individual components have diverse attributes (often referred to as covariates or metadata) and are involved in multiple forms of interactions. Mathematically, these interactions are depicted by an adjacency tensor $\boldsymbol{A}$ of dimension $L \times N \times N$, where N is the number of nodes common across all L layers. Each entry A_{ij}^{ℓ} in this tensor denotes the weight of a directed interaction of type ℓ from node i to node j. Notably, different layers can incorporate interactions

of diverse data types, depending on the nature of the underlying relationship. For instance, in social systems, one layer might represent binary relationships like friendships, another could describe nonnegative discrete interactions such as call counts, and a third might contain continuous real-valued measurements such as geographical distances between locations. In this scenario, the adjacency tensor would be represented as $\boldsymbol{A} = \{\boldsymbol{A^1} \in \{0,1\}^{N\times N}, \boldsymbol{A^2} \in \mathbb{N}_0^{N\times N}, \boldsymbol{A^3} \in \mathbb{R}_+^{N\times N}\}$. Node metadata describes additional information about the nodes. They are stored in a design matrix $\boldsymbol{X}$ with dimensions $N \times P$, where P is the total number of attributes and the entries X_{ix} represent the value of an attribute x for a node i. Similar to network interactions, different attributes can have different data types. An example of input data is given in Figure 5.1A.

PIHAM describes the structure of attributed multilayer networks, represented by $\boldsymbol{A}$ and $\boldsymbol{X}$, through a set of latent variables $\boldsymbol{\Theta}$. The goal is to infer $\boldsymbol{\Theta}$ from the input data. In particular, we want to estimate posterior distributions, as done in a probabilistic framework. These can be approximated as:

$$\begin{aligned} P(\boldsymbol{\Theta} \mid \boldsymbol{A}, \boldsymbol{X}) &\propto P(\boldsymbol{A}, \boldsymbol{X} \mid \boldsymbol{\Theta})\, P(\boldsymbol{\Theta}) \\ &= P(\boldsymbol{A} \mid \boldsymbol{\Theta})\, P(\boldsymbol{X} \mid \boldsymbol{\Theta})\, P(\boldsymbol{\Theta})\,. \end{aligned} \tag{5.1}$$

In this general setting, the proportionality is due to the omission of an intractable normalization term that does not depend on the parameters. The term $P(\boldsymbol{A}, \boldsymbol{X} \mid \boldsymbol{\Theta}) = P(\boldsymbol{A} \mid \boldsymbol{\Theta})\, P(\boldsymbol{X} \mid \boldsymbol{\Theta})$ represents the likelihood of the data, where we assume that $\boldsymbol{A}$ and $\boldsymbol{X}$ are conditionally independent given the parameters. This assumption allows to model separately the network structure and the node metadata. The term $P(\boldsymbol{\Theta})$ denotes the prior distributions of the latent variables, which we assume to be independent and Gaussian distributed, resulting in $P(\boldsymbol{\Theta}) = \prod_{\theta\in\boldsymbol{\Theta}} \mathcal{N}(\theta; \boldsymbol{\mu}^{\theta}, \boldsymbol{\Sigma}^{\theta})$. Importantly, we also make the assumption that the posterior distributions of the parameters can be approximated with Gaussian distributions, for which we have to estimate mean and covariance matrices: $P(\boldsymbol{\Theta} \mid \boldsymbol{A}, \boldsymbol{X}) \approx \prod_{\theta\in\boldsymbol{\Theta}} \mathcal{N}(\theta; \hat{\boldsymbol{\mu}}^{\theta}, \hat{\boldsymbol{\Sigma}}^{\theta})$.

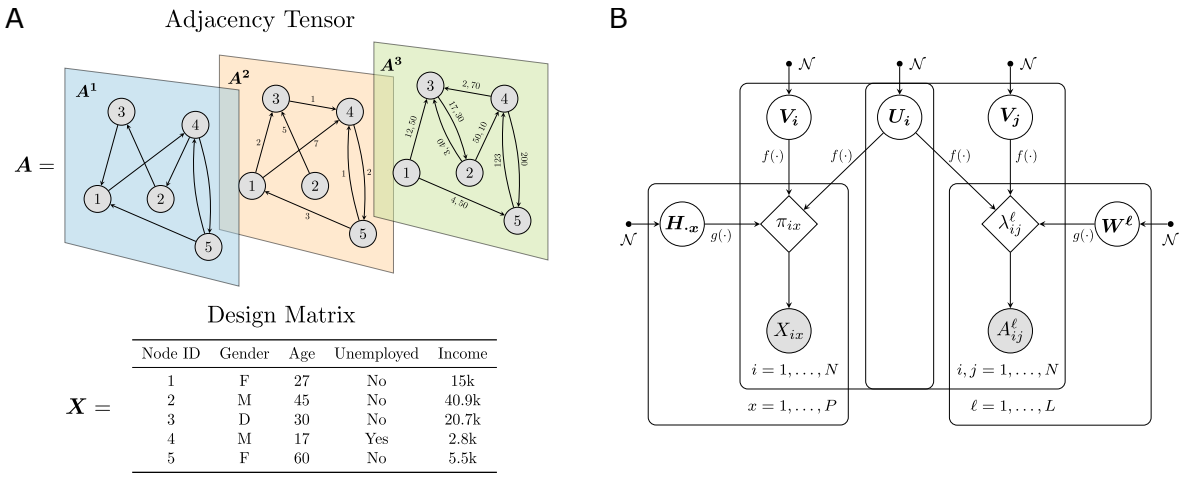
Remark 5.3 (Summary of PIHAM)
Procedure:

1. Define your model, i.e. how the community matrices generate the observations.
2. Define the likelihood terms to match the correct datatype, e.g. Bernoulli for binary data and Poisson for count data.
3. Define all latent variables to be transformations of Gaussians, e.g. probability vectors would be softmax-transformed Gaussians.
4. Use Automatic differentiation to get a Laplace approximation of the latent Gaussian posteriors.
5. Use Laplace matching to transform the latent Gaussians into posteriors in the correct domain.
6. Analyze these latent posteriors.

In the following subsections, we provide additional details on the role of the latent variables in shaping both interactions and node attributes, as well as the methods for inferring their posterior distributions.

Modelling the network structure

The interactions encoded in the adjacency tensor $\boldsymbol{A}$ are assumed to be conditionally independent given the latent variables, resulting in a decomposition of the likelihood across individual entries A_{ij}^{ℓ}. This factorization can be further unpacked by explicitly considering the distributions that describe each layer. For instance, in the scenario with binary, count-based, and continuous interactions, we can express the



Node ID	Gender	Age	Unemployed	Income
1	F	27	No	15k
2	M	45	No	40.9k
3	D	30	No	20.7k
4	M	17	Yes	2.8k
5	F	60	No	5.5k

Figure 5.1: Input data and graphical model representation. (A) The attributed multilayer network is represented by the interactions A^{ℓ}_{ij} and the node attributes X_{ix}. (B) PIHAM describes the observed data through a set of latent variables $\boldsymbol{\Theta} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H})$. $\boldsymbol{U_i}$ and $\mathbf{i}$ respectively depict the out-going and in-coming communities of node i; $\boldsymbol{W^{\ell}}$ is the affinity matrix associated to the layer ℓ and characterizes the edge density between different community pairs in the given layer; $\boldsymbol{H_{\cdot x}}$ is a K-dimensional vector that explains how an attribute x is distributed among the K communities. All latent variables are independent and normally distributed, and $f(\cdot)$ and $g(\cdot)$ are transformation functions to ensure that the expected values λ^{ℓ}_{ij} and π_{ix} belong to the correct parameter space for the various distribution types.

likelihood as follows:

$$
\begin{aligned}
P(\boldsymbol{A} \mid \boldsymbol{\Theta}) &= \prod_{\ell,i,j} P(A^{\ell}_{ij} \mid \boldsymbol{\Theta}) \\
&= \prod_{\ell \in L_B,i,j} \mathrm{Bern}(A^{\ell}_{ij}; \lambda^{\ell}_{ij}(\boldsymbol{\Theta})) \\
&\quad \times \prod_{\ell \in L_P,i,j} \mathrm{Pois}(A^{\ell}_{ij}; \lambda^{\ell}_{ij}(\boldsymbol{\Theta})) \\
&\quad \times \prod_{\ell \in L_G,i,j} \mathcal{N}(A^{\ell}_{ij}; \lambda^{\ell}_{ij}(\boldsymbol{\Theta}), \sigma^2)\,,
\end{aligned}
\tag{5.2}
$$

where σ^2 is a hyperparameter and L_B, L_P, and L_G are the sets of Bernoulli, Poisson, and Gaussian layers, respectively. We assume that each distribution is fully parametrized through the latent variables $\boldsymbol{\Theta}$ and these explicitly define the expected values λ^{ℓ}_{ij}, regardless of the data type.

Specifically, we adopt a multilayer mixed-membership model (De Bacco et al., 2017), and describe the observed interactions through K overlapping communities shared across all layers. Following this approach, the expected value of each interaction of type ℓ from node i to j can be approximated as:

$$
\lambda^{\ell}_{ij}(\boldsymbol{\Theta}) \approx \sum_{k,q=1}^{K} U_{ik} W^{\ell}_{kq} V_{jq}\,,
\tag{5.3}
$$

where the latent variables U_{ik} and V_{jq} denote the entries of K-dimensional vectors $\boldsymbol{U_i}$ and $\mathbf{i}$, which respectively represent the out-going and in-coming communities of node i. In undirected networks, we set $\boldsymbol{U} = \boldsymbol{V}$. Moreover, each layer ℓ is associated with an affinity matrix $\boldsymbol{W^{\ell}}$ of dimension $K \times K$, which characterizes the edge density between different community pairs in the given layer ℓ. This setup allows having diverse structural patterns in each layer, including arbitrarily mixtures of assortative, disassortative and core-periphery structures.

As a final remark, the approximation in Equation 5.3 arises from a discrepancy between the parameter space of the latent variables and that of the expected values of the distributions. In fact, while all variables are normally distributed, λ_{ij}^{ℓ} has to satisfy different constraints according to the distribution type. For instance, $\lambda_{ij}^{\ell} \in [0,1] \,\forall\, \ell \in L_B$ and $\lambda_{ij}^{\ell} \in (0,\infty) \,\forall\, \ell \in L_P$. For further details, we refer to the section Parameter space and transformations.

Modelling the node metadata

Similarly to the network edges, the node metadata are also considered to be conditionally independent given the latent variables. Therefore, when dealing with data that encompass categorical, count-based, and continuous attributes, the likelihood can be formulated as follows:

$$\begin{aligned} P(\boldsymbol{X} \mid \boldsymbol{\Theta}) &= \prod_{i,x} P(X_{ix} \mid \boldsymbol{\Theta}) \\ &= \prod_{i,x \in C_C} \mathrm{Cat}(X_{ix}; \pi_{ix}(\boldsymbol{\Theta})) \\ &\quad \times \prod_{i,x \in C_P} \mathrm{Pois}(X_{ix}; \pi_{ix}(\boldsymbol{\Theta})) \\ &\quad \times \prod_{i,x \in C_G} \mathcal{N}(X_{ix}; \pi_{ix}(\boldsymbol{\Theta}), \sigma^2)\,, \end{aligned} \tag{5.4}$$

where C_C, C_P, and C_G are the sets of categorical, Poisson, and Gaussian attributes, respectively.

Following previous work (Yang et al., 2013; Fajardo-Fontiveros et al., 2022; Contisciani et al., 2020), we assume that the attributes are also generated from the node community memberships, thereby creating dependencies between node metadata and network interactions. In particular, we approximate the expected value of an attribute x for node i as:

$$\pi_{ix}(\boldsymbol{\Theta}) \approx \frac{1}{2} \sum_{k=1}^{K} (U_{ik} + V_{ik})\, H_{kx}\,, \tag{5.5}$$

where $\boldsymbol{H}$ is a $K \times P$-dimensional community-covariate matrix, explaining how an attribute x is distributed among the K communities. For instance, if we consider income as node metadata and expect communities to group nodes with similar income values, then the column vector $\boldsymbol{H}_{\cdot x}$ describes how income varies across groups. It is important to observe that when the attribute x is categorical, the expression in Equation (5.5) becomes more complex because it must consider the total number of attribute categories Z. We provide additional details in the Supporting Information.

Notice that like λ_{ij}^{ℓ}, π_{ix} also needs to satisfy specific constraints depending on the distribution type. We clarify this in the next subsection.

Parameter space and transformations

A key technical aspect of PIHAM is the use of Gaussian distributions to model priors and posteriors of the latent variables $\boldsymbol{\Theta} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H})$.

Table 5.1: Functions $g(\cdot)$ used in our implementation to transform the latent variables as defined in Equation 5.6 and 5.7.

Distribution	Parameter space	Transformation function
Bernoulli	$[0, 1]$	Logistic
Poisson	$(0, \infty)$	Exponential
Gaussian	$\mathbb{R}$	Identity
Categorical	$p_z \geq 0 \forall z, \sum_z p_z = 1$	Softmax

This choice simplifies the inference by an additional step that ensures the expected values λ_{ij}^{ℓ} and π_{ix} belong to the correct parameter space for the various distribution types. To achieve this, we apply specific transformation functions to the latent variables, and model the expected values as follows:

$$\lambda_{ij}^{\ell}(\boldsymbol{\Theta}) = f(\boldsymbol{U}_i)\, g(\boldsymbol{W}^{\boldsymbol{\ell}})\, f(\mathbf{j}) \tag{5.6}$$

$$\pi_{ix}(\boldsymbol{\Theta}) = \frac{1}{2}\big(f(\boldsymbol{U}_i) + f(\mathbf{i})\big)\, g(\boldsymbol{H}_{\cdot x})\,. \tag{5.7}$$

The functions $f(\cdot)$ and $g(\cdot)$ can take various forms, as long as they adhere to the required constraints. In our implementation, we select $f(\cdot)$ to be the softmax function, which is applied to every row of the community membership matrices. This allows interpretability of the communities, as they result in quantities that are positive and normalized to one, as discussed in the section Parameter interpretation. Meanwhile, the choice of $g(\cdot)$ varies depending on the distribution type, as illustrated in Table 5.1.

One might argue that it would be simpler to employ a single link function for λ_{ij}^{ℓ} and π_{ix}, rather than applying individually transformations to the latent variables, as done in standard statistical approaches (McCullagh, 2019). However, this may not ensure interpretability of the communities, as we do with the softmax $f(\cdot)$. In addition, empirically we discovered that the approach outlined in Equation 5.6 and 5.7 gives more stable results, and it does not result in over- or under-flow numerical errors. Alternatively, another approach considers treating the transformed parameters as random variables and applies the probability transformation rule to compute their posterior distributions (Kucukelbir et al., 2017). While this method is theoretically well-founded, it comes with constraints regarding the choice of the transformation functions, which directly affects the feasibility of the inference process. Conversely, PIHAM offers the flexibility to use any set of transformation functions that respects the parameter space of the distribution types given by the network and covariates.

In Figure 5.1, we illustrate the input data and the graphical model representation of our approach.

Posterior inference

PIHAM aims at estimating the posterior distributions of the latent variables, as outlined in Equation 5.1. More precisely, this equation can be

reformulated as:

$$P(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H} \mid \boldsymbol{A}, \boldsymbol{X}) = P(\boldsymbol{A} \mid \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) \times P(\boldsymbol{X} \mid \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{H}) \times P(\boldsymbol{U})\, P(\boldsymbol{V})\, P(\boldsymbol{W})\, P(\boldsymbol{H})\,. \tag{5.8}$$

In general, this posterior distribution lacks a closed-form analytical solution and requires the use of approximations.

Common methods for inference in attributed networks typically rely on Expectation-Maximization (EM) (Dempster et al., 1977) or Variational Inference (VI) (Blei et al., 2017b) techniques. However, these approaches have limitations, as they require model-specific analytic computations for each new term added to the likelihood. For instance, an EM-based approach involves taking derivatives with respect to a given latent variable and setting them to zero. In a specific class of models where the likelihood and prior distributions are compatible, solving the resulting equation for the variable of interest can yield closed-form updates. Nonetheless, for generic models, there is no guarantee of a closed-form solution. Even when this does exist, slight variations in the input data may require entirely new derivations and updates. Consequently, most of these models are designed to handle only a single type of edge weight and a single type of attribute.

In contrast, our model takes a different approach and flexibly adapts to any combination of input data, regardless of their data types. We begin by assuming that the latent variables are conditionally independent given the data, allowing us to model each posterior distribution separately:

$$P(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H} \mid \boldsymbol{A}, \boldsymbol{X}) = P(\boldsymbol{U} \mid \boldsymbol{A}, \boldsymbol{X})\, P(\boldsymbol{V} \mid \boldsymbol{A}, \boldsymbol{X}) \times P(\boldsymbol{W} \mid \boldsymbol{A}, \boldsymbol{X})\, P(\boldsymbol{H} \mid \boldsymbol{A}, \boldsymbol{X})\,. \tag{5.9}$$

Subsequently, we employ a Laplace Approximation (LA) to approximate each posterior with a Gaussian distribution, resulting in:

$$P(\boldsymbol{\theta} \mid \boldsymbol{A}, \boldsymbol{X}) \approx \mathcal{N}(\boldsymbol{\theta}; \hat{\boldsymbol{\mu}}^{\boldsymbol{\theta}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{\theta}})\,,\ \forall\, \boldsymbol{\theta} \in \boldsymbol{\Theta}\,. \tag{5.10}$$

LA involves a second-order Taylor expansion around the Maximum A Posteriori estimate (MAP) of the right-hand side of Equation 5.8. We compute this estimate using Automatic Differentiation (AD), a gradient-based method that, in our implementation, employs the Adam optimizer to iteratively evaluate derivatives of the log-posterior. The MAP estimate found with AD also constitutes the mean $\hat{\boldsymbol{\mu}}^{\boldsymbol{\theta}}$ of $P(\boldsymbol{\theta} \mid \boldsymbol{A}, \boldsymbol{X})$. To go beyond point estimates and quantify uncertainty, one can further estimate the covariance matrix $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{\theta}}$, which is given by the inverted Hessian around the MAP:

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{\theta}} \approx \left[- \frac{\partial^2 P(\boldsymbol{\theta} \mid \boldsymbol{A}, \boldsymbol{X})}{\partial \boldsymbol{\theta}} (\hat{\boldsymbol{\mu}}^{\boldsymbol{\theta}}) \right]^{-1}. \tag{5.11}$$

Other inference methods can be employed to approximate Gaussian distributions, such as VI. However, in such situations, utilizing AD directly might not be feasible due to the involvement of uncertain expectations in the optimization cost function. On the other hand, LA naturally combines with AD, providing a flexible and efficient inference procedure.

Parameter interpretation

We approximate the posterior distributions of the latent variables using Gaussian distributions, as outlined in Equation 5.10. Consequently, all our estimated parameters belong to the real space. Although this approach is advantageous for developing an efficient and automated inference method, practitioners may desire different variable domains to enhance interpretability. In some instances, achieving this transformation is straightforward, involving the application of the probability transformation rule to obtain a distribution for the transformed variable within the desired constrained support. For example, if we are interested in expressing $\bar{\boldsymbol{U}} := \exp(\hat{\boldsymbol{U}}) \in \mathbb{R}_{>0}^{N\times K}$, we can simply employ the Lognormal$(\bar{\boldsymbol{U}}; \hat{\boldsymbol{\mu}}^{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{U}})$ distribution. Similarly, when seeking $\bar{\boldsymbol{U}} := \text{logistic}(\hat{\boldsymbol{U}}) \in (0,1)^{N\times K}$, we can just compute the Logitnormal$(\bar{\boldsymbol{U}}; \hat{\boldsymbol{\mu}}^{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{U}})$.

However, certain functions lack closed-form transformations. For instance, obtaining a probabilistic interpretation of the mixed-memberships of nodes requires applying the softmax function to each row of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, which is not a bijective function. To address this challenge, our framework employs the Laplace Matching (LM) (Hobbhahn and Hennig, 2021) to approximate the distributions of such transformations. This technique yields a bidirectional, closed-form mapping between the parameters of the Gaussian distribution and those of the approximated transformed distribution. In this scenario, we can derive:

$$\begin{aligned} \bar{\boldsymbol{U}}_{\boldsymbol{i}} := \text{softmax}(\hat{\boldsymbol{U}}_{\boldsymbol{i}}) \,, \bar{U}_{ik} \in [0,1] \text{ and } \sum_{k=1}^{K} \bar{U}_{ik} = 1 \\ \text{with } P(\bar{\boldsymbol{U}}_{\boldsymbol{i}}) = \text{Dir}(\bar{\boldsymbol{U}}_{\boldsymbol{i}}; \hat{\boldsymbol{\alpha}}_{\boldsymbol{i}}^{\boldsymbol{U}}) \,, \end{aligned} \tag{5.12}$$

where $\hat{\boldsymbol{\alpha}}_{\boldsymbol{i}}^{\boldsymbol{U}}$ is a K-dimensional vector obtained with LM, whose entries are described as:

$$\hat{\alpha}_{ik}^{U} = \frac{1}{\hat{\Sigma}_{ikk}^{U}} \left(1 - \frac{2}{K} + \frac{\exp(\hat{\mu}_{ik}^{U})}{K^2} \sum_{l=1}^{K} \exp(\hat{\mu}_{il}^{U})\right). \tag{5.13}$$

This approach is theoretically grounded and enables us to provide closed-form posterior distributions for the latent variables across a diverse range of domains. Consequently, it consistently allows for the estimation of uncertainties and other relevant statistical measures. Nonetheless, PIHAM can also be utilized for the sole purpose of determining point estimates of the latent variables, which are essentially given by the MAP estimates. In such scenarios, it remains feasible to map these point estimates to different supports by applying any desired function, without worrying about the transformation process. Although this approach lacks full posterior distributions, it significantly simplifies the inference process by avoiding the computation of the Hessian. The choice between these two approaches should be guided by the specific application under study.

5.3 Results

We demonstrate our method on both synthetic and real-world datasets, presenting a comprehensive analysis through quantitative and qual-

itative findings. Further explanations about the data generation and pre-processing procedures can be found in Appendix C, which also includes additional results. The settings used to run our experiments and the choice of the hyperparameters are also described in Appendix C. The code implementation of PIHAM is accessible at: https://github.com/mcontisc/PIHAM.

Simulation study

Comparison with existing methods in a homogeneous scenario. We first investigate the behavior of our model in a simpler and common scenario, characterized by attributed multilayer networks with nonnegative discrete weights and one categorical node attribute. This represents the most general case addressed by existing methods, which are specifically designed for homogeneous settings, where there is only one attribute and one data type. For comparison, we use MTCOV (Contisciani et al., 2020), a probabilistic model that assumes overlapping communities as the main mechanism governing both interactions and node attributes. In contrast to PIHAM MTCOV is tailor-made to handle categorical attributes and nonnegative discrete weights. Additionally, it employs an EM algorithm, with closed-form derivations for parameters inference strongly relying on the data type, making MTCOV a bespoke solution compared to the more general framework proposed by PIHAM. The results of this comparison are depicted in Figure C.1 in Appendix C, accompanied by additional details about the data generation and experiment settings. In principle, we expect MTCOV to exhibit better performance in this specific scenario due to its tailored development for such data and also its generative process aligning closely with the mechanism underlying the synthetic data. Nonetheless, despite the generality of our approach, we observe that PIHAM achieves comparable performance to MTCOV in link and attribute prediction, as well as community detection, especially in scenarios involving denser networks. These results collectively show that PIHAM is a valid approach even in less heterogeneous scenarios, as it can compete effectively with bespoke existing methods.

Validation on heterogeneous data. Having demonstrated that PIHAM performs comparably well to existing methods for attributed multilayer networks, we now demonstrate its behavior on more complex data containing heterogeneous information. To the best of our knowledge, this is the first probabilistic generative model designed to handle and perform inference on such data, and as a result, a comparative analysis is currently unavailable. Additionally, due to the absence of alternative benchmarks for data generation, we validate the performance of our method on synthetic data generated using the model introduced in this work.

We analyze attributed multilayer networks with $L = 3$ heterogeneous layers: one with binary interactions, one with nonnegative discrete weights, and one with real values. In addition, each node is associated with three covariates: one categorical with $Z = 4$ categories, one representing nonnegative discrete values, and one involving real values. To generate these networks, we initially draw the latent variables $\boldsymbol{\Theta} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H})$ from Gaussian distributions with specified hyperparameters. Subsequently,

we generate $\boldsymbol{A}$ and $\boldsymbol{X}$ according to the data types, following Equation 5.2 and 5.4. Our analysis spans networks with varying number of nodes $N \in \{100, 200, \ldots, 1000\}$ and diverse number of overlapping communities $K \in \{3, 4, 5\}$. Additional details on the generation process can be found in Appendix C.

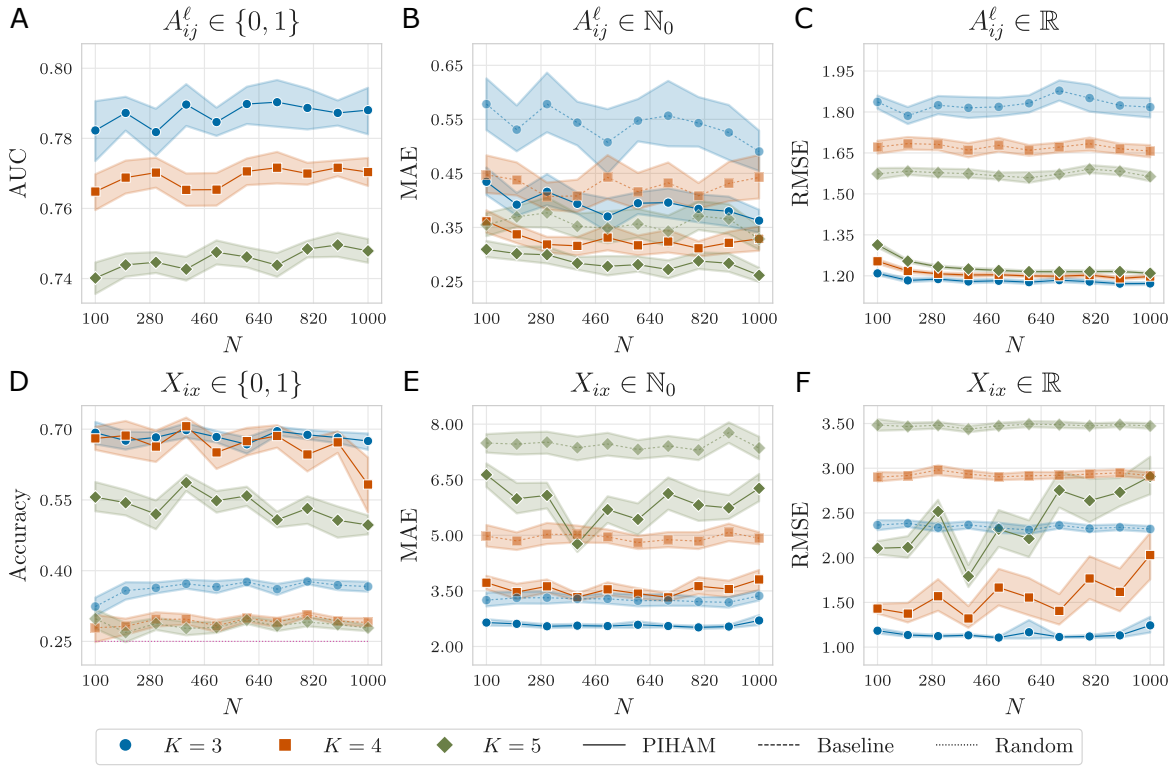


Figure 5.2: Prediction performance on synthetic data. We analyze synthetic attributed multilayer networks with $L = 3$ heterogeneous layers (one with binary interactions (A), one with nonnegative discrete weights (B), and one with real values(C)), three node covariates (one categorical with $Z = 4$ categories (D), one representing nonnegative discrete values (E), and one involving real values (F)), varying number of nodes N, and diverse number of overlapping communities K. We employ a 5-fold cross-validation procedure and plot averages and confidence intervals over 20 independent samples. The prediction performances are measured with different metrics according to the data type: Area Under the receiver-operator Curve (AUC) for binary interactions (A), the Maximum Absolute Error (MAE) for nonnegative discrete values (B, E), the Root Mean Squared Error (RMSE) for real values (C, F), and accuracy for categorical attributes (D). The baselines are given by the predictions obtained from either the average or the maximum frequency in the training set. For the categorical attribute, we also include the uniform random probability over Z, and for the AUC, the baseline corresponds to the random choice 0.5. Overall, PIHAM outperforms the baselines significantly for each type of information.

We assess the effectiveness of PIHAM by testing its prediction performance. To this end, we adopt a 5-fold cross-validation procedure, where we estimate the model's parameters on the training set and subsequently evaluate its prediction performance on the test set (see Appendix C for details). The presence of heterogeneous information complicates the measurement of goodness of fit, as distinct data types impose different constraints and domains. To address this complexity, we employ different metrics tailored to assess the prediction performance of each type of information. Specifically, we use the Area Under the receiver-operator Curve (AUC) for binary interactions, the Maximum Absolute Error (MAE) for nonnegative discrete values, the Root Mean Squared Error (RMSE) for real values, and the accuracy for categorical attributes. Further exploration to determine a unified metric could be a subject of future research.

The results are illustrated in Figure 5.2, where the performance of PIHAM is compared against baselines given by the predictions obtained from

either the average or the maximum frequency in the training set. For the categorical attribute, we also include the uniform random probability over Z, and for the AUC, the baseline corresponds to the random choice 0.5. Overall, PIHAM outperforms the baselines significantly for each type of information, with performance slightly decreasing as K increases. This is somewhat expected, considering the increased complexity of the scenarios. On the other hand, the performance remains consistent across varying values of N, indicating the robustness of our method and its suitability for larger networks.

Interpretation of posterior estimates. We have showcased the prediction performance of PIHAM across diverse synthetic datasets, and we now delve into the qualitative insights that can be extracted from the inferred parameters. In particular, we focus on the membership matrix $\boldsymbol{U}$. For this purpose, we examine the results obtained through the analysis of the synthetic data used in the section Comparison with existing methods in a homogeneous scenario, where ground truth mixed-memberships are represented as normalized vectors summing to 1. This scenario is particularly relevant for illustrating an example where the desired parameter space, defined by the simplex, differs from the inferred one existing in real-space.

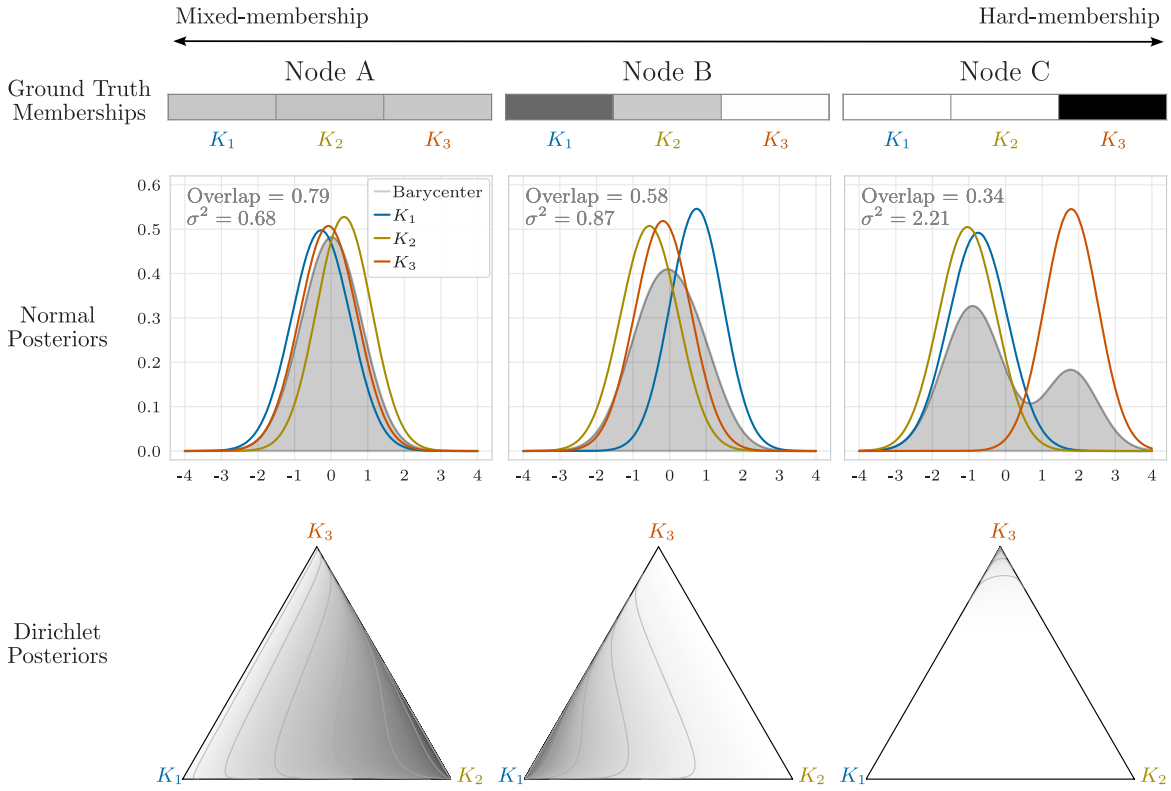


Figure 5.3: Interpretation of posterior distributions in comparison with ground truth memberships. We analyze a synthetic attributed multilayer network with ground truth mixed-memberships represented as normalized vectors summing to 1. In this case, $K = 3$. (Top row) Ground truth membership vectors for three representative nodes: Node A displays extreme mixed-membership, Node B shows a slightly lower mixed-membership, and Node C exhibits hard-membership. (Middle row) Inferred posterior distributions $\hat{U}_{ik} \sim \mathcal{N}(\hat{U}_{ik}; \hat{\mu}^U_{ik}, (\hat{\sigma}^U_{ik})^2)$, where different colors represent distinct communities, and the distribution in gray consists of the L_2-barycenter distribution. Overlap is the average of the area of overlap between every pair of distributions, and σ^2 is the variance of the barycenter distribution. (Bottom row) Transformed posterior distributions into the simplex space using the LM technique and employing Dirichlet distributions. The inferred node memberships reflect the ground truth behavior, as evidenced by the trends of Overlap and σ^2, which align with the decreasing degree of true mixed-membership. Additionally, the Dirichlet transformation provides a more straightforward interpretation, further supporting this conclusion.

To ease visualizations, we investigate a randomly selected network and focus on three representative nodes with distinct ground truth memberships: Node A has extreme mixed-membership, Node B slightly less mixed-membership, and Node C exhibits hard-membership. The results are depicted in Figure 5.3, with the top row displaying the ground truth membership vectors for these representative nodes. In the middle row, we plot the inferred posterior distributions $\hat{U}_{ik} \sim \mathcal{N}(\hat{U}_{ik}; \hat{\mu}^U_{ik}, (\hat{\sigma}^U_{ik})^2)$, where different colors represent distinct communities (in this case, $K = 3$). Through a comparative analysis of the three distributions for each node, we can gain insights into the nodes' behaviors: Node A exhibits greater overlap among the three distributions, Node B shows a slighter shift toward K_1, while Node C distinctly aligns more with community K_3. This preliminary investigation leads to the conclusion that the inferred communities reflect the ground truth behaviors. However, interpreting such patterns can be challenging, if not unfeasible, especially when dealing with large datasets. To address this issue, we quantitatively compute the area of overlap between every pair of distributions for each node and then calculate the average. For this purpose, we use the implementation proposed in (Wand et al., 2011) and we name this measure as *Overlap*. This metric ranges from 0 (indicating no overlap) to 1 (representing perfect matching between the distributions). Notably, the overlap decreases as we move from Node A to Node C, in line with the decreasing degree of mixed-membership.

Computing the Overlap for many communities can be computationally expensive due to the need to calculate all pairwise combinations. As an alternative solution, we suggest utilizing the L_2-barycenter distribution, which essentially represents a weighted average of the node-community distributions (Benamou et al., 2015; Coz et al., 2023). We show the barycenter distributions in gray in the second row of Figure 5.3. This approach allows focusing on a single distribution per node, instead of K different ones. To quantify this distribution, we calculate its variance (σ^2), where higher values indicate nodes with harder memberships, as the barycenter is more spread due to the individual distributions being more distant from each other. Conversely, lower variance suggests more overlap among the distributions, indicating a more mixed-membership scenario. We observe that σ^2 increases as we decrease the degree of mixed-membership, a trend consistent with that of the Overlap. Further details on the barycenter distribution and the metrics are provided in Appendix C.

To facilitate interpretability, a practitioner may desire to work within the simplex space. This also reflects the ground truth parameter space, as opposed to the normal posterior distributions. As discussed in the section Parameter interpretation, PIHAM employs the LM technique. This has the capability to transform in a principled way every membership vector $\hat{\boldsymbol{U}}_i$ into the simplex space using Dirichlet distributions. The outcomes of this transformation are depicted in the bottom row of Figure 5.3. By investigating these plots, it becomes even more apparent how the inferred memberships closely resemble the ground truth: the Dirichlet distributions gradually concentrate more towards a specific corner (K_1 for Node B and K_3 for Node C), instead of spreading across the entire area (as observed for Node A).

With this example, we presented a range of solutions for interpreting the

posterior distributions associated with the inferred node memberships. These options are not exhaustive, and other approaches may also be considered. For instance, a practitioner might focus solely on analyzing the point estimates for the sake of facilitating comparisons with the ground truth. In such cases, as discussed in the section Parameter interpretation, two procedures can be employed: i) applying a transformation to the point estimates, such as softmax, to align them with the ground truth space, or ii) using a sufficient statistic of the posterior distribution, where the mean of the Dirichlet distribution is a suitable option. The choice between these various approaches should be guided by the specific application under study, and the provided example serves as just one illustration.

Analysis of a social support network of a rural Indian village

We now turn our attention to the analysis of a real-world dataset describing a social support network within a village in Tamil Nadu, India, referred to as "Alakāpuram" (Power, 2015; Power, 2017). The data were collected in 2013 through surveys, in which adult residents were asked to nominate individuals who provided various types of support, such as running errands, offering advice, and lending cash or household items. Additionally, several attributes were gathered, encompassing information like gender, age, and caste, among others. The pre-processing of the dataset is described in Appendix C. The resulting heterogeneous attributed multilayer network comprises $N = 419$ nodes, $L = 7$ layers, and $P = 3$ node attributes. The initial six layers depict directed binary social support interactions among individuals, with average degree ranging from 1.8 to 4.2. The seventh, instead, contains information that is proportional to the geographical distance between individuals' households. The adjacency tensor is then represented as $\boldsymbol{A} = \{\boldsymbol{A}^{\ell} \in \{0,1\}^{N\times N} \; \forall \ell \in [1,6], \boldsymbol{A}^{7} \in \mathbb{R}^{N\times N}\}$. As node covariates, we consider the caste attribute with $Z_{caste} = 14$ categories, the religion attribute with $Z_{religion} = 3$ categories, and the attribute representing the years of education, that is $\boldsymbol{X}_{\cdot 3} \in \mathbb{N}_0^N$. Ethnographic work and earlier analyses (Power, 2017; Power and Ready, 2018) suggest that these attributes play an important role in how villagers relate to one another, with certain relationships being more strongly structured by these identities than others.

Remark 5.4 (Summary of Experiment) We want to understand the communities in a specific rural Indian village. We combine information about the surveyed relationships between the village members and their attributes, such as religion or caste membership. PIHAMis able to find highly plausible communities that can be seen in Figure 5.4.

Inference and prediction performance. We describe the likelihood of the real-world heterogeneous attributed multilayer network according to Equation 5.2 and 5.4, customized to suit the data types under examination. In particular, we employ Bernoulli distributions for the binary layers $[\boldsymbol{A}^{\ell}]_{\ell\in[1,6]}$ and Gaussian distributions for the distance layer $\boldsymbol{A}^{7}$. Moreover, we characterize the attributes caste $\boldsymbol{X}_{\cdot 1}$ and religion $\boldsymbol{X}_{\cdot 2}$ using Categorical distributions, and model the covariate $\boldsymbol{X}_{\cdot 3}$ with a Poisson distribution. The choice of the model hyperparameters and the algorithmic settings used in our experiments are described in the Supplementary Information.

Similarly to many real-world datasets, we lack the information about the true parameters underlying the network, including the node memberships. Hence, to determine the number of communities K, we employ a 5-fold cross-validation procedure for $K \in [1, 10]$ and select the value

that exhibits the optimal performance. Detailed results are displayed in Table C.2 in Appendix C. We set $K = 6$ as it achieves the best performance across the majority of prediction metrics. In fact, selecting a single metric to summarize and evaluate results in a heterogeneous setting is nontrivial, as discussed in the section Validation on heterogeneous data.

The results in Table C.2 additionally validate PIHAM's performance in inference tasks like edge and covariate prediction. Overall, our method demonstrates robust outcomes with the chosen fixed value of K and consistently outperforms the baselines, which are omitted for brevity.

Qualitative interpretation of the inferred parameters. We now shift our attention to analyze the results qualitatively, specifically focusing on the inferred communities. For easier interpretation, we apply a softmax transformation to the MAP estimates $\hat{\boldsymbol{\mu}}_i^{\boldsymbol{U}}$, allowing us to treat node memberships as probabilities. Opting for the softmax over the mean of the posterior Dirichlet distributions is primarily for visualization purposes, as it results in slightly less mixed-memberships, thereby improving clarity. The middle and bottom rows of Figure 5.4 depict the inferred out-going communities $\hat{\boldsymbol{U}}_i$, where darker values in the grayscale indicate higher values in the membership vector. In addition, the top row of Figure 5.4 displays the node attributes included in our analysis. Note also that the nodes' position reflects the geographical distance between individuals' households, and the depicted interactions refer to the first layer (talk about important matters). A full representation of the six binary layers is shown in Figure C.2 in Appendix C.

Upon initial examination, we observe a correspondence between various detected communities and the covariate information. For instance, the first and second communities predominantly consist of nodes belonging to the Yatavar and Paṟaiyar castes, respectively. Similarly, K_3 comprises nodes from the Kulalar and Maṟavar castes. This observation is supported by the inferred $K \times Z_{caste}$-dimensional matrix $\hat{\boldsymbol{H}}_{\cdot \mathbf{1}}$ (see Figure C.3 in Appendix C), which explains the contributions of each caste category to the formation of the k-th community. Furthermore, the affinity tensor $\hat{\boldsymbol{W}}$ (see Figure C.5 in Appendix C) suggests that these communities have an assortative structure, where nodes tend to interact more with individuals belonging to the same community as with those from different communities. This pattern reflects a typical behavior in social networks (M. E. Newman, 2002). Additionally, note that these communities contain nodes that are geographically close to each other and, in some cases, very distant from the majority.

In contrast to the first three, communities K_4, K_5, and K_6 are more nuanced. In fact, they are predominantly comprised of nodes from the Paḷḷar caste, which, however, is also the most represented caste in the dataset. Despite that, we observe some differences by examining other parameters. For instance, K_4 exhibits a strong assortative community structure, contrasting with the less structured nature of K_5 and K_6. This suggests that interactions play a more relevant role than attributes in determining the memberships of K_4. On the other hand, the attribute $\boldsymbol{X}_{\cdot \mathbf{3}}$ seems to play a bigger role in determining K_6, which includes nodes with more years of education. This correlation is depicted in Figure C.4

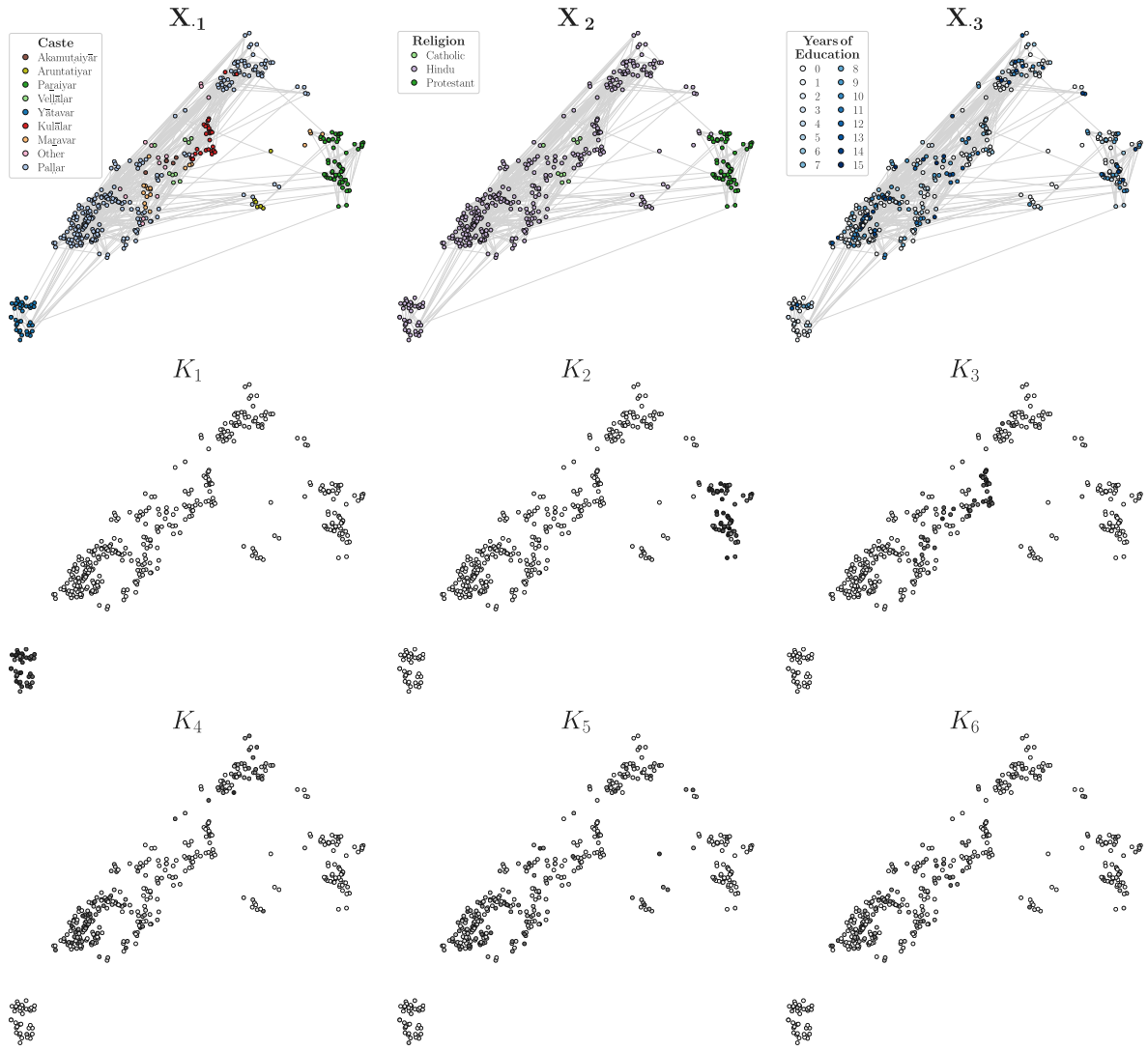


Figure 5.4: Inference of overlapping communities in a social support network. We analyze a real-world heterogeneous attributed multilayer network, which was collected in 2013 through surveys in the Indian village. This network comprises six binary layers representing directed social support interactions among individuals, alongside an additional layer reflecting information proportional to the distance between individuals' households. (Top row) As node covariates, we consider caste $\boldsymbol{X}_{\cdot 1}$, religion $\boldsymbol{X}_{\cdot 2}$, and years of education $\boldsymbol{X}_{\cdot 3}$. For privacy reasons, nodes belonging to castes with fewer than five individuals are aggregated into an "Other" category. Moreover, the displayed interactions refer only to the first layer (talk about important matters) to enhance clarity in visualization. (Middle-Bottom rows) We display the MAP estimates of the out-going communities inferred by PIHAM. For easier interpretation, we apply a softmax transformation to the MAP estimates of the membership vectors, and darker values in the grayscale indicate higher values in the membership vector $\hat{\boldsymbol{u}}_i$. The position of the nodes reflects the geographical distance between individuals' households. In summary, the inferred communities do not exclusively align with a single type of information. Rather, PIHAM incorporates all input information to infer partitions that effectively integrate them in a meaningful way.

in Appendix C, where the posterior distribution $\mathcal{N}(\hat{H}_{63}; \hat{\mu}^H_{63}, (\hat{\sigma}^H_{63})^2)$ of education years in K_6 significantly differs and is distant from the others.

By looking at the affinity matrices of the seven layers in Figure C.5, we see how layers have predominantly an assortative structure, but show also variations for certain layers. For instance, L_2 (help finding a job) has few non-zero diagonal values, suggesting that this type of support is one for which people must sometimes seek out others in different communities. In particular, L_7, corresponding to the geographical distance between nodes, has several off-diagonal entries, particularly for communities K_4, K_5, and K_6, suggesting a weakened effect for physical proximity for those communities.

Taken together, these findings suggest that the inferred communities do not solely correlate with one type of information, which may be the most dominant. Instead, PIHAM utilizes all the input information to infer partitions that effectively integrate all of them in a meaningful manner. In addition, the inferred affinity matrices illustrate how different layers can exhibit different community structures, a diversity that can be captured by our model.

5.4 Conclusion

In this work, we have introduced PIHAM, a probabilistic generative model designed to perform inference in heterogeneous and attributed multilayer networks. A significant feature of our approach is its flexibility to accommodate any combination of the input data, made possible through the use of Laplace approximations and automatic differentiation methods, which avoid the need for explicit derivations. Furthermore, PIHAM employs a Bayesian framework, enabling the estimation of posterior distributions, rather than only providing point estimates for the parameters.

When compared to other methods tailored for scenarios with only one type of attribute and interaction, PIHAM demonstrates comparable performance in prediction and community detection tasks, despite its broader formulation. Moreover, our approach significantly outperforms baseline metrics in more complex settings characterized by various attribute and interaction types, where existing methods for comparison are lacking. Furthermore, PIHAM employs the Laplace matching technique, offering a theoretically grounded approach to map posterior distributions to various desired domains, facilitating interpretation.

While PIHAM constitutes a principled and flexible method to analyze heterogeneous and attributed multilayer networks, several questions remain unanswered. For example, determining the most appropriate metric for summarizing prediction performance in heterogeneous scenarios, where information spans different spaces, is not straightforward. This aspect also influences the selection of the optimal model during cross-validation procedures. While we have provided explanations for our choices, we acknowledge that this remains an open question. Similarly, when dealing with many communities, summarizing posterior distributions becomes challenging due to computational constraints. We addressed this issue by employing L_2-barycenter distributions and proposing their variance to guide interpretation. Nevertheless, we believe there is still considerable room for improvement and exploration in this area. Our method could be further extended to accommodate distinct community-covariate contributions by integrating two separate $\boldsymbol{H}$ matrices for both in-coming and out-going communities, respectively. This modification will offer clearer insights into how covariates influence the partitions, especially when discrepancies arise between in-coming and out-going communities. Additionally, it would be interesting to expand this framework to incorporate higher-order interactions, an emerging area that has shown relevance in describing real-world data Badalyan et al., 2023.

In summary, PIHAM offers a flexible and effective approach for modeling heterogeneous and attributed multilayer networks, which arguably better

captures the complexity of real-world data, enhancing our capacity to understand and analyze the organization of real-world systems.

5.5 Practical Advice & Honest Thoughts

PIHAM is an improvement over existing frameworks for network analysis because it is much more general and easy to use. For example, many previous frameworks only work for specific data types and every additional variable would imply re-deriving all update rules explicitly. Since our framework merely requires an explicit (log-)posterior and uses Automatic Differentiation (AD), no new derivations are required.

This framework has applications beyond network analysis and community detection, and can be used in various Bayesian Inference tasks. Software packages like PyTorch and JAX already provide optimized building blocks for such a framework, simplifying the process of posterior fitting for practitioners. We also discovered that standard neural network heuristics, such as learning rate schedules and hyperparameter tuning tricks, are applicable to our framework.

The framework has limitations based on the differentiability of the probabilistic model. If the model is not fully differentiable, the framework cannot be applied. Despite this constraint, the framework still covers a wide range of potential models since sums, products, and many transformations are differentiable. However, it is not applicable to general Bayesian networks, such as DAGs, because they can include conditional probabilities that are not differentiable in general. In some cases, conditionals might be replaced with differentiable transformations, such as the "reparameterization trick" (Kingma et al., 2013). Nevertheless, this is not currently possible for arbitrary conditional probabilities.

While the framework shares many benefits with neural network training, it also inherits most of the problems. For example, there can be numerical instabilities during training, hyperparameter selection can be costly and unprincipled, and models can get stuck in local optima during training.

On balance, I would use this framework over any alternative that I'm aware of because it is more flexible and easier to set up. However, for high-dimensional problems (i.e. large parameter count), I would only compute the mode and not estimate the Hessian due to prohibitive costs.

Part III.

Conclusion

6. Conclusion

Bayesian Machine Learning offers significant advantages, including the ability to quantify uncertainty, seamlessly incorporate prior knowledge, easily specify generative processes, and achieve better generalization performance on out-of-distribution data. However, to fully harness these benefits, Bayesian tools must be accessible to practitioners and competitive with non-Bayesian alternatives both in terms of usability and computational efficiency. Despite substantial progress in the development of Bayesian techniques, many existing approaches remain computationally demanding, particularly when dealing with large-scale datasets or complex Bayesian models.

In this thesis, our objective was to develop fast and scalable Bayesian Machine Learning techniques. As highlighted in the introduction, these methods have the potential to:

1. Unlock new applications, particularly in large-scale settings where computational efficiency is crucial.
2. Serve as default baselines against which high-cost, high-fidelity methods can be compared and evaluated.
3. Provide informative priors for high-cost, high-fidelity methods, potentially reducing overall computational costs by guiding the exploration of the parameter space.

The following summary briefly outlines each paper and its impact, keeping these motivations at the forefront.

6.1 Summary & Impact

Laplace Matching

Gaussian distributions play a central role in probabilistic machine learning due to their ubiquity, well-established theoretical foundations, and desirable properties for Bayesian inference. Consequently, many tools and frameworks in the field are built around Gaussians or assume Gaussian distributions as the default choice. However, not all data types are well approximated by Gaussians. For instance, some distributions, such as the Beta distribution, are defined on the probability simplex, while others, like the Gamma distribution, are restricted to positive real numbers. These non-Gaussian distributions are often less well understood and may lack the extensive toolbox and favorable traits associated with Gaussians.

Remark 6.1 ("Gaussian Inference is Linear Algebra") Or how Philipp would say, "Gaussian Inference is linear algebra at its core" (PML, Lecture 06, slide 28).

In this chapter, we presented Laplace Matching, a simple technique that extends the advantages of Gaussian inference to every datatype with an exponential family conjugate prior. The key idea behind Laplace Matching is to transform the random variable of the exponential family into a different basis where it can be better approximated by a Gaussian distribution. By performing a Laplace approximation in this transformed

basis, we establish a bi-directional mapping between the parameters of the exponential family and the Gaussian distribution.

Laplace Matching is easy to implement and fast. It can be applied to any Gaussian latent model on non-Gaussian data with an exponential family conjugate prior, encompassing a wide range of commonly used probabilistic models. This versatility positions Laplace Matching as a powerful tool in the arsenal of probabilistic modeling techniques.

By leveraging Laplace Matching, we enabled novel applications. One notable example is the modeling of currency covariances using a latent Gaussian model, showcasing the potential of Laplace Matching to tackle high-dimensional complex problems. Furthermore, we conducted extensive comparisons, using Laplace Matching as a baseline against various high-fidelity high-cost methods. Notably, Laplace Matching achieves performance levels comparable to these more computationally intensive approaches while maintaining a significant advantage in terms of speed, often outperforming them by several orders of magnitude. This efficiency aligns well with our goal of developing fast and effective inference techniques.

Laplace Bridge for Bayesian Neural Networks

The Laplace Bridge for Bayesian Neural Networks is a noteworthy application of Laplace Matching. In classification neural networks, the final layer can be interpreted as a latent Gaussian model with probability outputs. The conventional approach involves sampling from this Gaussian distribution and then individually transforming each sample using the softmax function. However, the Laplace Bridge offers an alternative method. Instead of transforming individual samples, the Laplace Bridge enables the direct transformation of the entire Gaussian distribution into a Dirichlet distribution over the outputs. This approach provides a more principled and efficient way of obtaining a probabilistic representation of the network's predictions.

We demonstrated that the performance of our proposed method is comparable to other commonly used techniques while offering a significant advantage in computational efficiency. This increased speed allows for seamless scaling of the transformation to high-dimensional datasets, such as Imagenet with its 1000 classes. Moreover, the properties of the Dirichlet distribution enable novel approaches and techniques. For instance, by leveraging the fact that marginals of Dirichlet distributions are also Dirichlet distributions, we developed an uncertainty-aware top-k method. This method utilizes the uncertainty estimates provided by the neural network to individually determine the number of classes the network seriously considers for each image. By incorporating uncertainty information into the decision-making process, our approach offers a more nuanced and reliable classification framework that goes beyond simply choosing the same constant top-k classes for every image.

Flexible inference in heterogeneous and attributed multilayer networks

We extended the application of Laplace Matching from linear models to a broader framework for fast Bayesian inference in probabilistic

networks. This new framework, PIHAM (Probabilistic Inference in Heterogeneous and Attributed Multilayer Networks), accommodates sums, multiplications, and differentiable transformations.

PIHAM is demonstrated through network inference, specifically designed for both directed and undirected networks with heterogeneous attributes across multiple layers. It effectively combines diverse data types about individuals in a social network, such as personal attributes (e.g. religion or net worth) and inter-individual relationships, which may be binary, count, or real-valued.

The framework employs appropriate probability distributions for different data types (e.g., Poisson for count data and Bernoulli for binary data) and defines a joint latent model where all parameters are transformed Gaussians. Automatic Differentiation is then used to obtain a Laplace approximation for all Gaussian latent variables. Finally, Laplace Matching transforms these Gaussian latent variables back to their intended domains for analysis.

We applied PIHAM to a real-world dataset describing relationships and attributes in a small Indian village. The method successfully integrated diverse information types and identified plausible community structures.

The applicability of this framework extends beyond network inference to any probabilistic model involving sums, products, and differentiable transformations, broadening its potential use cases significantly.

6.2 Future Work

This thesis represents a small portion of a potentially vast and ambitious research endeavor. In the following, we explore two research directions that naturally extend from the work presented here, serving as logical next steps in advancing the field of fast Bayesian inference.

Fast Bayesian approximate inference for arbitrary probabilistic graphical models

In this thesis, we initially focused on developing a fast approximate Bayesian inference scheme for latent Gaussian models and subsequently extended it to a subset of probabilistic graphical models that involve multiplication and addition operations. However, it is important to acknowledge that our current frameworks have limitations in their applicability to arbitrary probabilistic graphical models. Specifically, they lack the capability to handle conditionals, which is a significant constraint.

Several general Bayesian inference frameworks, such as Variational Inference and Markov Chain Monte Carlo methods, provide the flexibility to specify arbitrary probabilistic graphical models. These frameworks offer a powerful toolset for modeling complex relationships and dependencies among variables in a probabilistic setting. In addition to these inference frameworks, various probabilistic programming languages have emerged to facilitate the specification and practical implementation

of probabilistic graphical models. These languages allow practitioners to define their desired model in a concise and intuitive manner, abstracting away the intricacies of the underlying inference algorithms. By providing a high-level interface, probabilistic programming languages enable users to focus on the model design while seamlessly leveraging the inference capabilities of the chosen framework.

An ambitious goal would be to develop a fast approximate inference framework that leverages transformation of variables, Laplace approximations, automatic differentiation, and other statistical techniques to handle arbitrary probabilistic graphical models. This framework could be seamlessly integrated into existing probabilistic programming languages in multiple ways, offering significant benefits to users.

Firstly, the framework could serve as a standalone feature, empowering users to scale their methods to much larger dataset and model sizes compared to what is currently feasible with existing approaches.

Secondly, the framework could support a two-stage approach, where high-fidelity-high-cost methods utilize the posteriors obtained from the fast techniques as priors. By leveraging the outputs of the fast inference framework as informative priors, the overall computational costs of the high-fidelity methods could be significantly reduced. The success of this two-stage approach hinges on the quality of the approximations provided by the fast inference framework. If the approximations are sufficiently accurate, they can serve as effective priors for the high-fidelity methods, guiding them towards the desired solution more efficiently. By providing a good starting point, the fast approximations can potentially reduce the number of iterations and computational resources required by the high-fidelity methods to converge to a satisfactory result.

A key challenge in developing such a framework lies in finding efficient approximations for updates through conditionals. Traditionally, expensive sampling methods have been employed to address conditional distributions. However, in certain cases, innovative approaches like the reparametrization trick (Kingma et al., 2013) can be used for faster automatic differentiation for conditional distributions. The reparametrization trick has been successfully applied to continuous distributions (Maddison et al., 2017), discrete distributions (Tokui et al., 2016), and even acquisition functions (J. T. Wilson et al., 2017). An initial step towards developing a framework for fast Bayesian approximate inference would involve integrating these various reparametrization techniques with other rapid inference methods.

Scale fast Bayesian inference to large Neural Networks

Neural networks have achieved remarkable progress in recent years, enabling a wide range of new applications across various domains. Despite their impressive performance, understanding the internal mechanisms and decision-making processes of neural networks remains a significant challenge. Moreover, properly quantifying the uncertainty associated with the parameters of neural networks is another important aspect that requires attention.

A natural extension of this thesis would be to explore scalable techniques that combine the strengths of neural networks and probabilistic graphical models. The aim would be to develop systems that leverage neural networks for perception, translating stimuli into more abstract variables, and then employ probabilistic graphical models to manipulate these abstract variables. Ideally, this approach would enable a more robust understanding and enhanced steerability of the overall system.

There are already multiple approaches that combine probabilistic inference and neural networks in various ways.

For example, Johnson et al. (2016) use a probabilistic graphical model as a latent backbone and a variational autoencoder as a recognition network that translates between the raw data and the latent variables. The structure is jointly trained with a single objective. While the theory is appealing, the training is numerically challenging, more error-prone, and slower than typical training for both single components would be.

As a second example, consider Normalizing Flows (Kobyzev et al., 2019), which describe a transformation of a simple probability distribution (e.g., a Gaussian) into a different distribution by a sequence of invertible and differentiable mappings. These mappings are often learned using neural networks and can thereby learn a generative distribution of complex datatypes such as images.

Thirdly, GFlowNets (Bengio et al., 2021) learn a flow function, forward & backward transition policies and a termination probability. This allows them to dynamically generate new flow graphs by taking the current state as an input and sampling an action based on it. E. J. Hu et al. (2023) drastically improves the chain-of-thought performance of an LLM by interpreting reasoning as a probabilistic inference problem and then fine-tuning the LLM using a GFlowNet diversity-seeking reinforcement learning objective. While the resulting LLM does not explicitly use a graphical model to do the reasoning, it is implicitly learned through the objective function and thus the resulting policy is more interpretable.

While normalizing Flows and GFlowNets have shown state-of-the-art results for their scale, they are still more costly to train than standard neural networks. A natural extension of this thesis would be to look for tricks and approximations for both of these approaches that prioritize speed above everything else but allow to scale them up even further to the largest state-of-the-art models.

Part IV.

Appendix

Appendix A: Laplace Matching A.

A.1 Example Transformation

It is well-known that a chi-square distribution with k degrees of freedom describes the sum of the squares of k independent, standard normal random variables. To introduce a certain 'trick' we show the forward and backward transformation between chi-square and normal distribution when $k = 1$. The trick deals with the problem that the square-root is not a bijective transformation and has to be split into multiple ranges.

Let X be normal with $\mu = 0, \sigma^2 = 1$. Let $Y = X^2$ and therefore $g(x) = x^2$, which is neither monotonic nor injective. Take $I_1 = (-\infty, 0)$ and $I_2 = [0, +\infty)$. Then g is monotonic and injective on I_1 and I_2 and $I_1 \cup I_2 = \mathbb{R}$. $g(I_1) = (0, \infty)$ and $g(I_2) = [0, \infty)$. Then $g_1^{-1} : [0, \infty) \to \mathbb{R}$ by $g_1^{-1}(y) = -\sqrt{y}$ and $g_2^{-1} : [0, \infty) \to \mathbb{R}$ by $g_2^{-1}(y) = \sqrt{y}$. Let furthermore $\mathbf{1}_r(y)$ be one if y is the range r and 0 otherwise. The derivative is given by

$$\left| \frac{\partial g_i^{-1}(y)}{\partial y} \right| = \left| \frac{1}{2\sqrt{y}} \right| = \frac{1}{2\sqrt{y}}$$

Applying a change of variable (Equation 2.4) we transform a standard normal distribution to a chi-square distribution.

$$f_Y(y) = f_X(g_1^{-1}(y)) \left| \frac{\partial g_1^{-1}(y)}{\partial y} \right| \mathbf{1}_r(y) + f_X(g_2^{-1}(y)) \left| \frac{\partial g_2^{-1}(y)}{\partial y} \right| \mathbf{1}_r(y) \quad \text{(A.1a)}$$

$$= \frac{1}{\sqrt{2\pi}} \exp(-\frac{y}{2}) \frac{1}{2\sqrt{y}} + \frac{1}{\sqrt{2\pi}} \exp(-\frac{y}{2}) \frac{1}{2\sqrt{y}} \qquad (y > 0) \quad \text{(A.1b)}$$

$$= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{y}} \exp(-\frac{y}{2}) \quad \text{(A.1c)}$$

$$= \frac{1}{2^{\frac{1}{2}}} \frac{1}{\Gamma(\frac{1}{2})} y^{-\frac{1}{2}} \exp(-\frac{1}{2}) \quad \text{(A.1d)}$$

$$= \chi^2(1) \quad \text{(A.1e)}$$

The 'trick' was to split up the variable transformation in two parts to adjust for the fact that the square-root is not bijective on $\mathbb{R}$. We can reverse the same procedure to transform a chi-square to a normal distribution. We keep the variable names from before. Let $X = \sqrt{Y}$ and therefore $h(x) = \sqrt{x}$. Then $h_1^{-1} : \mathbb{R} \to (-\infty, 0)$ by $h_1^{-1}(x) = -x^2$ and $h_2^{-1} : \mathbb{R} \to [0, \infty)$ by $h_2^{-1}(x) = x^2$. Then

$$\left| \frac{\partial h_i^{-1}(y)}{\partial y} \right| = |2y| \quad \text{(A.2)}$$

and

$$f_X(x) = f_y(h_1^{-1}(x))\left|\frac{\partial h_1^{-1}(y)}{\partial y}\right|\mathbf{1}_r(y) + f_y(h_2^{-1}(x))\left|\frac{\partial h_2^{-1}(y)}{\partial y}\right|\mathbf{1}_r(y)$$

$$= \frac{1}{\sqrt{2\pi}}\frac{1}{2\sqrt{x^2}}\exp(-\frac{x^2}{2})|2x|\mathbf{1}_{(-\infty,0)}(x) + \frac{1}{\sqrt{2\pi}}\frac{1}{2\sqrt{x^2}}\exp(-\frac{x^2}{2})|2x|\mathbf{1}_{[0,\infty)}(x) \tag{A.3a}$$

$$= \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2}) \tag{A.3b}$$

$$= \mathcal{N}(x; \mu = 0, \sigma^2 = 1) \tag{A.3c}$$

which is defined on the entirety of $\mathbb{R}$.

A generalization to matrices

A positive definite matrix A has n distinct eigenvalues and 2^n possible square roots. A has a decomposition $A = UDU^{-1}$ where Us columns are the eigenvectors of A and D is a diagonal matrix containing the eigenvalues λ_i of A. Any square root of A is given by $A^{\frac{1}{2}} = UD^{\frac{1}{2}}U^{-1}$. Since there are two possible choices for the square root of each eigenvalue $+\sqrt{\lambda_i}$ and $-\sqrt{\lambda_i}$ there are 2^n possible choices for the matrix $D^{\frac{1}{2}}$.

If the matrix A is symmetric and positive definite, its square root $A^{\frac{1}{2}}$ is also symmetric because of the decomposition $A = UDU^{-1}$. The square root of A that uses only the positive square roots of the eigenvalues is called the principle square root of A.

The Gamma distribution is the 1-dimensional special case of the Wishart distribution and therefore has $2^1 = 2$ possible options for $\lambda^{\frac{1}{2}}$, namely $-\lambda^{\frac{1}{2}}$ and $+\lambda^{\frac{1}{2}}$. Higher dimensional functions such as Wishart and inverse Wishart have 2^n different possibilities which have to be accounted for within the transformation.

A.2 Derivations of the Transformations

A.2.1 Exponential Distribution

Standard Exponential Distribution

The pdf of the exponential distribution is

$$p(x|\lambda) = \lambda\exp(-\lambda x) \tag{A.4a}$$

$$= \exp\left[-\lambda x + \log\lambda\right] \tag{A.4b}$$

with exponential family values $h(x) = 1$, $\phi(x) = x$, $w = -\lambda$ and $Z(\lambda) = -\log\lambda$.

Laplace Approximation of the Exponential Distribution

$$\begin{aligned} \text{log-pdf: } & (\log\lambda - \lambda x) \\ \text{1st derivative: } & -\lambda \\ \text{2nd derivative: } & 0 \end{aligned}$$

The Laplace Approximation in the standard base is not defined since the second derivative is not positive.

Log-transformed Exponential Distribution

We choose $X = \log(Y)$ with $g(x) = \log(x)$ and $x(y) = g^{-1}(y) = \exp(y)$. Also, $\left|\frac{\partial x(y)}{\partial y}\right| = \exp(y)$. It follows that the pdf in log-basis is

$$\begin{aligned} \mathcal{E}_{Y_{\log}}(y;\lambda) &= \lambda\exp(-\lambda x(y))\cdot\exp(y) && \text{(A.5a)} \\ &= \lambda\exp(-\lambda\exp(y)+y) && \text{(A.5b)} \\ &= \exp\left[-\lambda\exp(y)+y+\log\lambda\right] && \text{(A.5c)} \end{aligned}$$

with exponential family values $h(y) = 1$, $\phi(x) = (y, \exp(y))$, $w = (1, -\lambda)$ and $Z(\lambda) = \log\lambda$.

Laplace Approximation of the log-transformed Exponential Distribution

$$\begin{aligned} \text{log-pdf: } & -\lambda\exp(y)+y+\log\lambda \\ \text{1st derivative: } & -\lambda\exp(y)+1 \\ \text{mode: } & y = \log(1/\lambda) \\ \text{2nd derivative: } & -\lambda\exp(y) \\ \text{insert mode: } & -\lambda\exp(1/\lambda) = -1 \\ \text{invert \& times -1: } & \sigma^2 = 1 \end{aligned}$$

Therefore, the Laplace approximation in the transformed basis is given by $\mathcal{N}(y, \log(1/\lambda), 1)$.

The Bridge for the log-transformed Exponential Distribution

We have already found μ and σ. The inverse transformation is easily found through $\mu = \log(1/\lambda) \Leftrightarrow \lambda = 1/\exp(\mu)$. In summary:

$$\begin{aligned} \mu &= \log(1/\lambda) && \text{(A.6a)} \\ \sigma^2 &= 1 && \text{(A.6b)} \\ \lambda &= 1/\exp(\mu) && \text{(A.6c)} \end{aligned}$$

A visual interpretation can be found in Figure A.1.

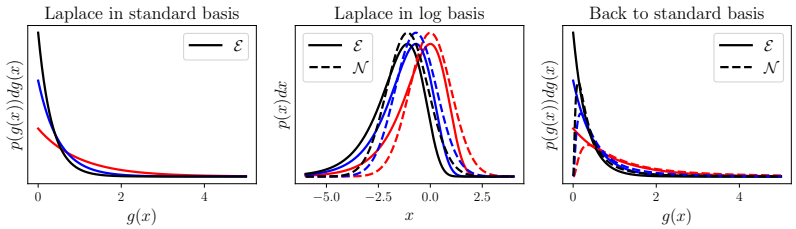


Figure A.1: log-Bridge for the Exponential distribution

Sqrt-transformed Exponential Distribution

We transform the Exponential distribution with the sqrt-transformation, i.e. $Y = \sqrt{X}, g(x) = \sqrt{x}, x_1(y) = g_1^{-1}(y) = -y^2, x_2(y) = g_2^{-1}(y) = y^2$ and $\left|\frac{\partial x_i(y)}{\partial y}\right| = \left|\frac{\partial g_i^{-1}(y)}{\partial y}\right| = |2y|$.

$$\mathcal{E}_{Y_{\text{sqrt}}}(y;\lambda) = \lambda \exp(-\lambda y^2) \cdot 2y \tag{A.7a}$$
$$= 2 \cdot \exp\left[\log(y) - \lambda y^2 + \log\lambda\right] \tag{A.7b}$$

with exponential family values $h(y) = 2$, $\phi(y) = (\log(y), y^2))$, $w = (1, -\lambda)$ and $Z(\lambda) = \log\lambda$.

Laplace Approximation of the sqrt-transformed Exponential Distribution

$$\text{log-pdf: } \log(y) - \lambda y^2 + \log\lambda$$
$$\text{1st derivative: } \frac{1}{y} - 2\lambda y$$
$$\text{mode: } y = \sqrt{\frac{1}{2\lambda}}$$
$$\text{2nd derivative: } -\frac{1}{y^2} - 2\lambda$$
$$\text{insert mode: } -\frac{1}{\frac{1}{2\lambda}} - 2\lambda = -4\lambda$$
$$\text{invert \& times -1: } \sigma^2 = \frac{1}{4\lambda}$$

Therefore the resulting Gaussian is $\mathcal{N}\left(y; \mu = \sqrt{\frac{1}{2\lambda}}, \sigma^2 = \frac{1}{4\lambda}\right)$

The Bridge for the sqrt-transformed Exponential Distribution

To get the inverse of the Bridge in the sqrt-base we can invert the mode $\mu = \sqrt{\frac{1}{2\lambda}} \Leftrightarrow \frac{1}{2\mu^2}$. In summary we have

$$\mu = \sqrt{\frac{1}{2\lambda}} \tag{A.8a}$$
$$\sigma^2 = \frac{1}{4\lambda} \tag{A.8b}$$
$$\lambda = \frac{1}{2\mu^2} \tag{A.8c}$$

A visual interpretation can be found in Figure A.2.

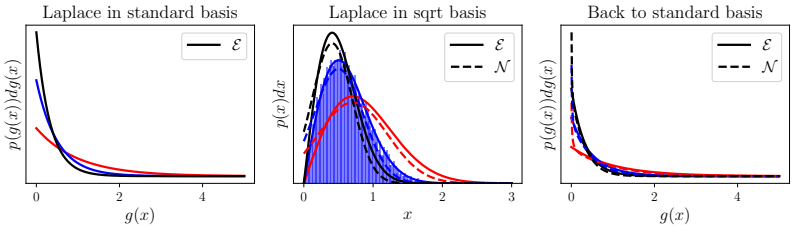


Figure A.2: sqrt-bridge for the Exponential distribution. Transformed samples indicate that the closed-form transformation is correct.

A.2.2 Gamma Distribution

Standard Gamma Distribution

The pdf of the Gamma distribution in the standard base is

$$\mathcal{G}_X(x, \alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot x^{(\alpha-1)} \cdot e^{(-\lambda x)} \tag{A.9}$$

where $\Gamma(\alpha)$ is the Gamma function. This can be written as

$$\mathcal{G}_X(x, \alpha, \lambda) = \exp\left[(\alpha - 1)\log(x) - \lambda x + \alpha \log(\lambda) - \log(\Gamma(\alpha))\right] \tag{A.10a}$$

$$= \frac{1}{x} \exp\left[\alpha \log(x) - \lambda x + \alpha \log(\lambda) - \log(\Gamma(\alpha))\right] \tag{A.10b}$$

with exponential family values $h(x) = \frac{1}{x}$, $\phi(x) = (\log x, x)$, $w = (\alpha, -\lambda)$ and $Z(\alpha, \lambda) = \log(\Gamma(\alpha)) - \alpha \log(\lambda)$.

Laplace Approximation of the Gamma Distribution

$$\begin{aligned}
\text{log-pdf: } & \log\left(\frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot x^{(\alpha-1)} \cdot e^{(-\lambda x)}\right) \\
& = \alpha \cdot \log(\lambda) - \log(\Gamma(\alpha)) + (\alpha - 1)\log(x) - \lambda x \\
\text{1st derivative: } & \frac{(\alpha - 1)}{x} - \lambda \\
\text{mode: } & \frac{(\alpha - 1)}{x} - \lambda = 0 \Leftrightarrow x = \frac{\alpha - 1}{\lambda} \\
\text{2nd derivative: } & -\frac{(\alpha - 1)}{x^2} \\
\text{insert mode: } & -\frac{(\alpha - 1)}{(\frac{\alpha-1}{\lambda})^2} = -\frac{\lambda^2}{\alpha - 1} \\
\text{invert and times -1: } & \sigma^2 = \frac{\alpha - 1}{\lambda^2}
\end{aligned}$$

The Laplace approximation of the Gamma distribution is therefore $\mathcal{N}\left(x; \frac{\alpha-1}{\lambda}, \frac{\alpha-1}{\lambda^2}\right)$.

Log-Transform of the Gamma Distribution

We transform the Gamma Distribution with the Log-Transformation, i.e. $Y = \log(X), g(x) = \log(x), x(y) = g^{-1}(x) = \exp(x)$. Also, $\left|\frac{\partial x(y)}{\partial y}\right| = \exp(y)$. The transformed pdf is

$$\mathcal{G}_{Y_{\log}}(y, \alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot x(y)^{(\alpha-1)} \cdot e^{-\lambda x(y)} \cdot \exp(y) \tag{A.11a}$$

$$= \frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot \exp(y)^\alpha \cdot e^{-\lambda \exp(y)} \tag{A.11b}$$

$$= \exp\left[\alpha y - \lambda \exp(y) - \Gamma(\alpha) + \alpha \log(\lambda)\right] \tag{A.11c}$$

with exponential family parameters $h(y) = 1$, $\phi(y) = (y, \exp(y)), \eta = (\alpha, -\lambda)$ and $Z(\alpha, \lambda) = \log(\Gamma(\alpha)) - \alpha \log(\lambda)$.

Laplace Approximation of the log-transformed Gamma Distribution

$$\begin{aligned}
\text{log-pdf:} &= \alpha \log(\lambda) - \log(\Gamma(\alpha)) + \alpha y - \lambda \exp(y) \\
\text{1st derivative:} &\ \alpha - \lambda \exp(y) \\
\text{mode:} &\ \alpha - \lambda \exp(y) = 0 \Leftrightarrow y = \log\left(\frac{\alpha}{\lambda}\right) \\
\text{2nd derivative:} &\ -\lambda \exp(y) \\
\text{insert mode:} &\ -\lambda \exp(\log\left(\frac{\alpha}{\lambda}\right)) = -\alpha \\
\text{invert and times -1:} &\ \sigma^2 = \frac{1}{\alpha}
\end{aligned}$$

The resulting Gaussian is $\mathcal{N}(y; \log\left(\frac{\alpha}{\lambda}\right), \frac{1}{\alpha})$.

The bridge for the log-transformation

We already know how to get μ and σ from λ and α. To invert we calculate $\mu = \log(\alpha/\lambda) \Leftrightarrow \lambda = \alpha/\exp(\mu)$ and insert $\alpha = 1/\sigma^2$. In summary we have

$$\mu = \log\left(\frac{\alpha}{\lambda}\right) \tag{A.12a}$$

$$\sigma^2 = \frac{1}{\alpha} \tag{A.12b}$$

$$\lambda = \frac{1}{\exp(\mu)\sigma^2} \tag{A.12c}$$

$$\alpha = \frac{1}{\sigma^2} \tag{A.12d}$$

A visual interpretation can be found in Figure A.3.

Sqrt-Transform of the Gamma Distribution

We transform the Gamma Distribution with the sqrt-transformation, i.e. $Y = \sqrt{X}, g(x) = \sqrt{x}, x_1(y) = g_1^{-1}(y) = -y^2, x_2(y) = g_2^{-1}(y) = y^2$ and

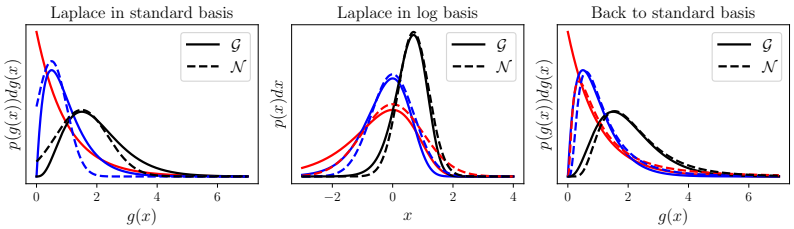


Figure A.3: log-bridge for Gamma distribution

$\left|\frac{\partial x_i(y)}{\partial y}\right| = \left|\frac{\partial g_i^{-1}(y)}{\partial y}\right| = |2y|$. We use the same 'trick' as in Appendix A.1 to split up the transformation in two parts.

$$\mathcal{G}_Y(y) = \frac{1}{2} \cdot \mathcal{G}_X(x_1(y)) \left|\frac{\partial x_1(y)}{\partial y}\right| \mathbf{1}_r(y) + \frac{1}{2} \cdot \mathcal{G}_X(x_2(y)) \left|\frac{\partial x_2(y)}{\partial y}\right| \mathbf{1}_r(y) \tag{A.13a}$$

$$= \frac{1}{2y^2} \exp[\alpha \log(y^2) - \lambda y^2 - A(\alpha,\lambda)]|2y|\mathbf{1}_{(-\infty,0)}(y) + \frac{1}{2y^2} \exp[\alpha \log(y^2) - \lambda y^2 - A(\alpha,\lambda)]|2y|\mathbf{1}_{[0,\infty)}(y) \tag{A.13b}$$

$$= \frac{1}{\sqrt{y}} \exp[2\alpha \log(y) - \lambda y^2 - A(\alpha,\lambda)]\mathbf{1}_{(-\infty,+\infty)}(y) \tag{A.13c}$$

$$= \frac{1}{\sqrt{y}} \exp[2\alpha \log(y) - \lambda y^2 - A(\alpha,\lambda)] \tag{A.13d}$$

which is defined on the entirety of $\mathbb{R}$ and is an exponential family with $h(y) = \frac{1}{y}$, $\phi(y) = (\log(y), y^2)$, $w = (2\alpha, -\lambda)$ and $Z(\alpha,\lambda) = \log(\Gamma(\alpha)) - \alpha \log(\lambda)$.

Laplace Approximation of the sqrt-transformed Gamma Distribution

$$\text{log-pdf: } (2\alpha - 1)\log(y) - \lambda y^2 + \alpha \log(\lambda) - \log(\Gamma(\alpha))$$

$$\text{1st derivative: } \frac{2\alpha - 1}{y} - 2\lambda y$$

$$\text{mode: } \frac{2\alpha - 1}{y} - 2\lambda y = 0 \Leftrightarrow y = \sqrt{\frac{\alpha - 0.5}{\lambda}}$$

$$\text{2nd derivative: } -\frac{2\alpha - 1}{x^2} - 2\lambda$$

$$\text{insert mode: } -\frac{2\alpha - 1}{\frac{\alpha-0.5}{\lambda}} - 2\lambda = -4\lambda$$

$$\text{invert and times -1: } \sigma^2 = \frac{1}{4\lambda}$$

The resulting Gaussian is $\mathcal{N}\left(y; \sqrt{\frac{\alpha-0.5}{\lambda}}, \frac{1}{4\lambda}\right)$.

The bridge for the sqrt-transformation

We already know how to get μ and σ from λ and α. To invert we calculate $\mu = \sqrt{\frac{\alpha-0.5}{\lambda}} \Leftrightarrow \alpha = \frac{\mu^2}{\lambda} - 0.5$ and insert $\lambda = \frac{4}{\sigma^2}$. In summary we have

$$\mu = \sqrt{\frac{\alpha - 0.5}{\lambda}} \tag{A.14a}$$

$$\sigma^2 = \frac{1}{4\lambda} \tag{A.14b}$$

$$\lambda = \frac{4}{\sigma^2} \tag{A.14c}$$

$$\alpha = \frac{4\mu^2}{\sigma^2} + 0.5 \tag{A.14d}$$

A visual interpretation can be found in Figure A.4.

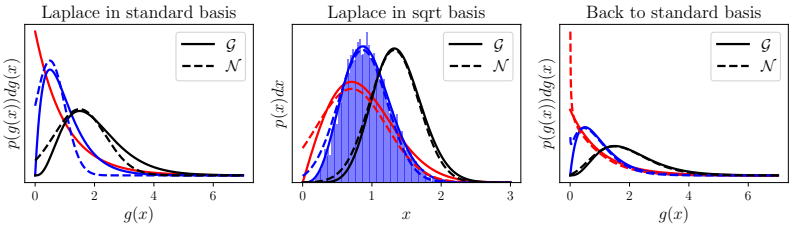


Figure A.4: sqrt-bridge for the Gamma distribution. The transformed samples indicate that the closed-form transformation is correct.

A.2.3 Inverse Gamma Distribution

Standard Inverse Gamma Distribution

The pdf of the inverse Gamma is

$$\mathcal{IG}(x, \alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{-\alpha-1} \exp(-\frac{\lambda}{x}) \tag{A.15a}$$

$$= \exp \frac{1}{x} \left[-\alpha \log(x) - \lambda/x + \alpha \log(\lambda) - \log \Gamma(\alpha)\right] \tag{A.15b}$$

with exponential family values $h(x) = \frac{1}{x}$, $\phi(x) = (\log(x), x), w = (-\alpha, -\lambda)$ and $Z(\alpha, \lambda) = \log \Gamma(\alpha) - \alpha \log \lambda$.

Laplace Approximation of the standard inverse gamma distribution

$$
\begin{aligned}
\text{log-pdf: } & (-\alpha-1)\log(y) - \lambda/x + \alpha\log(\lambda) - \log\Gamma(\alpha) \\
\text{1st derivative: } & \frac{-\alpha-1}{y} + \frac{\lambda}{y^2} \\
\text{mode: } & \frac{-\alpha-1}{y} + \frac{\lambda}{y^2} = 0 \Leftrightarrow y = \frac{\lambda}{a+1} \\
\text{2nd derivative: } & \frac{\alpha+1}{y^2} - 2\frac{\lambda}{y^3} \\
\text{insert mode: } & \frac{\alpha+1}{\frac{\lambda}{a+1}^2} - 2\frac{\lambda}{\frac{\lambda}{a+1}^3} = -\frac{(\alpha+1)^3}{\lambda^2} \\
\text{invert and times -1: } & \sigma^2 = \frac{\lambda^2}{(\alpha+1)^3}
\end{aligned}
$$

Therefore the resulting Gaussian is $q(y) = \mathcal{N}(y; \mu = \frac{\lambda}{a+1}, \sigma^2 = \frac{\lambda^2}{(\alpha+1)^3})$.

Log-Transform of the inverse Gamma distribution

We transform the Inverse Gamma distribution with the log-transformation, i.e. $Y = \log(X)$. Therefore $g(x) = \log(x)$, and thereby $x(y) = g^{-1}(x) = \exp(y)$. It follows that the new pdf is

$$
\begin{aligned}
\mathcal{IG}_{Y_{\log}}(y, \alpha, \lambda) &= \frac{\lambda^\alpha}{\Gamma(\alpha)} \exp(y)^{-\alpha-1} \exp(-\lambda/\exp(y)) \cdot \exp(y) && \text{(A.16a)} \\
&= \frac{\lambda^\alpha}{\Gamma(\alpha)} \exp(y)^{-\alpha} \exp(-\lambda/\exp(y)) && \text{(A.16b)} \\
&= \exp\left[-\alpha x - \frac{\lambda}{\exp(x)} - \log\Gamma(\alpha) + \alpha\log(\lambda)\right] && \text{(A.16c)}
\end{aligned}
$$

with exponential family values $h(y) = 1$, $\phi(y) = (y, \exp(y))$, $w = (-\alpha, \lambda)$ and $Z(\alpha, \lambda) = \log\Gamma(\alpha) - \alpha\log\lambda$.

Laplace Approximation of the log-transformed Inverse Gamma Distribution

$$
\begin{aligned}
\text{log-pdf: } & -\alpha y - \frac{\lambda}{\exp(y)} + Z(\alpha, \lambda) \\
\text{1st derivative: } & -\alpha + \frac{\lambda}{\exp(y)} \\
\text{mode: } & -\alpha + \frac{\lambda}{\exp(y)} = 0 \Leftrightarrow y = \log(\lambda/\alpha) \\
\text{2nd derivative: } & -\frac{\lambda}{\exp(y)} \\
\text{insert mode: } & -\frac{\lambda}{\exp(\log(\lambda/\alpha))} = -\alpha \\
\text{invert and times -1: } & \sigma^2 = \frac{1}{\alpha}
\end{aligned}
$$

Therefore the resulting Gaussian is $\mathcal{N}(y; \log(\frac{\lambda}{\alpha}), \frac{1}{\alpha})$.

The Bridge for the log-transformed Inverse Gamma Distribution

Inverting the equations from above we get in summary

$$\mu = \log\left(\frac{\lambda}{\alpha}\right) \tag{A.17a}$$

$$\sigma^2 = \frac{1}{\alpha} \tag{A.17b}$$

$$\alpha = \frac{1}{\sigma^2} \tag{A.17c}$$

$$\lambda = \frac{\exp(\mu)}{\sigma^2} \tag{A.17d}$$

For a visual interpretation consider Figure A.5.

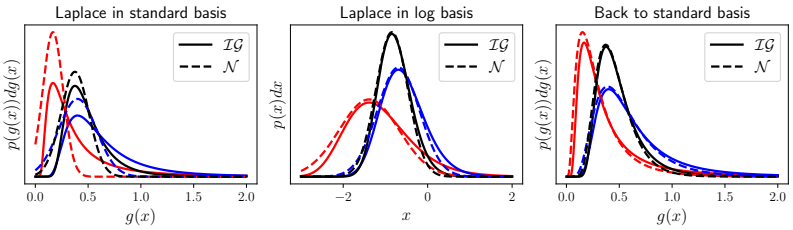


Figure A.5: inverse gamma log-bridge

Sqrt-Transform of the inverse Gamma distribution

We transform the Inverse Gamma Distribution with the sqrt-transformation, i.e. $Y = \sqrt{X}, g(x) = \sqrt{x}, x_1(y) = g_1^{-1}(y) = -y^2, x_2(y) = g_2^{-1}(y) = y^2$ and $\left|\frac{\partial x_i(y)}{\partial y}\right| = \left|\frac{\partial g_i^{-1}(y)}{\partial y}\right| = |2y|$. We use the same 'trick' as in Appendix A.1 to split up the transformation in two parts.

$$\mathcal{G}^{-1}_{Y_{\text{sqrt}}}(y) = \frac{1}{2}\cdot\mathcal{G}_X^{-1}(x_1(y))\left|\frac{\partial x_1(y)}{\partial y}\right|\mathbf{1}_r(y) + \frac{1}{2}\cdot\mathcal{G}_X^{-1}(x_2(y))\left|\frac{\partial x_2(y)}{\partial y}\right|\mathbf{1}_r(y) \tag{A.18a}$$

$$= \frac{1}{2}\frac{\lambda^\alpha}{\Gamma(\alpha)}y^{-2\alpha-1}\exp(-\lambda/y^2))|2y|\mathbf{1}_{(-\infty,0)}(y) + \frac{1}{2}\frac{\lambda^\alpha}{\Gamma(\alpha)}y^{-2\alpha-1}\exp(-\lambda/y^2))|2y|\mathbf{1}_{[0,\infty)}(y) \tag{A.18b}$$

$$= \frac{\lambda^\alpha}{\Gamma(\alpha)}y^{-2\alpha-1}\exp(-\lambda/y^2))\mathbf{1}_{(-\infty,+\infty)}(y) \tag{A.18c}$$

$$= \frac{1}{\sqrt{y}}\exp\left[(-2\alpha-1)\log(y) - \frac{\lambda}{y^2} - \log\Gamma(\alpha) + \alpha\log(\lambda)\right] \tag{A.18d}$$

with exponential family values $h(y) = \frac{1}{\sqrt{y}}, \phi(y) = (\log(y), y^2), w = (-2\alpha, -\lambda)$ and $Z(\alpha, \lambda) = \log\Gamma(\alpha) - \alpha\log\lambda$.

Laplace Approximation of the sqrt-transformed Inverse Gamma Distribution

$$\begin{aligned}
\text{log-pdf: } & (-2\alpha - 1)\log(y) - \frac{\lambda}{y^2} + Z(a, \lambda) \\
\text{1st derivative: } & -\frac{2\alpha + 1}{y} + 2\frac{\lambda}{y^3} \\
\text{mode: } & y = \sqrt{\frac{\alpha + 0.5}{\lambda}} \\
\text{2nd derivative: } & \frac{2\alpha + 1}{y^2} - 6\frac{\lambda}{y^4} \\
\text{insert mode: } & -4\frac{(\alpha + 0.5)^2}{\lambda} \\
\text{invert and times -1: } & \sigma^2 = \frac{\lambda}{4(\alpha + 0.5)^2}
\end{aligned}$$

yielding $q(y) = \mathcal{N}(y; \mu = \sqrt{\frac{\alpha+0.5}{\lambda}}, \sigma^2 = \frac{\lambda}{4(\alpha+0.5)^2})$.

The Bridge for the sqrt-transformed Inverse Gamma Distribution

We get α and λ by inverting the equations for μ and σ. In summary we get

$$\mu = \sqrt{\frac{\lambda}{\alpha}} \tag{A.19a}$$

$$\sigma^2 = \frac{\lambda}{4\alpha^2} \tag{A.19b}$$

$$\alpha = \frac{\mu^2}{4\sigma^2} - 0.5 \tag{A.19c}$$

$$\lambda = \frac{\mu^4}{4\sigma^2} \tag{A.19d}$$

For a visual interpretation consider Figure A.6.

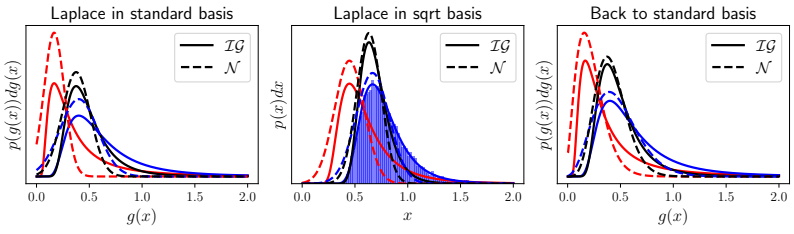


Figure A.6: inverse gamma sqrt-bridge. Transformed samples indicate that the closed-form transformation is correct.

A.2.4 Chi-square Distribution

Standard Chi-square distribution

The pdf of the Chi-square distribution in the standard basis is

$$\chi^2_X(x,k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1}\exp(-x/2) \tag{A.20a}$$

$$= \frac{1}{x}\exp\left[(k/2)\log(x) - x/2 - \log(2^{k/2}\Gamma(k/2))\right] \tag{A.20b}$$

with exponential family values $h(x) = \frac{1}{x}$, $\phi(x) = (\log(x), x), w = k/2$ and $Z(k) = \log(2^{k/2}\Gamma(k/2))$.

Laplace approximation of the standard Chi-square distribution

$$\begin{aligned}
\text{log-pdf: } & (k/2-1)\log(x) - x/2 - \log(2^{k/2}\Gamma(k/2)) \\
\text{1st derivative: } & \frac{k/2-1}{x} - \frac{1}{2} \\
\text{mode: } & \frac{k/2-1}{x} - \frac{1}{2} = 0 \Leftrightarrow x = k-2 \\
\text{2nd derivative: } & -\frac{k/2-1}{x^2} \\
\text{insert mode: } & -\frac{k/2-1}{(k-2)^2} = -\frac{(k-2)}{2(k-2)^2} \\
\text{invert and times -1: } & \sigma^2 = 2(k-2)
\end{aligned}$$

The resulting Gaussian is therefore $\mathcal{N}(x; k-2, 2(k-2))$ for $k > 2$.

Log-Transformed Chi-square distribution

we transform the distribution with $g(x) = \log(x)$, i.e. $x(y) = g^{-1}(x) = \exp(y)$. The new pdf becomes

$$\chi^2_{Y_{\log}}(y,k) = \frac{1}{2^{k/2}\Gamma(k/2)}\exp(y)^{k/2-1}\exp(-\exp(y)/2)\cdot\exp(y) \tag{A.21a}$$

$$= \frac{1}{2^{k/2}\Gamma(k/2)}\exp(y)^{k/2}\exp(-\exp(y)/2) \tag{A.21b}$$

$$= \exp\left[\frac{k}{2}y - \frac{\exp(y)}{2} - \log(2^{k/2}\Gamma(k/2))\right] \tag{A.21c}$$

with $h(y) = 1$, $\phi(y) = (y, \exp(y))$, $\eta = (k/2)$ and $Z(k) = \log(2^{k/2}\Gamma(k/2))$.

Laplace approximation of the log-transformed Chi-square distribution

$$
\begin{aligned}
\text{log-pdf: } & \frac{k}{2}y - \frac{\exp(y)}{2} - \log(2^{k/2}\Gamma(k/2)) \\
\text{1st derivative: } & \frac{k}{2} - \frac{\exp(y)}{2} \\
\text{mode: } & k/2 - \frac{\exp(y)}{2} = 0 \Leftrightarrow y = \log(k) \\
\text{2nd derivative: } & -\frac{\exp(y)}{2} \\
\text{insert mode: } & -\frac{\exp(y)}{2} = -k/2 \\
\text{invert and times -1: } & \sigma^2 = 2/k
\end{aligned}
$$

which yields the Laplace Approximation $q(y) = \mathcal{N}(y; \mu = \log(k), \sigma^2 = 2/k)$.

The Bridge for log-transform

In summary we get

$$
\begin{aligned}
\mu &= \log(k) && \text{(A.22a)} \\
\sigma^2 &= 2/k && \text{(A.22b)} \\
k &= \exp(\mu) && \text{(A.22c)} \\
\text{or} \quad k &= 2/\sigma^2 && \text{(A.22d)}
\end{aligned}
$$

For a visual interpretation consider Figure A.7.

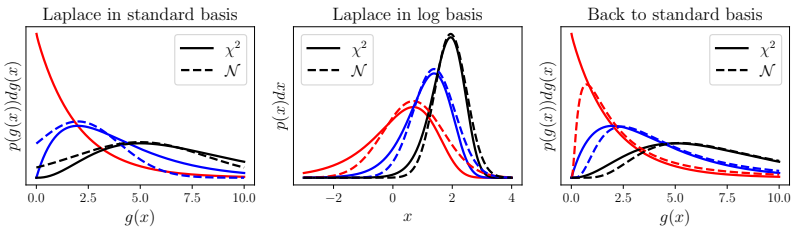


Figure A.7: Log-bridge for the Chi-square distribution.

Sqrt-Transformed Chi-square distribution

We transform the chi-square distribution with the sqrt-transformation, i.e. $Y = \sqrt{X}, g(x) = \sqrt{x}, x_1(y) = g_1^{-1}(y) = -y^2, x_2(y) = g_2^{-1}(y) = y^2$ and $\left|\frac{\partial x_i(y)}{\partial y}\right| = \left|\frac{\partial g_i^{-1}(y)}{\partial y}\right| = |2y|$. We use the same 'trick' as in Appendix A.1 to split up the transformation in two parts.

$$\chi^2_{Y_{\text{sqrt}}}(y) = \frac{1}{2} \cdot \chi^2_X(x_1(y)) \left| \frac{\partial x_1(y)}{\partial y} \right| \mathbf{1}_r(y) + \frac{1}{2} \cdot \chi^2_X(x_2(y)) \left| \frac{\partial x_2(y)}{\partial y} \right| \mathbf{1}_r(y) \tag{A.23a}$$

$$= \frac{1}{2} \frac{1}{2^{k/2}\Gamma(k/2)} y^{k-1} \exp(-\frac{y^2}{2}) |2y| \mathbf{1}_{(-\infty,0)}(y) + \frac{1}{2} \frac{1}{2^{k/2}\Gamma(k/2)} y^{k-1} \exp(-\frac{y^2}{2}) |2y| \mathbf{1}_{[0,\infty)}(y) \tag{A.23b}$$

$$= \frac{1}{2^{k/2}\Gamma(k/2)} y^{k-1} \exp(-\frac{y^2}{2}) \mathbf{1}_{(-\infty,+\infty)}(y) \tag{A.23c}$$

$$= \exp\left[(k-1)\log(y) - \frac{y^2}{2} - \log(2^{k/2}\Gamma(k/2)) \right] \tag{A.23d}$$

with exponential family values $h(y) = 1, \phi(y) = (\log(y), y^2), w = (k, 1/2)$ and $Z(k) = \log(2^{k/2}\Gamma(k/2))$.

Laplace approximation of the sqrt-transformed Chi-square distribution

$$\text{log-pdf: } ((k-1)\log(x) - \frac{x^2}{2} - \log(2^{k/2}\Gamma(k/2))$$

$$\text{1st derivative: } \frac{k}{x} - x$$

$$\text{mode: } \frac{k-1}{x} - x = 0 \Leftrightarrow x = \sqrt{k-1}$$

$$\text{2nd derivative: } -\frac{k-1}{x^2} - 1$$

$$\text{insert mode: } -\frac{k-1}{k-1} - 1 = -2$$

$$\text{invert and times -1: } \sigma^2 = 1/2$$

yielding $q(y) = \mathcal{N}(y; \mu = \sqrt{k}, \sigma^2 = 1/2)$.

The Bridge for sqrt-transform

In summary we have

$$\mu = \sqrt{k-1} \tag{A.24a}$$

$$\sigma^2 = 1/2 \tag{A.24b}$$

$$k = \mu^2 + 1 \tag{A.24c}$$

For a visual interpretation consider Figure A.8.

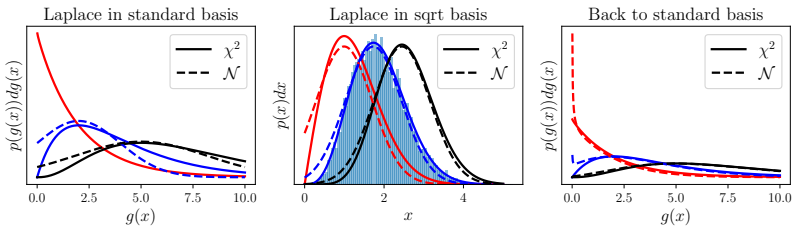


Figure A.8: Sqrt-bridge for the Chi-square distribution. Transformed samples indicated that the closed-form transformation is correct.

A.2.5 Beta Distribution

Standard Beta Distribution

The pdf of the Beta distribution in the standard basis is

$$\mathcal{B}(x,\alpha,\beta) = \frac{x^{(\alpha-1)}\cdot(1-x)^{(\beta-1)}}{B(\alpha,\beta)} \tag{A.25a}$$

$$= \exp\left[(\alpha-1)\log(x) + (\beta-1)\log(1-x) - \log(B(\alpha,\beta))\right] \tag{A.25b}$$

$$= \frac{1}{x(1-x)}\exp\left[\alpha\log(x) + \beta\log(1-x) - \log(B(\alpha,\beta))\right] \tag{A.25c}$$

with exponential family values $h(x) = \frac{1}{x(1-x)}$, $\phi(x) = (\log(x), \log(1-x))$, $w = (\alpha,\beta)$ and $Z(\alpha,\beta) = \log(B(\alpha,\beta))$ where $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and $\Gamma(x)$ is the Gamma function.

Laplace approximation of the standard Beta distribution

$$\text{log-pdf: } \log\left(\frac{x^{(\alpha-1)}\cdot(1-x)^{(\beta-1)}}{B(\alpha,\beta)}\right)$$
$$= (\alpha-1)\log(x) + (\beta-1)\log(1-x) - \log(B(\alpha,\beta)))$$
$$\text{1st derivative: } \frac{(\alpha-1)}{x} - \frac{(\beta-1)}{1-x}$$
$$\text{mode: } \frac{(\alpha-1)}{x} - \frac{(\beta-1)}{1-x} = 0 \Leftrightarrow x = \frac{\alpha-1}{\alpha+\beta-2}$$
$$\text{2nd derivative: } \frac{\alpha-1}{x^2} + \frac{\beta-1}{(1-x)^2}$$
$$\text{insert mode: } \frac{\alpha-1}{\frac{\alpha-1}{\alpha+\beta-2}^2} + \frac{\beta-1}{(1-\frac{\alpha-1}{\alpha+\beta-2})^2} = \frac{(\alpha+\beta-2)^3}{(\alpha-1)(\beta-1)}$$
$$\text{invert: } \frac{(\alpha-1)(\beta-1)}{(\alpha+\beta-2)^3}$$

The Beta distribution in standard basis is therefore approximated by $N(\mu = \frac{\alpha-1}{\alpha+\beta-2}, \sigma^2 = \frac{(\alpha-1)(\beta-1)}{(\alpha+\beta-2)^3})$.

Logit-Transform of the Beta distribution

We transform the Beta distribution using $g(x) = \log(\frac{x}{1-x})$. Therefore $x(y) = g^{-1}(y) = \sigma(y) = \frac{1}{1+\exp(-y)}$. This yields the following pdf

$$\mathcal{B}_{Y_{\text{logit}}}(y,\alpha,\beta) = \frac{1}{\sigma(y)(1-\sigma(y))}\exp\left[\alpha\log(\sigma(y))) + \beta\log(1-\sigma(y)) - \log(B(\alpha,\beta)))\right]\cdot(\sigma(y)(1-\sigma(y)) \tag{A.26a}$$

$$= \exp\left[\alpha\log(\sigma(y)) + \beta\log(1-\sigma(y)) - \log(B(\alpha,\beta)))\right] \tag{A.26b}$$

with exponential family values $h(y) = 1, \phi(y) = (\log(\sigma(y)), \log(1 - \sigma(y)), w = (\alpha, \beta)$ and $Z(\alpha, \beta) = \log(B(\alpha, \beta))$.

Laplace approximation of the logit transformed Beta distribution

$$
\begin{aligned}
\text{log-pdf: } & \log\left(\frac{\sigma(y)^\alpha \cdot (1 - \sigma(y)^\beta)}{B(\alpha, \beta)}\right) \\
& = \alpha \log(\sigma(y)) + \beta \log(1 - \sigma(x)) - \log(B(\alpha, \beta)) \\
\text{1st derivative: } & \alpha(1 - \sigma(y)) - \beta\sigma(y) \\
\text{mode: } & \alpha(1 - \sigma(y)) - \beta\sigma(y) = 0 \Leftrightarrow y = \log(\frac{\alpha}{\beta}) \\
\text{2nd derivative: } & (\alpha + \beta)\sigma(y)(1 - \sigma(y)) \\
\text{insert mode: } & (\alpha + \beta)\sigma(-\log(\frac{\beta}{\alpha}))(1 - \sigma(-\log(\frac{\beta}{\alpha}))) = \frac{\alpha\beta}{\alpha + \beta} \\
\text{invert: } & \frac{\alpha + \beta}{\alpha\beta}
\end{aligned}
$$

The Laplace approximation of the Beta in the logit base is therefore given by $\mathcal{N}(y; \mu = \log(\frac{\alpha}{\beta}), \sigma^2 = \frac{\alpha+\beta}{\alpha\beta})$.

The Bridge for the logit transformation

In summary we have

$$\mu = \log\left(\frac{\alpha}{\beta}\right) \tag{A.27a}$$

$$\sigma^2 = \frac{\alpha + \beta}{\alpha\beta} \tag{A.27b}$$

$$\alpha = \frac{\exp(\mu) + 1}{\sigma^2} \tag{A.27c}$$

$$\beta = \frac{\exp(-\mu) + 1}{\sigma^2} \tag{A.27d}$$

For a visual interpretation consider Figure A.9.

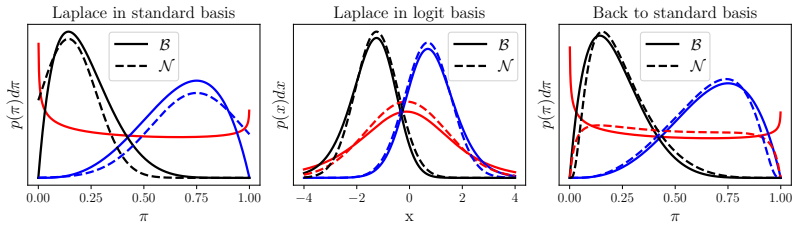


Figure A.9: beta logit bridge

A.2.6 Dirichlet Distribution

Standard Dirichlet distribution

The pdf for the Dirichlet distribution in the standard basis (i.e. probability space) is

$$\text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1} \tag{A.28a}$$

$$= \frac{1}{B(\alpha)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1} \tag{A.28b}$$

$$= \exp\left[\sum_k(\alpha_k-1)\log(\pi_k) - \log(B(\alpha))\right] \tag{A.28c}$$

$$= \frac{1}{\prod_k \pi_k}\exp\left[\sum_k \alpha_k\log(\pi_k) - \log(B(\alpha))\right] \tag{A.28d}$$

$$\tag{A.28e}$$

with sufficient statistics $\phi(x_i) = \log(x_i)$, natural parameters $w_i = \alpha_i$, base measure $h(x) = \prod_k x_k$, and partition function $Z(w) = \log(B(\alpha))$.

Laplace approximation of the standard Dirichlet distribution

$$\text{log-pdf: } f = \sum_k(\alpha_k-1)\log(x_k) - \log(B(\alpha))$$

$$\text{1st derivative: } \frac{\partial f}{\partial x_i} = \frac{\alpha_i - 1}{x_i}$$

$$\text{mode: } x_i = \frac{(\alpha_i-1)}{\sum_k \alpha_k - K}$$

$$\text{2nd derivative: } \frac{\partial^2 f}{\partial x_i \partial x_j} = -\delta_{ij}\frac{(\alpha_i-1)}{x_i^2}$$

$$\text{insert mode: } -\delta_{ij}\frac{(\sum_k \alpha_k - K)^2}{(\alpha_i-1)}$$

$$\text{invert and times -1: } \Sigma_{ij} = \delta_{ij}\frac{\alpha_i - 1}{(\sum_k \alpha_k - K)^2}$$

Which yields a diagonal Covariance matrix for the Laplace approximation.

Softmax-Transform of the Dirichlet distribution

We aim to transform the basis of this distribution from base $\mathbf{y}$ via the softmax transform to be in the new base π:

$$\pi_k(\mathbf{y}) := \frac{\exp(y_k)}{\sum_{l=1}^{K}\exp(y_l)}, \tag{A.29}$$

David J. MacKay used this transformation in (D. J. MacKay, 1998) already. We provide another, potentially simpler, derivation for the Laplace approximation of the Dirichlet in the softmax basis.

The softmax transform has no analytic inverse $\pi_k^{-1}(y)$. However, the inverse-softmax is not necessary for our computation since we assume $\pi(y)$ to be the inverse transformation already (i.e. $g^{-1}(y) = \pi(y)$). Our transformation is from a variable in $\mathbb{R}^d$ (which has d degrees of freedom) to a variable that is in $\mathbb{P}^d$ (which has $d-1$ degrees of freedom). To account

for the difference in size of the two spaces we create a helper variable for the transformation as described in the following.

We want to transform D variables y_i from $\mathbb{R}^d$ to $\tau_i = \exp(y_i)$. For τ_i to be equal to π_i we need to ensure that it sums to 1, $u = \sum_i \tau_i = 1$. With the helper-variable u our variable transform $g(\pi, u)$ becomes

$$p_{y,u}(\pi(y), u) = p_{\pi,u}(\pi(y), u) |\det g(\pi(y), u)| \tag{A.30}$$

with

$$|\det g(\pi(y), u)| = \left| \det \begin{pmatrix} \frac{\partial \tau_1}{\partial y_1} & \cdots & \frac{\partial \tau_k}{\partial y_1} & \frac{\partial u}{\partial y_1} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial \tau_1}{\partial y_k} & \cdots & \frac{\partial \tau_k}{\partial y_k} & \frac{\partial u}{\partial y_k} \\ \frac{\partial \tau_1}{\partial u} & \cdots & \frac{\partial \tau_k}{\partial u} & \frac{\partial u}{\partial u} \end{pmatrix} \right| \tag{A.31a}$$

$$= \left| \det \begin{pmatrix} \tau_1 = \exp(y_1) & \cdots & 0 & \tau_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \tau_k & \tau_k \\ 0 & \cdots & 0 & 1 \end{pmatrix} \right| \tag{A.31b}$$

$$= \prod_i^K \tau_i \tag{A.31c}$$

To get $p_y(\pi(y))$ we have to integrate out u.

$$p_y(\pi(y)) = \int_{-\infty}^{\infty} p_{\pi,u}(\pi(y), u) |\det g(\pi, u)| du \tag{A.32a}$$

$$= \int_{-\infty}^{\infty} p_\pi(\pi(y)) \prod_i^K \tau_i du \tag{A.32b}$$

$$= p_\pi(\pi(y)) \int_0^{\infty} \prod_i^K \tau_i \delta(u-1) du \tag{A.32c}$$

$$= p_\pi(\pi(y)) \int_0^{\infty} \prod_i^K \tau_i \frac{u}{u} \delta(u-1) du \tag{A.32d}$$

$$= p_\pi(\pi(y)) \int_0^{\infty} \underbrace{\prod_i^K \pi_i u}_{f(u)} \delta(u-1) du \tag{A.32e}$$

$$= p_\pi(\pi(y)) \cdot \prod_i^K \pi_i(y) \tag{A.32f}$$

$$= \frac{1}{\prod_k \pi_k(y)} \exp\left[\sum_k \alpha_k \log(\pi_k(y)) - \log(B(\alpha)) \right] \prod_k^K \pi_k(y) \tag{A.32g}$$

$$= \exp\left[\sum_k \alpha_k \log(\pi(y_k)) - \log(B(\alpha)) \right] \tag{A.32h}$$

where we used the fact that $u > 0$ since it is a sum of exponentials and $\frac{\tau_i(y)}{u} = \pi_i(y)$. We multiplied with $\delta(u-1)$ since this transformation is only valid if $\sum_i \tau_i = u = 1$ because otherwise it is not a probability space.

Additionally, we use

$$\int_{-\infty}^{\infty} f(x)\delta(x-t)dx = f(t) \tag{A.33}$$

which is known as the shifting property or sampling property of the Dirac delta function δ. Using all of the above we get the pdf of the Dirichlet distribution in the new basis $\mathbf{y}$:

$$\mathrm{Dir}_{\mathbf{y}}(\boldsymbol{\pi}(\mathbf{y})|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k(\mathbf{y})^{\alpha_k} \tag{A.34a}$$

$$= \exp\left[\sum_k \alpha_k \log(\pi(y_k)) - \log(B(\alpha))\right] \tag{A.34b}$$

with sufficient statistics $\phi(y_i) = \log(\pi_i(y))$, natural parameters $w_i = \alpha_i$, base measure $h(y) = 1$ and normalizing constant $Z = \log(B(\alpha))$.

Laplace approximation of the softmax-transformed Dirichlet distribution

The inversion of the Laplace approximation in the softmax basis has been provided in (Hennig, 2010). The following is merely a summary. The mean and mode of the inverse-softmax Dirichlet are identical, i.e. $\boldsymbol{\pi}(\mathbf{y}) = \frac{\alpha}{\sum_i \alpha_i}$. For a visual confirmation of this fact, consider the figure of the Beta distribution in Subsection A.2.5. Additionally, the elements of $\mathbf{y}$ must sum to zero. These two constraints combined yield only one possible solution for $\boldsymbol{\mu}$.

$$\mu_k = \log\alpha_k - \frac{1}{K}\sum_{l=1}^{K}\log\alpha_l \tag{A.35}$$

Calculating the covariance matrix $\boldsymbol{\Sigma}$ is more complicated but detailed in the following. The logarithm of the Dirichlet is, up to additive constants

$$\log p_y(y|\alpha) = \sum_k \alpha_k \pi_k \tag{A.36}$$

Using π_k as the softmax of $\mathbf{y}$ as shown in Equation A.29 we can find the elements of the Hessian $\mathbf{L}$

$$L_{kl} = \hat{\alpha}(\delta_{kl}\hat{\pi}_k - \hat{\pi}_k\hat{\pi}_l) \tag{A.37}$$

where $\hat{\boldsymbol{\alpha}} := \sum_k \alpha_k$ and $\hat{\boldsymbol{\pi}} = \frac{\alpha_k}{\hat{\alpha}}$ for the value of $\boldsymbol{\pi}$ at the mode. Analytically inverting $\mathbf{L}$ is done via a lengthy derivation using the fact that we can write $\mathbf{L} = \boldsymbol{A} + \boldsymbol{X}\boldsymbol{B}\boldsymbol{X}^\top$ and inverting it with the Schur-complement. This process results in the inverse of the Hessian

$$L^{-1}_{kl} = \delta_{kl}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_l} - \frac{1}{K}\left(\sum_u^K \frac{1}{\alpha_u}\right)\right] \tag{A.38}$$

We are mostly interested in the diagonal elements, since we desire a sparse encoding for computational reasons and we otherwise needed to map a $K \times K$ covariance matrix to a $K \times 1$ Dirichlet parameter vector which would be a very overdetermined mapping. Note that K is a scalar

not a matrix. The diagonal elements of $\mathbf{\Sigma} = \mathbf{L}^{-1}$ can be calculated as

$$\Sigma_{kk} = \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_{l}^{k}\frac{1}{\alpha_l}. \tag{A.39}$$

To invert this mapping we transform Equation B.17 to

$$\alpha_k = e^{\mu_k}\prod_{l}^{K}\alpha_l^{1/K} \tag{A.40}$$

by applying the logarithm and re-ordering some parts. Inserting this into Equation B.21 and re-arranging yields

$$\prod_{l}^{K}\alpha_l^{1/K} = \frac{1}{\mathbf{\Sigma}_{kk}}\left[e^{-\mu}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_{u}^{K}e^{-\mu_u}\right] \tag{A.41}$$

which can be re-inserted into Equation B.22 to give

$$\alpha_k = \frac{1}{\Sigma_k k}\left(1 - \frac{2}{K} + \frac{e^{-\mu_k}}{K^2}\sum_{l}^{K}e^{-\mu_k}\right) \tag{A.42}$$

which is the final mapping. With Equations B.17 and B.21 we are able to map from Dirichlet to Gaussian and with Equation A.42 we are able to map the inverse direction.

The Bridge for the inverse-softmax transform

In summary we get the following forward and backward transformations between $\mathbf{y} \in \mathbb{R}^d$ and $\pi \in \mathbb{P}^d$.

$$\mu_k = \log\alpha_k - \frac{1}{K}\sum_{l=1}^{K}\log\alpha_l\,, \tag{A.43a}$$

$$\Sigma_{k\ell} = \delta_{k\ell}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_\ell} - \frac{1}{K}\sum_{u=1}^{K}\frac{1}{\alpha_u}\right]. \tag{A.43b}$$

The corresponding derivations require care because the Gaussian parameter space is evidently larger than that of the Dirichlet and not fully identified by the transformation. The pseudo-inverse of this map—as summarized above—was provided by Hennig, Stern, et al. (2012). It maps the Gaussian parameters to those of the Dirichlet as

$$\alpha_k = \frac{1}{\Sigma_{kk}}\left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2}\sum_{l=1}^{K}e^{-\mu_l}\right) \tag{A.44}$$

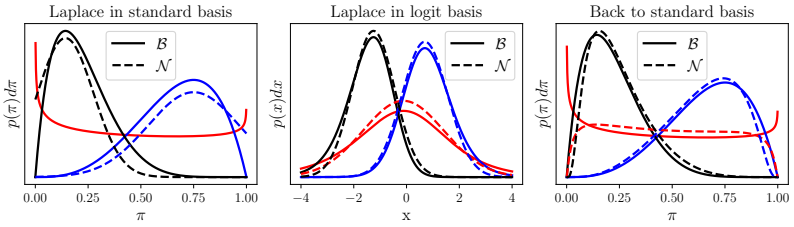


Figure A.10: inverse-softmax Dirichlet bridge. We show the Beta distribution as it is the 1-dimensional Version of the Dirichlet.

A.2.7 Wishart Distribution

Background: Kronecker-product

Kronecker-product: $A \otimes B \in \mathbb{R}^{(m_1 m_2)\times(n_1 n_2)}$ is defined by $(A \otimes B)_{(i-1)m_2+j,(k-1)n_2+l} = a_{ik}b_{jl} = (A \otimes B)_{(ij)(kl)}$.

Box-product: $A \boxtimes B \in \mathbb{R}^{(m_1 m_2)\times(n_1 n_2)}$ is defined by $(A \boxtimes B)_{(i-1)m_2+j,(k-1)n_1+l} = a_{il}b_{jk} = (A \boxtimes B)_{(ij)(kl)}$.

Symmetric Kronecker Product: $A \circledast B = A \otimes B + A \boxtimes B + B \otimes A + B \boxtimes A$

Standard Wishart distribution

the pdf of the Wishart is

$$\mathcal{W}(X; n, p, V) = \frac{1}{2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)} |\mathbf{X}|^{(n-p-1)/2} e^{-(1/2)\,\mathrm{tr}(\mathbf{V}^{-1}\mathbf{X})} \tag{A.45a}$$

$$= \exp\left[(n-p-1)/2 \log(|X|) - (1/2)\,\mathrm{tr}(\mathbf{V}^{-1}\mathbf{X}) - \log\left(2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)\right)\right] \tag{A.45b}$$

$$= \frac{1}{X^{\frac{1}{2}}} \exp\left[(n-p)/2 \log(|X|) - (1/2)\,\mathrm{tr}(\mathbf{V}^{-1}\mathbf{X}) - \log\left(2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)\right)\right] \tag{A.45c}$$

with exponential family values $h(X) = 1/X^{\frac{1}{2}}$, $\phi(X) = (\log(X), X)$, $w = ((n-p)/2, V^{-1})$ and $Z(n, p, V) = \log\left(2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)\right)$.

Laplace Approximation of the standard Wishart distribution

Using $\frac{\partial \det(X)}{\partial X} = \det(X)(X^{-1})^\top$ and $\frac{\partial}{\partial X} Tr(AX) = A^\top$ we can calculate the mode by setting the first derivative of the log-pdf to zero

$$\frac{\partial \log \mathcal{W}(X; n, p, V)}{\partial X} = \frac{(n-p-1)\det(X)(X^{-\top})}{2\det(X)} - \frac{V^{-\top}}{2}$$

$$\Rightarrow 0 = \frac{(n-p-1)X^{-1}}{2} - \frac{V^{-1}}{2}$$

$$\Leftrightarrow \frac{(n-p-1)X^{-1}}{2} = \frac{V^{-1}}{2}$$

$$\Leftrightarrow X = (n-p-1)V$$

Using the fact that the matrix is symmetric and $\frac{\partial X^{-1}}{\partial X} = -X^{-1} \otimes X^{-1}$ we get

$$\frac{\partial^2 \log f(X; n, p, V)}{\partial^2 X} = -\frac{(n-p-1)}{2} X^{-1} \otimes X^{-1}$$

Using $(\alpha A)^{-1} = \alpha^{-1}A^{-1}$, the linearity of the Kronecker product to pull out scalars and $X^{-1} \otimes X^{-1} = (X \otimes X)^{-1}$ to insert the mode and invert we

get:

$$
\begin{aligned}
-\frac{(n-p-1)}{2}X^{-1}\otimes X^{-1} &= -\frac{(n-p-1)}{2}\frac{1}{(n-p-1)}V^{-1}\otimes\frac{1}{(n-p-1)}V^{-1} \\
&= -\frac{1}{2(n-p-1)}(V\otimes V)^{-1} \\
\Rightarrow \Sigma &= 2(n-p-1)(V\otimes V)
\end{aligned}
$$

In summary, the Laplace approximation of a Wishart distribution in the standard basis is $\mathcal{N}(X;(n-p-1)V, 2(n-p-1)(V\otimes V))$, where the representation of the symmetric positive definite matrices has been changed from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n^2}$.

Laplace Approximation of the sqrtm-transformed Wishart distribution

For the derivations we will introduce a symmetry constraint since the matrix-sqrt of a symmetric matrix is also symmetric. The constraint is $\frac{1}{2\beta}||Y-Y^\top||_F^2$ with $\beta\to\infty$ such that the constraint becomes a dirac delta in the limit.

Using $\frac{\partial \det(Y)}{\partial Y} = \det(Y)(Y^{-1})^\top$ and $\frac{\partial}{\partial Y}\operatorname{tr}(AY^\top Y) = YA^\top + YA$ we can calculate the mode by setting the first derivative of the log-pdf to zero

$$
\frac{\partial \log \mathcal{W}_{\text{sqrtm}}(Y;n,p,V)}{\partial Y} = \frac{\partial}{\partial Y}\left[(n-p)\log(|\mathbf{Y}|) - (1/2)\operatorname{tr}(\mathbf{V}^{-1}\mathbf{Y}^\top\mathbf{Y}) - C - \frac{1}{2\beta}||Y-Y^\top||_F^2\right] \tag{A.46a}
$$

$$
= \frac{(n-p)\det(Y)(Y^{-\top})}{\det(Y)} - \frac{(YV^{-\top}+YV^{-1})}{2} - \frac{1}{\beta}(Y-Y^\top) \tag{A.46b}
$$

$$
= \frac{(n-p)\det(Y)(Y^{-\top})}{\det(Y)} - YV^{-1} - \frac{1}{\beta}(Y-Y^\top) \tag{A.46c}
$$

$$
\Rightarrow 0 = (n-p)Y^{-\top} - YV^{-1} \tag{A.46d}
$$

$$
\Rightarrow (n-p)Y^{-1} = YV^{-1} \tag{A.46e}
$$

$$
\Rightarrow Y = \text{sqrtm}\,((n-p)V) \tag{A.46f}
$$

Computing the second derivative by using $\frac{\partial}{\partial Y}Y^{-1} = -Y^{-\top}\otimes Y^{-1}$, $\frac{\partial (YA)_{kl}}{\partial Y_{ij}} = \delta_{ki}A_{jl} \Leftrightarrow \frac{\partial AY}{\partial Y} = I\otimes A$. To get the covariance matrix we multiply with

−1 and invert the matrix.

$$\frac{\partial^2 \log \mathcal{W}_{\text{sqrtm}}(Y; n, p, V)}{\partial^2 Y} = \frac{\partial}{\partial Y}\left[(n-p)Y^{-\top} - YV^{-1} - \frac{1}{\beta}(Y - Y^\top)\right] \tag{A.47a}$$

$$= -(n-p)(Y^{-\top} \otimes Y^{-1}) - I \otimes V^{-1} - \frac{1}{\beta}(I \otimes I - I \boxtimes I) \tag{A.47b}$$

$$\overset{\text{mode}}{\Rightarrow} -(n-p)\left[\sqrt{\frac{1}{(n-p)}}V^{-\frac{1}{2}} \otimes \sqrt{\frac{1}{(n-p)}}V^{-\frac{1}{2}}\right] - I \otimes V^{-1} - \frac{1}{\beta}(I \otimes I - I \boxtimes I) \tag{A.47c}$$

$$= -\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1} + \frac{1}{\beta}(I \otimes I - I \boxtimes I)\right) \tag{A.47d}$$

$$\Leftrightarrow \frac{\partial^2 \log \mathcal{W}_{\text{sqrtm}}(Y; n, p, V)}{\partial Y_{ik} \partial Y_{jl}} = -\left(V_{ik}^{-\frac{1}{2}} V_{jl}^{-\frac{1}{2}} + \delta_{ik} V_{jl}^{-1} + \frac{1}{\beta}(\delta_{ik}\delta_{jl} - \delta_{il}\delta_{jk})\right) \tag{A.47e}$$

In general, any matrix can be split into a symmetric and a skew-symmetric matrix, $A = \frac{1}{2}(A + A^\top) + \frac{1}{2}(A - A^\top)$. For both parts we can define projection operators Γ and Δ with

$$\Gamma A = \frac{1}{2}(A + A^\top), \Gamma_{(ij),(kl)} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{kj}) \tag{A.48a}$$

$$\Delta A = \frac{1}{2}(A - A^\top), \Delta_{(ij),(kl)} = \frac{1}{2}(\delta_{ik}\delta_{jl} - \delta_{il}\delta_{kj}) \tag{A.48b}$$

(A.48c)

For these projection operators it holds that $\Gamma\Gamma = \Gamma$, $\Delta\Delta = \Delta$, $\Delta\Gamma = \Gamma\Delta = \mathbf{0}$, and $\Delta + \Gamma = \Gamma + \Delta = \mathbf{I}$. Furthermore, it holds that

$$A \otimes B = (\Gamma + \Delta)(A \otimes B)(\Gamma + \Delta)^\top \tag{A.49a}$$

$$= \Gamma(A \otimes B)\Gamma^\top + \Gamma(A \otimes B)\Delta^\top + \Delta(A \otimes B)\Gamma^\top + \Delta(A \otimes B)\Delta^\top \tag{A.49b}$$

In the context of the Wishart this means

$$\Gamma\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1} + \frac{1}{\beta}\Delta\right)\Gamma^\top + \Gamma\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1} + \frac{1}{\beta}\Delta\right)\Delta^\top \tag{A.50}$$

$$+ \Delta\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1} + \frac{1}{\beta}\Delta\right)\Gamma^\top + \Delta\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1} + \frac{1}{\beta}\Delta\right)\Delta^\top \tag{A.51}$$

$$= \left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1}\right) + 0 + 0 + 0 \tag{A.52}$$

because all the terms involving Δ yield 0 probability mass on symmetric matrices due to the symmetry constraint.

In practice, e.g. for the computation of the KL-divergence, the symmetric Kronecker product is suboptimal for computation because it is only invertible in $\mathbb{R}_S^d$ but not in $\mathbb{R}^d$. Thus we chose to compute the covariance

matrix with

$$\Sigma = \left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1}\right)^{-1} \tag{A.53}$$

and adapt the pdf of the normal distribution to the space of symmetric matrices

$$\mathcal{N}(X, \mu, \Sigma) = \frac{\frac{1}{2}d(d+1)}{d^2}(2\pi)^{-\frac{2}{d}} \det(U^\top \Sigma U)^{-\frac{1}{2}} \exp -1/2(X-\mu)^\top \Sigma^{-1}(X-\mu) \tag{A.54}$$

where $\frac{\frac{1}{2}d(d+1)}{d^2}$ is the ratio of degrees of freedom between a symmetric and asymmetric matrix, U are all eigenvectors of the symmetry projection Γ and d is the number of dimensions.

We can simplify the Hessian further to make it computationally easier to invert it.

$$\ldots = -\Gamma \left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1}\right) \Gamma^\top \tag{A.55a}$$

$$= -\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}} + I_p \otimes V^{-1}\right) \tag{A.55b}$$

$$\overset{-1}{=} \left(I_{p^2} + V^{\frac{1}{2}} \otimes V^{-\frac{1}{2}}\right)\left(V^{-\frac{1}{2}} \otimes V^{-\frac{1}{2}}\right) \tag{A.55c}$$

$$\Rightarrow \Sigma = \left(V^{\frac{1}{2}} \otimes V^{\frac{1}{2}}\right)\left(I_{p^2} + V^{\frac{1}{2}} \otimes V^{-\frac{1}{2}}\right)^{-1} \tag{A.55d}$$

which could be inverted more easily using equation 5 of Stegle et al., 2011. However, if you aren't interested in the covariance matrix itself, but e.g. only want to compute the Gaussian, it makes sense to compute everything with the inverse covariance matrix to save the computational effort of an inversion.

The Bridge for sqrtm-tranform

We use $\mu = ((n-p)V)^{\frac{1}{2}} \Leftrightarrow \mu^2 = (n-p)V \Leftrightarrow V = \frac{\mu^2}{(n-p)}$. Remember that μ is reshaped to be the same size as V even though we usually think of it in vector-form.

$$\mu = ((n-p)V)^{\frac{1}{2}} \tag{A.56a}$$

$$\Sigma = \left(V^{\frac{1}{2}} \otimes V^{\frac{1}{2}}\right)\left(I_{p^2} + V^{\frac{1}{2}} \otimes V^{-\frac{1}{2}}\right)^{-1} \tag{A.56b}$$

$$V = ** \tag{A.56c}$$

$$n = ** \tag{A.56d}$$

The resulting Σ cannot be easily solved for V and thus there are three ways to choose a matching from μ, Σ to n, V. a) We can assume that Σ has to have the same structure as shown above, i.e. a product of Kronecker products. Then we can compute V and insert it in the equation for μ to get n. b) We can treat V or n as a free parameter and compute our solution solely from the equation of μ. c) We could just use the logm-transform which is has good inversions for both n and V.

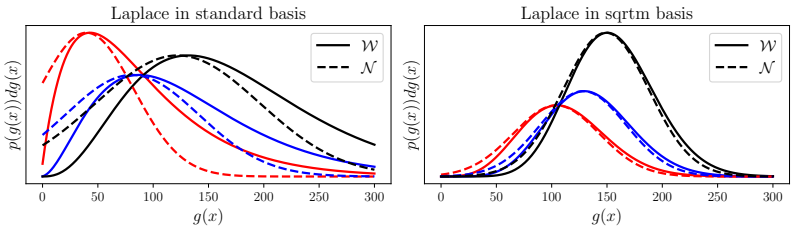


Figure A.11: sqrtm-bridge for the Wishart distribution.

Logm-Transformed Wishart distribution

We transform the distribution with $g(X) = \text{logm}(X)$, i.e. $X(Y) = g^{-1}(X) = \text{expm}(Y)$, where $\text{expm}(Y)$ is the matrix exponential and $\text{logm}(Y)$ is the matrix logarithm of Y. The new pdf becomes

$$\mathcal{W}_{\text{logm}}(Y; n, p, V) = \frac{1}{2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)} |\text{expm}\,\mathbf{Y}|^{(n-p-1)/2} e^{-(1/2)\,\text{tr}(\mathbf{V}^{-1}\text{expm}\,\mathbf{Y})} \cdot |\,\text{expm}\,Y| \tag{A.57a}$$

$$= \frac{1}{2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)} |\text{expm}\,\mathbf{Y}|^{(n-p+1)/2} e^{-(1/2)\,\text{tr}(\mathbf{V}^{-1}\text{expm}\,\mathbf{Y})} \tag{A.57b}$$

$$= \exp\left[C + \frac{(n-p+1)}{2}\log(|\text{expm}\,\mathbf{Y}|) - \frac{1}{2}\,\text{tr}(\mathbf{V}^{-1}\text{expm}\,\mathbf{Y})\right] \tag{A.57c}$$

with exponential family values $h(Y) = Y^{\frac{1}{2}}, \phi(Y) = (Y, \text{expm}\,Y), w = (n-p), V^{-1}$ and $Z(n,p,V) = \log\left(2^{np/2}\,|\mathbf{V}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)\right)$.

Laplace Approximation of the logm-transformed Wishart distribution

The matrix logarithm of a symmetric matrix yields a symmetric matrix again. Thus the symmetry-constraint of the matrix-sqrt transformation also applies here. For brevity we have decided not to include it in the derivation and point it out at selected parts of the text.

To compute the first derivative we use the following

$$\frac{\partial \log(\det(\text{expm}(Y)))}{\partial Y} = \frac{\partial \log(\det(\text{expm}(Y)))}{\partial \det(\text{expm}(Y))} \cdot \frac{\partial \det(\text{expm}(Y))}{\partial \text{expm}(Y)} \cdot \frac{\partial \text{expm}(Y)}{\partial Y} \tag{A.58a}$$

$$= \frac{1}{\det(\text{expm}(Y))} \cdot \det(\text{expm}(Y))\,\text{expm}(Y)^{-\top} \cdot \text{expm}(Y) \tag{A.58b}$$

$$= I_p \tag{A.58c}$$

where I_p is the identity matrix of size p and we use the fact that the matrix logarithm of a symmetric matrix is symmetric, implying $\text{expm}(Y)^{-\top} =$

$\exp(Y)^{-1}$. With this we get the first derivative

$$\frac{\partial \log W_{log}}{\partial Y} = \frac{\partial}{\partial Y}\left[C + \frac{(n-p+1)}{2}\log(|\exp \mathbf{Y}|) - \frac{1}{2}\operatorname{tr}(\mathbf{V}^{-1}\exp \mathbf{Y})\right] \tag{A.59a}$$

$$= \frac{(n-p+1)}{2} I_p - \frac{1}{2} V^{-1}\exp \mathbf{Y} \tag{A.59b}$$

By setting this to zero we get a mode of

$$0 = \frac{(n-p+1)}{2} I_p - \frac{1}{2} V^{-1}\exp \mathbf{Y} \tag{A.60a}$$

$$\Leftrightarrow (n-p+1) I_p = V^{-1}\exp \mathbf{Y} \tag{A.60b}$$

$$\Leftrightarrow Y = \operatorname{logm}((n-p+1)V) \tag{A.60c}$$

For the second derivative we use the fact that

$$\frac{\partial (B\exp(Y))_{kl}}{\partial Y_{ij}} = \delta_{jl}(B\operatorname{exmp}(Y))_{ki} \tag{A.61a}$$

$$\Leftrightarrow \frac{\partial B\exp(Y)}{\partial Y} = \left(B\exp(Y) \otimes I_p\right) \tag{A.61b}$$

yielding

$$\frac{\partial^2 \log \mathcal{W}_{\text{logm}}}{\partial^2 Y} = \frac{\partial \log \mathcal{W}_{\text{logm}}}{\partial Y}\left[\frac{(n-p+1)}{2} I_p - \frac{1}{2} V^{-1}\exp \mathbf{Y}\right] \tag{A.62a}$$

$$= -\frac{1}{2}(V^{-1}\exp \mathbf{Y} \otimes I_p) \tag{A.62b}$$

$$\overset{\text{mode}}{\Rightarrow} -\frac{1}{2}((n-p+1)V^{-1}V \otimes I_p) \tag{A.62c}$$

$$= -\frac{(n-p+1)}{2}(I_p \otimes I_p) \tag{A.62d}$$

$$\Leftrightarrow \Sigma = \frac{2}{n-p+1} I_{p^2} \tag{A.62e}$$

where I_{p^2} is an Identity matrix of size p^2. With the symmetry constraint, we would get $(I_p \circledast I_p)$ instead of $(I_p \otimes I_p)$.

The Bridge for logm-tranform

μ and Σ are already given by the Laplace approximation. Inverting the mode yields an estimate for V.

$$\mu = \operatorname{logm}((n-p+1)V) \Leftrightarrow \exp(\mu) = (n-p+1)V \Leftrightarrow V = \frac{\exp(\mu)}{(n-p+1)} \tag{A.63}$$

where μ and V are reshaped to a matrix of size $p \times p$. The parameter n can be derived from the equation of Σ by

$$\Sigma = \frac{2}{n-p+1} I_{p^2} \tag{A.64}$$

$$\Leftrightarrow \operatorname{tr}(\Sigma) = \frac{2p}{n-p+1} p^2 \tag{A.65}$$

$$\Leftrightarrow n = \frac{2p^2}{\operatorname{tr}(\Sigma)} + p - 1 \tag{A.66}$$

In summary this yields

$$\mu = \operatorname{logm}((n-p+1)V) \tag{A.67a}$$

$$\Sigma = \frac{2}{n-p+1} (I_p \otimes I_p)^{-1} \tag{A.67b}$$

$$V = \frac{\operatorname{expm}(\mu)}{(n-p+1)} \tag{A.67c}$$

$$n = \frac{2p^2}{\operatorname{tr}(\Sigma)} + p - 1 \tag{A.67d}$$

Similar to the sqrtm-transformation, we can adapt the normal distribution to the space of symmetric matrices only and compute with the unconstrained version (see previous subsection).

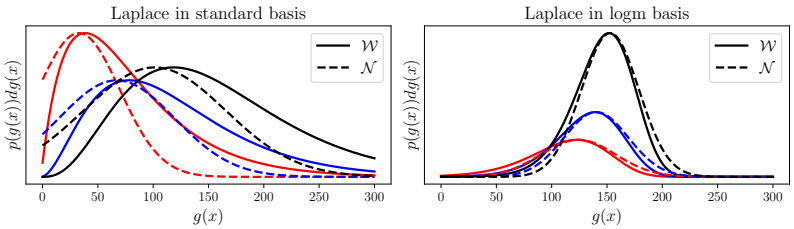


Figure A.12: logm-bridge for the Wishart distribution

A.2.8 Inverse Wishart Distribution

Standard Inverse Wishart distribution

The pdf of the inverse Wishart is

$$\mathcal{IW}_{\mathbf{x}}(\mathbf{x}; \mathbf{\Psi}, \nu) = \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2} \Gamma_p(\frac{\nu}{2})} |\mathbf{x}|^{-(\nu+p+1)/2} e^{-\frac{1}{2}\operatorname{tr}(\mathbf{\Psi}\mathbf{x}^{-1})} \tag{A.68a}$$

$$= \exp\left[-(\nu+p+1)/2 \log(|x|) - \frac{1}{2}\operatorname{tr}(\Psi x^{-1}) + \log\left(\frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})}\right) \right] \tag{A.68b}$$

with exponential family values $h(X) = 1/X^{\frac{1}{2}}$, $\phi(X) = (\log(X), X^{-1})$, $w = (-(\nu+p)/2, \Psi)$ and $Z(n,p,V) = -\log(\frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})})$.

Laplace Approximation of the standard inverse Wishart distribution

Using $\frac{\partial \det(X)}{\partial X} = \det(X)(X^{-1})^\top$ and $\frac{\partial}{\partial X}\operatorname{tr}(AX)^{-1} = (X^{-1}AX^{-1})^\top$ we can calculate the mode by setting the first derivative of the log-pdf to zero:

$$\frac{\partial \log \mathcal{IW}_{\mathbf{X}}(\mathbf{X};\mathbf{\Psi},\nu)}{\partial X} = \frac{-(\nu+p+1)\det(X)X^{-\top}}{2\det(X)} + \frac{(X^{-1}\Psi X^{-1})^\top}{2} \tag{A.69a}$$

$$= \frac{-(\nu+p+1)X^{-\top}}{2} + \frac{(X^{-1}\Psi X^{-1})^\top}{2} \tag{A.69b}$$

$$\Rightarrow 0 = \frac{-(\nu+p+1)X^{-\top}}{2} + \frac{(X^{-1}\Psi X^{-1})^\top}{2} \tag{A.69c}$$

$$\Leftrightarrow (\nu+p+1)X^{-1} = X^{-1}\Psi X^{-1} \tag{A.69d}$$

$$\Leftrightarrow (\nu+p+1) = X^{-1}\Psi \tag{A.69e}$$

$$\Leftrightarrow X = \frac{1}{\nu+p+1}\Psi \tag{A.69f}$$

Using

$$\frac{\partial (XBX)_{kl}}{\partial X_{ij}} = \delta_{ki}(BX)_{lj} + \delta_{lj}(XB)_{ki} \tag{A.70a}$$

$$\frac{\partial X^{-1}}{\partial X} = -(X^{-1}\otimes X^{-1}) \tag{A.70b}$$

$$\frac{\partial (X^{-1}BX^{-1})}{\partial X} = \frac{\partial (X^{-1}BX^{-1})}{\partial X^{-1}}\frac{\partial X^{-1}}{\partial X} = -[\delta_{ki}(BX^{-1})_{lj} + \delta_{lj}(X^{-1}B)_{ki}](X^{-1}\otimes X^{-1}) \tag{A.70c}$$

$$(AB)^{-1} = B^{-1}A^{-1} \tag{A.70d}$$

we can get the covariance matrix by inverting the Hessian and multiplying with -1.

$$\frac{\partial^2 \log f_{\mathbf{X}}(\mathbf{X};\mathbf{\Psi},\nu)}{\partial^2 X} = \frac{(\nu+p+1)}{2}(X^{-1}\otimes X^{-1})^\top - \frac{[\delta_{ki}(\Psi X^{-1})_{lj}+\delta_{lj}(X^{-1}\Psi)_{ki}]}{2}(X^{-1}\otimes X^{-1})^\top \tag{A.71a}$$

$$= \left\{\frac{(\nu+p+1)}{2}I_{n^2} - \frac{[\delta_{ki}(\Psi X^{-1})_{lj}+\delta_{lj}(X^{-1}\Psi)_{ki}]}{2}\right\}(X^{-1}\otimes X^{-1})^\top \tag{A.71b}$$

$$\overset{\text{mode}}{=} \left\{\frac{(\nu+p+1)}{2}I_{n^2} - \frac{[\delta_{ki}((\nu+p+1)\Psi\Psi^{-1})_{lj}+\delta_{lj}((\nu+p+1)\Psi^{-1}\Psi)_{ki}]}{2}\right\}(\nu+p+1)^2(\Psi\otimes\Psi)^{-\top} \tag{A.71c}$$

$$= \left\{\frac{(\nu+p+1)}{2}I_{n^2} - \underbrace{\frac{[\delta_{ki}((\nu+p+1)I_n)_{lj}+\delta_{lj}((\nu+p+1)I_n)_{ki}]}{2}}_{=(\nu+p+1)I_{n^2}}\right\}(\nu+p+1)^2(\Psi\otimes\Psi)^{-\top} \tag{A.71d}$$

$$= -\underbrace{\frac{1}{2}(\nu+p+1)I_{n^2}}_{A}\underbrace{(\nu+p+1)^2(\Psi\otimes\Psi)^{-\top}}_{B} \tag{A.71e}$$

$$\overset{\text{invert}}{\Rightarrow} -\frac{1}{(\nu+p+1)^2}(\Psi\otimes\Psi)^\top\frac{2}{(\nu+p+1)}I_{n^2} \tag{A.71f}$$

$$\overset{\cdot^{-1}}{\Rightarrow} \Sigma = \frac{2}{(\nu+p+1)^3}(\Psi\otimes\Psi)^\top \tag{A.71g}$$

where I_n and I_n^2 are the identity matrix of size n and n^2 respectively. We can also ignore the transpose since we are dealing with symmetric positive definite matrices when it comes to the inverse Wishart distribution. This yields a multivariate Gaussian of the form $\mathcal{N}\left(X;\mu=\frac{1}{\nu+p+1}\Psi,\Sigma=\frac{2}{(\nu+p+1)^3}(\Psi\otimes\Psi)\right)$.

Logm-Transformed inverse Wishart distribution

Similar to the Wishart derivation, we compute all derivations with a symmetry-constraint. For brevity we omit the constraint but point it out where it makes a difference.

We transform the distribution with $g(X) = \text{logm}(X)$, i.e. $X(Y) = g^{-1}(X) = \text{expm}(Y)$, where $\text{expm}(Y)$ is the matrix exponential. The new

pdf becomes

$$\mathcal{IW}_{\text{logm}}(\mathbf{\Upsilon};\mathbf{\Psi},\nu) = \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})} |\text{expm}\,\mathbf{\Upsilon}|^{-(\nu+p+1)/2} e^{-\frac{1}{2}\text{tr}(\mathbf{\Psi}(\text{expm}\,\mathbf{\Upsilon})^{-1})} \cdot |\,\text{expm}(\mathbf{\Upsilon})| \tag{A.72a}$$

$$= \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})} |\text{expm}\,\mathbf{\Upsilon}|^{-(\nu+p-1)/2} e^{-\frac{1}{2}\text{tr}(\mathbf{\Psi}(\text{expm}\,\mathbf{\Upsilon})^{-1})} \tag{A.72b}$$

$$= \exp\left[C - (\nu+p-1)/2 \log|\text{expm}\,\mathbf{\Upsilon}| - -\frac{1}{2}\text{tr}(\mathbf{\Psi}\,\text{expm}(-\mathbf{\Upsilon}))\right] \tag{A.72c}$$

with exponential family values $h(Y) = Y^{\frac{1}{2}}$, $\phi(Y) = (\log(\det(\text{expm}(Y))), \text{expm}(Y))$, $w = ((\nu+p)/2, \Psi)$ and $Z(n,p,V) = -\log\left(\frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})}\right)$

Laplace Approximation of the logm-transformed inverse Wishart distribution

For the first derivative we use the same concepts as for the Wishart in Subsection A.2.7.

$$\frac{\partial \log \mathcal{IW}_{logm}}{\partial Y} = \frac{\partial}{\partial Y} - (\nu+p-1)/2 \log|\text{expm}\,\mathbf{\Upsilon}| - -\frac{1}{2}\text{tr}(\mathbf{\Psi}\,\text{expm}(-\mathbf{\Upsilon})) \tag{A.73a}$$

$$= -\frac{(\nu+p-1)}{2} I_p + \frac{1}{2}\mathbf{\Psi}\,\text{expm}(-\mathbf{\Upsilon}) \tag{A.73b}$$

which yields the mode by setting it to zero and solving for Y.

$$0 = -\frac{(\nu+p-1)}{2} I_p + \frac{1}{2}\mathbf{\Psi}\,\text{expm}(-\mathbf{\Upsilon}) \tag{A.74a}$$

$$\Leftrightarrow (\nu+p-1) I_p = \mathbf{\Psi}(\text{expm}(\mathbf{\Upsilon}))^{-1} \tag{A.74b}$$

$$\Leftrightarrow Y = \text{logm}\left(\frac{\mathbf{\Psi}}{\nu+p-1}\right) \tag{A.74c}$$

For the second derivative we also use the same concepts as for the Wishart and get

$$\frac{\partial^2 \log \mathcal{IW}_{logm}}{\partial^2 Y} = \frac{\partial}{\partial Y}\left[-\frac{(\nu+p-1)}{2} + \frac{1}{2}\mathbf{\Psi}\,\text{expm}(-\mathbf{\Upsilon})\right] \tag{A.75a}$$

$$= -\frac{1}{2}\left[\mathbf{\Psi}\,\text{expm}(\mathbf{\Upsilon})^{-1} \otimes I_p\right] \tag{A.75b}$$

$$\overset{\text{mode}}{\Rightarrow} -\frac{1}{2}\left[\mathbf{\Psi}\frac{\mathbf{\Psi}^{-1}}{(\nu+p-1)} \otimes I_p\right] \tag{A.75c}$$

$$= -\frac{1}{2(\nu+p-1)} I_{p\times p} \tag{A.75d}$$

$$\Rightarrow \Sigma = 2(\nu+p-1) I_{p\times p} \tag{A.75e}$$

With the symmetry-constraint $I_{p\times p}$ becomes $(I \otimes I)^{-1}$. This yields a multivariate Normal distribution $\mathcal{N}\left(Y; \mu = \text{logm}\left(\frac{\mathbf{\Psi}}{n+p-1}\right), \Sigma = 2(\nu+p-1) I_{p\times p}\right)$

which is constraint to symmetric matrices and its normalization is thus adapted as described in Equation A.54.

The Bridge for the logm-transformed inverse Wishart distribution

We get μ and Ψ from the Laplace approximation and determine V by inverting μ

$$\mu = \operatorname{logm}\left(\frac{\mathbf{\Psi}}{n+p-1}\right) \Leftrightarrow \operatorname{expm}(\mu) = \frac{\mathbf{\Psi}}{n+p-1} \Leftrightarrow \mathbf{\Psi} = \operatorname{expm}(\mu)(n+p-1) \tag{A.76a}$$

Additionally we can get ν from the equation for Σ by

$$\Sigma = 2(\nu+p-1)I_{p\times p} \tag{A.77}$$
$$\Leftrightarrow \operatorname{tr}(\Sigma) = 2(\nu+p-1)p^2 \tag{A.78}$$
$$\Leftrightarrow \nu = \frac{\operatorname{tr}(\Sigma)}{2p^2} - p + 1 \tag{A.79}$$

In summary we have

$$\mu = \operatorname{logm}\left(\frac{\mathbf{\Psi}}{n+p-1}\right) \tag{A.80a}$$
$$\Sigma = 2(\nu+p-1)I_{p\times p} \tag{A.80b}$$
$$\mathbf{\Psi} = (\nu+p-1)\operatorname{expm}(\mu) \tag{A.80c}$$
$$\nu = \frac{\operatorname{tr}(\Sigma)}{2p^2} - p + 1 \tag{A.80d}$$

where $\mathbf{\Psi}$ and μ are reshaped to a $p \times p$ matrix. For this derivation we assume to use only symmetric matrices. If this constraint is lifted, we have to replace $I_{p\times p}$ with $I_p \otimes I_p$.

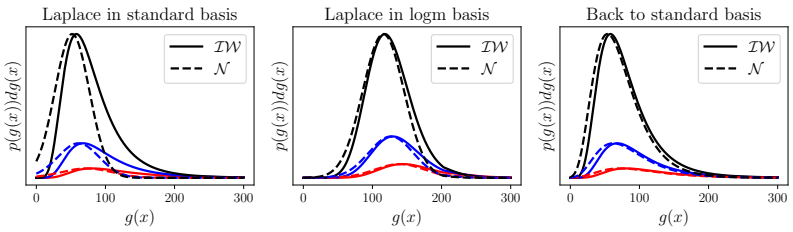


Figure A.13: logm-bridge for the inverse Wishart distribution.

Sqrtm-Transformed inverse Wishart distribution

Similar to the derivation of the sqrtm-transformation for the Wishart (see Subsection A.2.7, the inverse Wishart also has a symmetry constraint. For brevity, we omit it for the derivation and point out where it makes a difference.

We transform the distribution with $g(X) = \operatorname{sqrtm}(X) = X^{\frac{1}{2}}$, i.e. $X(Y) = g^{-1}(X) = Y^2$, where $\operatorname{sqrtm}(Y)$ is the square root of the matrix. We

choose the principle square root of the positive definite matrix X, i.e. $X = Y^\top Y = YY$. The new pdf becomes

$$\mathcal{IW}_{\text{sqrtm}}(\mathbf{Y}; \mathbf{\Psi}, \nu) = \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})} \left|\mathbf{Y}^2\right|^{-(\nu+p+1)/2} e^{-\frac{1}{2}\operatorname{tr}(\mathbf{\Psi}\mathbf{Y}^\top\mathbf{Y}^{-1})} |2Y| \tag{A.81a}$$

$$= \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})} |\mathbf{Y}|^{-(\nu+p+1)} e^{-\frac{1}{2}\operatorname{tr}(\mathbf{\Psi}\mathbf{Y}^\top\mathbf{Y}^{-1})} 2^p |Y| \tag{A.81b}$$

$$= \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})} |\mathbf{Y}|^{-(\nu+p)} e^{-\frac{1}{2}\operatorname{tr}(\mathbf{\Psi}\mathbf{Y}^{-2})} 2^p \tag{A.81c}$$

$$= \exp\left[-(\nu+p)\log(|Y|) - \frac{1}{2}\operatorname{tr}(\Psi(Y^\top Y)^{-1}) + \log(C)\right] \tag{A.81d}$$

with $h(Y) = 1, \phi(Y) = (\log(|Y|), Y^{-2}), w = (-(\nu+p), \Psi)$ and $Z(w) = \log\left(\frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\nu p/2}\Gamma_p(\frac{\nu}{2})}\right)$.

Laplace Approximation of the sqrtm-transformed inverse Wishart distribution

Using

$$\frac{\partial \det(Y)}{\partial Y} = \det(Y)(Y^{-1})^\top \frac{\partial \operatorname{tr}(\Psi(Y^\top Y)^{-1})}{\partial Y} = 2\Psi Y^{-3} \tag{A.82a}$$

we can calculate the mode by setting the derivative of the log-pdf to zero:

$$\frac{\partial \log \mathcal{IW}_{\text{sqrtm}}(Y, \Psi, \nu)}{\partial Y} = \frac{-(\nu+p)\det(Y)Y^{-\top}}{\det(Y)} + 2\Psi Y^{-3} \tag{A.83a}$$

$$= -(\nu+p)Y^{-\top} + 2\Psi Y^{-3} \tag{A.83b}$$

$$\Rightarrow 0 = -(\nu+p)Y^{-\top} + 2\Psi Y^{-3} \tag{A.83c}$$

$$\Leftrightarrow (\nu+p)Y^{-\top} = 2\Psi Y^{-3} \tag{A.83d}$$

$$\Leftrightarrow Y = \text{sqrtm}\left(\frac{1}{(\nu+p)}\Psi\right) \tag{A.83e}$$

Using

$$\frac{\partial X^{-1}}{\partial X} = -X^{-1} \otimes X^{-1} \tag{A.84a}$$

$$\frac{\partial Y X^{-3}}{\partial X} = \frac{\partial \Psi X^{-3}}{\partial X^{-3}} \frac{\partial X^{-3}}{\partial X^{-1}} \frac{\partial X^{-1}}{\partial X} \tag{A.84b}$$

$$= -(\Psi \otimes I)\left(I \otimes X^{-2} + X^{-1} \otimes X^{-1} + X^{-2} \otimes I\right)\left(X^{-1} \otimes X^{-1}\right) \tag{A.84c}$$

We can calculate the Hessian and by multiplying with -1 and inverting it we get the covariance matrix.

$$\frac{\partial^2 \log \mathcal{IW}_{\text{sqrtm}}(Y, \Psi, \nu)}{\partial^2 Y} = \frac{\partial}{\partial Y} - (\nu + p)Y^{-\top} + \Psi Y^3 \tag{A.85a}$$

$$= (\nu + p)(Y^{-1} \otimes Y^{-1}) - (\Psi \otimes I)\left(I \otimes Y^{-2} + Y^{-1} \otimes Y^{-1} + Y^{-2} \otimes I\right)\left(Y^{-1} \otimes Y^{-1}\right) \tag{A.85b}$$

$$\overset{\text{mode}}{=} (\nu + p)(\sqrt{(\nu + p)}\Psi^{-\frac{1}{2}} \otimes \sqrt{(\nu + p)}\Psi^{-\frac{1}{2}}) \tag{A.85c}$$

$$- (\Psi \otimes I)\left(I \otimes (\nu + p)\Psi^{-1} + \sqrt{(\nu + p)}\Psi^{-\frac{1}{2}} \otimes \sqrt{(\nu + p)}\Psi^{-\frac{1}{2}} + (\nu + p)\Psi^{-1} \otimes I\right) \tag{A.85d}$$

$$\cdot \left(\sqrt{(\nu + p)}\Psi^{-\frac{1}{2}} \otimes \sqrt{(\nu + p)}\Psi^{-\frac{1}{2}}\right) \tag{A.85e}$$

$$= (\nu + p)^2\left(\Psi^{-\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}}\right) - (\nu + p)^2\left(\Psi \otimes \Psi^{-1} + \Psi^{\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}} + I_{p^2}\right)\left(\Psi^{-\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}}\right) \tag{A.85f}$$

$$= -(\nu + p)^2\left(\Psi \otimes \Psi^{-1} + \Psi^{\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}}\right)\left(\Psi^{-\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}}\right) \tag{A.85g}$$

$$= -(\nu + p)^2\left(\Psi^{\frac{1}{2}} \otimes \Psi^{-\frac{1}{2}} + I_{p^2}\right)\left(I_p \otimes \Psi^{-1}\right) \tag{A.85h}$$

$$\Leftrightarrow \Sigma = \frac{1}{(\nu + p)^2}\left(I_p \otimes \Psi\right)\left(\Psi^{\frac{1}{2}} \otimes \Psi^{\frac{1}{2}} + I_p\right)^{-1} \tag{A.85i}$$

which could be inverted more easily using equation 5 of Stegle et al. (2011). For further notes on efficient computation see Equation A.54 and surrounding text. With the symmetry constraints, all $\otimes$ become $\otimes$.

The Bridge for the sqrtm-transformed inverse Wishart distribution

The resulting Σ cannot be easily solved for Ψ and thus there are three ways to choose a matching from μ, Σ to ν, Ψ. a) We can assume that Σ has to have the same structure as shown above, i.e. a product of Kronecker products. Then we can compute Ψ and insert it in the equation for μ to get ν. b) We can treat Ψ or ν as a free parameter and compute our solution solely from the equation of μ. c) We could just use the logm-transform which is has good inversions for both ν and Ψ.

In summary we have

$$\mu = \text{sqrtm}\left(\frac{1}{(\nu + p)}\Psi\right) \tag{A.86}$$

$$\Sigma = \frac{1}{(\nu + p)^2}\left(I_p \otimes \Psi\right)\left(\Psi^{\frac{1}{2}} \otimes \Psi^{\frac{1}{2}} + I_p\right)^{-1} \tag{A.87}$$

$$\Psi = ** \tag{A.88}$$

$$n = ** \tag{A.89}$$

where $**$ is described above.

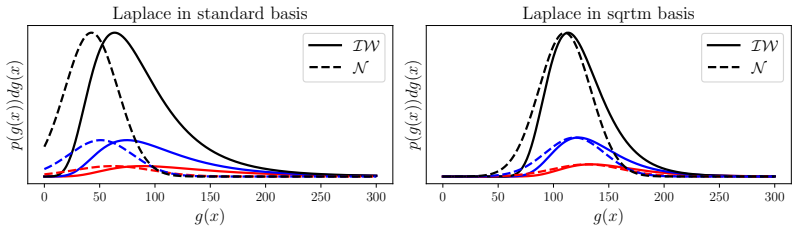


Figure A.14: sqrtm-bridge for the inverse Wishart distribution.

A.3 Experimental details

A.3.1 Distances

Parameters for KL and MMD

As already discussed in the main text, the 10 (pairs of) parameters used to compute the KL-divergence and MMD are chosen such that they start from a small value which doesn't yield a valid Laplace approximation in the standard base and end with a larger value that is approximately Gaussian in the standard base since this is a feature many exponential families share. In Table A.1 you can find the exact parameters used for the figures in Table 3.8.

Table A.1: Parameters for the computation of KL-divergence and MMD.

Distribution	Parameter 1	Parameter 2
Exponential	$\lambda = [1, 2, ..., 10]$	-
Gamma	$\alpha = [0.5, 1.5, ..., 9.5]$	$\lambda = [0.5, 1, ..., 5.5]$
Inv. Gamma	$\alpha = [1, ..., 10]$	$\lambda = [0.5, 1, ..., 5.5]$
Chi-squared	$k = [1, ...10]$	-
Beta	$\alpha = [0.7, 1.2, ..., 5.2]$	$\beta = [0.8, 1.05, ..., 3.05]$
Dirichlet	$\alpha = [0.8, 0.8, 0.8] * [1.5, 1, 0.75] * i$	for i in 1,...,10
Wishart	Discussed below	
Inv. Wishart	Discussed below	

Dirichlet KL computation

Computing the KL-divergence for the Dirichlet is tricky because the Gaussian and the Dirichlet are defined on different domains. Computing the KL-divergence between the Gaussian and the Dirichlet in the inverse-softmax basis is also complicated because there is no inverse-softmax transformation that could transform samples as the softmax is not a bijective function. To solve this problem we transform the Gaussian into the probability domain. Since the simplex has one degree of freedom less than $\mathbb{R}^K$ we have to update μ and Σ with a rank-1 constraint as already discussed in Hennig, 2010.

$$\bar{\mu} = \mu - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \mu}{\mathbf{1}^\top \Sigma \mathbf{1}} \tag{A.90}$$

$$\bar{\Sigma} = \Sigma - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \Sigma}{\mathbf{1}^\top \Sigma \mathbf{1}} \tag{A.91}$$

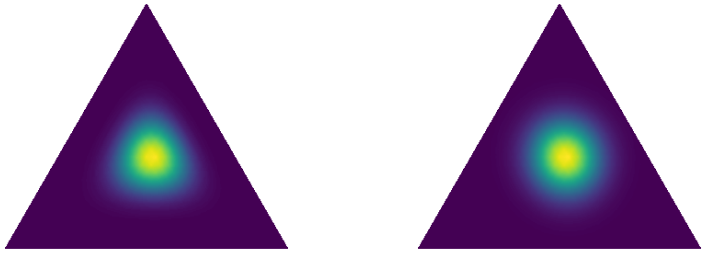


Figure A.15: Comparing original Dirichlet (left) with the fake softmax Gaussian descibed in the text (right).

Then we project the variable x into a $K-1$ dimensional subspace with $U = Ax$ where A_{ij} for $i = 1, ..., K-1$ yielding

$$p(u) = \mathcal{N}(u|A\bar{\mu}, A\bar{\Sigma}A^\top)|\det \frac{\partial u}{\partial y}| \tag{A.92}$$

with $x(u) = [u, -\sum_i u_i]^\top$. Since we chose $y = \sigma(x) = \frac{\exp(x)}{\sum_j \exp(x_j)}$ we get $|\det \frac{\partial u}{\partial y}| = \frac{1}{y_i}\delta_{ij} - \frac{1}{K}\frac{1}{y_i}$ for $i = 1, ..., K-1$. By using

$$|Z + UWV| = |Z||W||W^{-1} + V^\top Z^{-1}U| \tag{A.93}$$

we get

$$|\det \frac{\partial u}{\partial y}| = (\prod_j^{K-1} \frac{1}{y_j})\frac{1}{K}(K - \underbrace{\frac{1}{y}\text{diag}(y))}_{K-1} \tag{A.94}$$

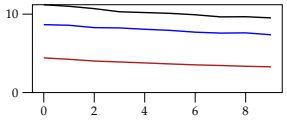
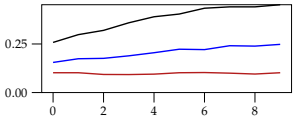
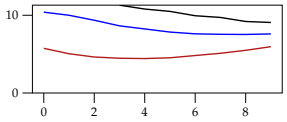
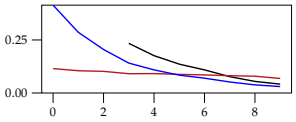
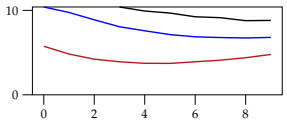
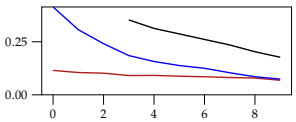
$$= \frac{1}{K}\prod_i^{K-1} \frac{1}{y_i} \tag{A.95}$$

We then draw samples y from a Dirichlet distribution transform them with a fake inverse-softmax $x = \log(y) - \frac{1}{K}\sum_i \log(y_i)$ and apply the transformed Gaussian to get $\sum_j \log(\mathcal{D}(y_j)/\mathcal{N}(x_j)$ to estimate the KL-divergence. The implementation can be found in the accompanying code. We find that the Dirichlet and our constructed Gaussian look very similar (see Figure A.15).

(inverse-)Wishart KL-divergence

The Wishart distribution has two parameters n and V. The parameters of the inverse Wishart are ν and Ψ. The parameters for the distances always have the form of $n_0 + n_0 \cdot c \cdot i$ and $V_0 + V_0 \cdot d \cdot i$ for $i = 0, 1, ..., 9$ where c and d are constants. Then there are three interesting combinations of parameters to investigate: fixed ν with increasing Ψ, increasing ν with fixed Ψ, and both increasing. In the main paper we present only the first combination, here we present all of them. For the following we choose $n_0 = 2.5, c = 0.5, V_0 = [[0.75, 0.5], [0.5, 1]], d = 0.25$. Results for the inverse Wishart for all three scenarios can be found in Table A.2. Setup and trends are similar for the Wishart distribution.

Table A.2: Comparison of Distances for Wishart distributions.

Version	KL-Divergence ↓	MMD ↓
static ν, inc. Ψ		
inc ν, static Ψ		
inc. ν, inc. Ψ		

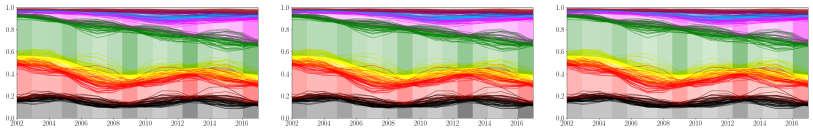


Figure A.16: Tübingen elections from 2002 to 2017 for three different neighborhoods of Tübingen.

A.3.2 German Elections

In the main text we compute the marginal distributions and the approximation qualities on a smaller dataset which uses much less datapoints, i.e. votes. In Figure A.16 we present a similar visualization for the small local elections as we showed in the main text for the German elections.

Appendix B: Laplace Bridge for BNNs

B.

B.1 Correction for zero-sum constraint

We know that the product rule of Gaussians yields

$$p(x|Ax = y) = \frac{p(x,y)}{p(y)} \tag{B.1}$$

$$= \mathcal{N}(x;\mu + \Sigma A^\top (A\Sigma A^\top)^{-1}(y - A\mu), \tag{B.2}$$
$$\Sigma - \Sigma A^\top (A\Sigma A^\top)^{-1} A\Sigma)$$

In our particular setup we have

$$p(x) = \mathcal{N}(x;\mu,\Sigma) \tag{B.3}$$

with constraint

$$p(I|x) = \delta(1x^\top - 0) = \lim_{\epsilon\to\infty} \mathcal{N}(0;1^\top x, \frac{1}{\epsilon}) \tag{B.4}$$

Therefore we get

$$p(x|I) = \mathcal{N}(x;\mu + \Sigma 1(1^\top \Sigma 1 - \frac{1}{\epsilon})^{-1}(0 - 1^\top \mu), \tag{B.5}$$
$$\Sigma - \Sigma 1(1^\top \Sigma 1 - \frac{1}{\epsilon})^{-1} 1^\top \Sigma)$$
$$= \mathcal{N}\left(x;\mu - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \mu}{\mathbf{1}^\top \Sigma \mathbf{1}}, \Sigma - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \Sigma}{\mathbf{1}^\top \Sigma \mathbf{1}}\right) \tag{B.6}$$

Variance correction

As described in the main text, the original Laplace Bridge scales worse with Σ than sampling and applying the softmax. In Figure B.1 you can see a contourplot that shows the scaling of mean and variance with and without correction. As suggested, the Variance has nearly no influence on the result before the correction but our correction fixes that.

Here is a short and informal explanation of how we found these limitations. During the experimentation with the LB, we found that it doesn't approximate the sample distribution well when Σ gets large. We then understood why (as detailed in the limitations section) and proposed a fix

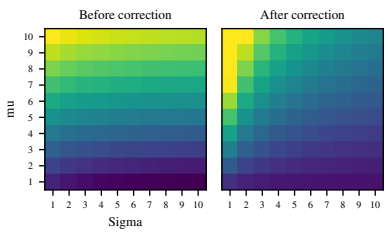


Figure B.1: Contourplot showing the scaling behavior of μ and Σ. In the left figure, we see that Sigma has nearly no influence on the scaling. Our correction in the right figure fixes that. Contour levels show the first entry of α on a log scale.

for these scenarios without damaging its behavior in all other scenarios. We experimented with multiple fixes and the result you see in the paper is the one that fulfilled most of our criteria. Therefore, the correction doesn't come from a principled theoretical derivation but is motivated by the theoretical findings.

B.2 Derivation of the Laplace Bridge

Assume we have a Dirichlet in the standard basis with parameter vector $\boldsymbol{\alpha}$ and probability density function:

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1}\,, \tag{B.7}$$

We aim to transform the basis of this distribution via the softmax transform to be in the new base π:

$$\pi_k(\mathbf{z}) := \frac{\exp(z_k)}{\sum_{l=1}^{K}\exp(z_l)}\,, \tag{B.8}$$

Usually, to transform the basis we would need the inverse transformation $H^{-1}(\mathbf{z})$ as described in the main paper. However, the softmax does not have an analytic inverse. Therefore David JC MacKay uses the following trick. Assume we know that the distribution in the transformed basis is:

$$\mathrm{Dir}_{\mathbf{z}}(\boldsymbol{\pi}(\mathbf{z})|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k(\mathbf{z})^{\alpha_k}\,, \tag{B.9}$$

then we can show that the original distribution is the result of the basis transform by the softmax.

The Dirichlet in the softmax basis: We show that the density over $\boldsymbol{\pi}$ shown in Equation B.9 transforms into the Dirichlet over $\mathbf{z}$. First, we consider the special case where $\boldsymbol{\pi}$ is confined to an $I-1$ dimensional subspace satisfying $\sum_i \boldsymbol{\pi}_i = c$. In this subspace we can represent φ by an $I-1$ dimensional vector φ such that

$$\pi_i = \varphi_i \quad i, \ldots, I-1 \tag{B.10}$$

$$\pi_I = c - \sum_{i}^{I-1}\varphi_i \tag{B.11}$$

and similarly we can represent $\mathbf{z}$ by an $I-1$ dimensional vector $\mathbf{a}$:

$$z_i = \mathbf{a}_i \quad i, \ldots, I-1 \tag{B.12}$$

$$z_I = 1 - \sum_{i}^{I-1}\mathbf{a}_i \tag{B.13}$$

then we can find the density over $\mathbf{a}$ (which is proportional to the required density over $\mathbf{z}$) from the density over φ (which is proportional to the given density over $\boldsymbol{\pi}$) by finding the determinant of the $(I-1)\times(I-1)$ Jacobian $\boldsymbol{J}$ given by

$$\begin{aligned} J_{ik} &= \frac{\partial \varphi_i}{\partial \mathbf{a}_l} = \sum_j^I \frac{\partial \boldsymbol{\pi}_i}{\partial \mathbf{z}_j}\frac{\partial \mathbf{z}_j}{\partial \mathbf{a}_k} \\ &= \delta_{ik}\boldsymbol{\pi}_i - \boldsymbol{\pi}_i\boldsymbol{\pi}_k + \boldsymbol{\pi}_i\boldsymbol{\pi}_I = \boldsymbol{\pi}_i(\delta_{ik} - (\boldsymbol{\pi}_k - \boldsymbol{\pi}_I)) \end{aligned} \tag{B.14}$$

We define two additional $I-1$ dimensional helper vectors $\mathbf{z}_k^+ := \mathbf{z}_k - \mathbf{z}_I$ and $n_k := 1$, and use $\det(I - xy^T) = 1 - x \cdot y$ from linear algebra. It follows that

$$\begin{aligned} \det J &= \prod_{i=1}^{I-1} \boldsymbol{\pi}_i \times \det[I - n\boldsymbol{\pi}^{+^T}] \\ &= \prod_{i=1}^{I-1} \boldsymbol{\pi}_i \times (1 - n \cdot \boldsymbol{\pi}^+) \\ &= \prod_{i=1}^{I-1} \boldsymbol{\pi}_i \times \left(1 - \sum_k \boldsymbol{\pi}_k^+\right) = I \prod_{i=1}^{I} \boldsymbol{\pi}_i \end{aligned} \tag{B.15}$$

Therefore, using Equation B.9 we find that

$$P(\boldsymbol{\pi}) = \frac{P(\mathbf{z})}{|\det \boldsymbol{J}|} \propto \prod_{i=1}^{I} \boldsymbol{\pi}_i^{\alpha_i - 1} \tag{B.16}$$

This result is true for any constant c since it can be put into the normalizing constant. Thereby we make sure that the integral of the distribution is 1 and we have a valid probability distribution.

B.3 Inversion of the Laplace Bridge

Through the figures of the 1D Dirichlet approximation in the main paper we have already established that the mode of the Dirichlet lies at the mean of the Gaussian distribution and therefore $\boldsymbol{\pi}(\mathbf{y}) = \frac{\alpha}{\sum_i \alpha_i}$. Additionally, the elements of $\mathbf{y}$ must sum to zero. These two constraints combined yield only one possible solution for $\boldsymbol{\mu}$.

$$\mu_k = \log \alpha_k - \frac{1}{K}\sum_{l=1}^{K} \log \alpha_l \tag{B.17}$$

Calculating the covariance matrix $\boldsymbol{\Sigma}$ is more complicated but layed out in the following. The logarithm of the Dirichlet is, up to additive constants

$$\log p_{\mathbf{z}}(\mathbf{z}|\alpha) = \sum_k \alpha_k \pi_k \tag{B.18}$$

Using π_k as the softmax of $\mathbf{y}$ as shown in Equation B.8 we can find the elements of the Hessian $\mathbf{L}$

$$L_{kl} = \hat{\alpha}(\delta_{kl}\hat{\pi_k} - \hat{\pi_k}\hat{\pi_l}) \tag{B.19}$$

where $\hat{\boldsymbol{\alpha}} := \sum_k \alpha_k$ and $\hat{\boldsymbol{\pi}} = \frac{\alpha_k}{\hat{\alpha}}$ for the value of $\boldsymbol{\pi}$ at the mode. Analytically inverting $\mathbf{L}$ is done via a lengthy derivation using the fact that we can write $\mathbf{L} = \boldsymbol{A} + \boldsymbol{X}\boldsymbol{B}\boldsymbol{X}^\top$ and inverting it with the Schur-complement. You can find the derivation in (Hennig, 2010). This process results in the inverse of the Hessian

$$L_{kl}^{-1} = \delta_{kl}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_l} - \frac{1}{K}\left(\sum_u^K \frac{1}{\alpha_u}\right)\right] \tag{B.20}$$

We are mostly interested in the diagonal elements, since we desire a sparse encoding for computational reasons and we otherwise needed to map a $K \times K$ covariance matrix to a $K \times 1$ Dirichlet parameter vector which would be a very overdetermined mapping. Note that K is a scalar not a matrix. The diagonal elements of $\boldsymbol{\Sigma} = \mathbf{L}^{-1}$ can be calculated as

$$\Sigma_{kk} = \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_l^k \frac{1}{\alpha_l}. \tag{B.21}$$

To invert this mapping we transform Equation B.17 to

$$\alpha_k = e^{\mu_k}\prod_l^K \alpha_l^{1/K} \tag{B.22}$$

by applying the logarithm and re-ordering some parts. Inserting this into Equation B.21 and re-arranging yields

$$\prod_l^K \alpha_l^{1/K} = \frac{1}{\boldsymbol{\Sigma}_{kk}}\left[e^{-\mu}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_u^K e^{-\mu_u}\right] \tag{B.23}$$

which can be re-inserted into Equation B.22 to give

$$\alpha_k = \frac{1}{\Sigma_{kk}}\left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2}\sum_l^K e^{-\mu_k}\right) \tag{B.24}$$

which is the final mapping. With Equations B.17 and B.21 we are able to map from Dirichlet to Gaussian and with Equation B.24 we are able to map the inverse direction.

B.4 Experimental Details

The exact experimental setups, i.e. network architectures, learning rates, random seeds, etc. can be found in the accompanying GitHub repository *. This section is used to justify some of the decisions we made during the process in more detail, highlight some miscellaneous interesting things and showcase the additional experiments promised in the main paper.

* https://github.com/mariushobbhahn/LB_for_BNNs_official

Table B.1: Comparing the extended probit approximation with the normalized version of the LB norm in the KFAC setting. The probit approximation seems to break down in the MNIST scenarios.

		KFAC Probit				KFAC LB norm			
Train	**Test**	**MMC ↓**	**AUROC ↑**	**ECE ↓**	**NLL ↓**	**MMC ↓**	**AUROC ↑**	**ECE ↓**	**NLL ↓**
MNIST	MNIST	0.105	0.000	2.258	0.883	0.975	0.000	0.043	0.018
MNIST	FMNIST	0.102	0.955	2.302	0.032	0.444	0.990	2.871	0.364
MNIST	notMNIST	0.103	0.922	2.300	0.043	0.409	0.986	2.854	0.294
MNIST	KMNIST	0.102	0.962	2.304	0.012	0.414	0.991	3.162	0.328
CIFAR10	CIFAR10	0.548	0.000	0.661	0.404	0.941	0.000	0.195	0.017
CIFAR10	CIFAR100	0.358	0.896	2.652	0.253	0.662	0.866	3.871	0.558
CIFAR10	SVHN	0.307	0.956	2.567	0.195	0.441	0.965	2.837	0.327

Mathematical description of the setup

In principle, the Gaussian over the weights required by the Laplace Bridge for BNNs can be constructed by any Gaussian approximate Bayesian method such as variational Bayes (Graves, 2011; Blundell et al., 2015) and Laplace approximations for NNs (D. J. C. MacKay, 1992; Ritter et al., 2018). We will focus on the Laplace approximation, which uses the same principle as the Laplace Bridge. However, in the Laplace approximation for neural networks, the posterior distribution over the weights of a network is the one that is approximated as a Gaussian, instead of a Dirichlet distribution over the outputs as in the Laplace Bridge.

Given a dataset $\mathrm{D} := \{(\mathbf{x}_i, t_i)\}_{i=1}^{D}$ and a prior $p(\boldsymbol{\theta})$, let

$$p(\boldsymbol{\theta}|\mathrm{D}) \propto p(\boldsymbol{\theta})p(\mathrm{D}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{(\mathbf{x},t)\in\mathrm{D}} p(y = t|\boldsymbol{\theta}, \mathbf{x}) \,, \tag{B.25}$$

be the posterior over the parameter $\boldsymbol{\theta}$ of an L-layer network $f_{\boldsymbol{\theta}}$. Then we can get an approximation of the posterior $p(\boldsymbol{\theta}|\mathrm{D})$ by fitting a Gaussian $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ where

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}} &= \boldsymbol{\theta}_{\mathrm{MAP}} \,, \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}} &= (-\nabla^2|_{\theta_{\mathrm{MAP}}} \log p(\boldsymbol{\theta}|\mathrm{D}))^{-1} =: \boldsymbol{H}_{\boldsymbol{\theta}}^{-1} \,. \end{aligned}$$

That is, we fit a Gaussian centered at the mode $\boldsymbol{\theta}_{\mathrm{MAP}}$ of $p(\boldsymbol{\theta}|\mathrm{D})$ with the covariance determined by the curvature at that point. We assume that the prior $p(\boldsymbol{\theta})$ is a zero-mean isotropic Gaussian $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma^2\boldsymbol{I})$ and the likelihood function is the Categorical density

$$p(\mathrm{D}|\boldsymbol{\theta}) = \prod_{(\mathbf{x},t)\in\mathrm{D}} \mathrm{Cat}(y = t|\mathrm{softmax}(f_{\boldsymbol{\theta}}(\mathbf{x}))) \,.$$

For various applications in Deep Learning, an approximation with full Hessian is often computationally too expensive. Indeed, for each input $\mathbf{x} \in \mathbb{R}^N$, one has to do K backward passes to compute the Jacobian $\boldsymbol{J}(\mathbf{x})$. Moreover, it requires an $\mathcal{O}(PK)$ storage which is also expensive since P is often in the order of millions. A cheaper alternative is to fix all but the last layer of $f_{\boldsymbol{\theta}}$ and only apply the Laplace approximation on $\boldsymbol{W}_L$, the last layer's weight matrix. This scheme has been used successfully by Snoek et al. (2015), A. G. Wilson, Z. Hu, et al. (2016), and Brosse et al. (2020), etc. and has been shown theoretically that it can mitigate overconfidence problems in ReLU networks (Kristiadi et al., 2020). In this case, given the

approximate last-layer posterior

$$p(\boldsymbol{W}^L|\mathrm{D}) \approx \mathcal{N}(\mathrm{vec}(\boldsymbol{W}^L)|\mathrm{vec}(\boldsymbol{W}^L_{\mathrm{MAP}}), \boldsymbol{H}^{-1}_{\boldsymbol{W}^L}), \tag{B.26}$$

one can efficiently compute the distribution over the logits. That is, let $\boldsymbol{\phi} : \mathbb{R}^N \to \mathbb{R}^Q$ be the first $L-1$ layers of $f_{\boldsymbol{\theta}}$, seen as a feature map. Then, for each $\mathbf{x} \in \mathbb{R}^N$, the induced distribution over the logit $\boldsymbol{W}^L\boldsymbol{\phi}(\mathbf{x}) =: \mathbf{z}$ is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{W}^L_{\mathrm{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top \otimes \boldsymbol{I})\boldsymbol{H}^{-1}_{\boldsymbol{W}^L}(\boldsymbol{\phi}(\mathbf{x}) \otimes \boldsymbol{I})), \tag{B.27}$$

where $\otimes$ denotes the Kronecker product.

An even more efficient last-layer approximation can be obtained using a Kronecker-factored matrix normal distribution (Ritter et al., 2018; Louizos et al., 2016; Sun et al., 2017). That is, we assume the posterior distribution to be

$$p(\boldsymbol{W}^L|\mathrm{D}) \approx \mathcal{MN}(\boldsymbol{W}^L|\boldsymbol{W}^L_{\mathrm{MAP}}, \boldsymbol{U}, \boldsymbol{V}), \tag{B.28}$$

where $\boldsymbol{U} \in \mathbb{R}^{K\times K}$ and $\boldsymbol{V} \in \mathbb{R}^{Q\times Q}$ are the Kronecker factorization of the inverse Hessian matrix $\boldsymbol{H}^{-1}_{\boldsymbol{W}^L}$ (Martens et al., 2015) and $\mathcal{MN}$ denotes the Matrix Normal distribution. In this case, for any $\mathbf{x} \in \mathbb{R}^N$, one can easily show that the distribution over logits is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{W}^L_{\mathrm{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{V}\boldsymbol{\phi}(\mathbf{x}))\boldsymbol{U}), \tag{B.29}$$

which is easy to implement and computationally cheap. Finally, and even more efficient, is a last-layer approximation scheme with a diagonal Gaussian approximate posterior, i.e. the so-called mean-field approximation. In this case, we assume the posterior distribution to be

$$p(\boldsymbol{W}^L|\mathrm{D}) \approx \mathcal{N}(\mathrm{vec}(\boldsymbol{W}^L)|\mathrm{vec}(\boldsymbol{W}^L_{\mathrm{MAP}}), \mathrm{diag}\,\boldsymbol{\sigma}^2), \tag{B.30}$$

where $\boldsymbol{\sigma}^2$ is obtained via the diagonal of the Hessian of the log-posterior w.r.t. $\mathrm{vec}(\boldsymbol{W}^L)$ at $\mathrm{vec}(\boldsymbol{W}^L_{\mathrm{MAP}})$.

OOD Detection

The test scenarios are: A two-layer convolutional network trained on the MNIST dataset (LeCun, 1998). The OOD datasets for this case are FMNIST (Xiao et al., 2017), notMNIST (Bulatov, 2011), and KMNIST (Clanuwat et al., 2018). For larger datasets, i.e. CIFAR-10 (Krizhevsky, Nair, et al., 2014), SVHN (Netzer et al., 2011), and CIFAR-100 (Krizhevsky, Nair, et al., 2014), we use a ResNet-18 network (He et al., 2016). In all scenarios, the networks are well-trained with 99% test accuracy on MNIST, 95.4% on CIFAR-10, 76.6% on CIFAR-100, and 100% on SVHN. For the sampling baseline, we use 100 posterior samples.

All network have been trained with conventional setups, i.e. we use ADAM with learning rate 1e−3 and weight decay 5e−4 for the MNIST experiments and SGD with a cosine annealing scheduler starting at learning rate 0.1 and momentum 0.9 for the CIFAR and SVHN experiments.

Probit vs LB

The KFAC setting of the probit comparison can be found in Table B.1. Especially in the MNIST scenario the probit approximation seems to break down since even in-dist detection is at chance level. The LB, on the other hand, yields reasonable results.

Appendix C: PIHAM C.

C.1 Modelling categorical node metadata

Equation 5.5 in the main text describes the general formulation of the expected value of an attribute x for node i. However, when the attribute x is categorical, the expression becomes more intricate because it has to account for the total number of attribute categories Z. In this case, $\boldsymbol{\pi_{ix}}(\boldsymbol{\Theta}) = [\pi_{ixz}(\boldsymbol{\Theta})]_{z\in[1,Z]}$ and $\boldsymbol{H_{kx}} = [H_{kxz}]_{z\in[1,Z]}$ are Z-dimensional vectors. Within this framework, H_{kxz} explains how much information from the z-th category of attribute x is used to create the k-th community, and $\pi_{ixz}(\boldsymbol{\Theta})$ follows the modelling approach outlined in Contisciani et al., 2020:

$$\pi_{ixz}(\boldsymbol{\Theta}) \approx \frac{1}{2}\sum_{k=1}^{K}(U_{ik}+V_{ik})\,H_{kxz}\,. \tag{C.1}$$

C.2 Model settings and hyperparameters choice

We employ PIHAM consistently, maintaining the same configurations and hyperparameters across all experiments. The only variation lies in the choice of the likelihood function, customized to match the data types under examination. Specifically, we adopt Bernoulli distributions for binary information, Poisson distributions for nonnegative discrete data, Gaussian distributions for real values, and Categorical distributions for categorical data.

We set the prior distributions as standard normal distributions $\mathcal{N}(0,1)$, serving as shrinkage regularization. Indeed, due to the complexity of the data, the objective function may become non-identifiable, thus requiring the enforcement of concavity and differentiability. The selection of these priors accommodates this necessity. On the contrary, for parameter initialization, we opt for wider normal distributions $\mathcal{N}(0,9)$ to facilitate exploration of various initial points.

PIHAM performs inference using the gradient-based method Automatic Differentiation, and in our implementation we employs the Adam optimizer to iteratively evaluate derivatives of the log-posterior. We set the learning rate of the optimizer equal to 0.5 and run the optimization procedure for 2000 iterations, maintaining a tolerance threshold of 10^{-8}. Furthermore, we execute the algorithm 50 times, each time starting from a different random initialization, and output the parameters corresponding to the realization with the highest objective function.

C.3 Comparison with existing methods in a homogeneous scenario

Data generation

We construct directed attributed multilayer networks following a procedure similar to that described in Contisciani et al., 2020. Initially, we generate interactions using a multilayer mixed-membership stochastic block model De Bacco et al., 2017. Subsequently, we assign node metadata, ensuring a 50% match with the node communities, while the remaining ones are made randomly. We set a configuration with $N = 500$ nodes, $L = 2$ layers of which one being assortative and the other disassortative, a categorical attribute with $Z = 3$ categories, and $K = 3$ overlapping communities. Networks are generated with increasing average degrees $\langle k \rangle \in \{10, 15, 20, \dots, 50\}$, producing 20 independent samples for each $\langle k \rangle$ value. To generate the membership matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, we initially assign equal-size unmixed group memberships and then introduce the overlapping for 20% of the nodes. The correlation between $\boldsymbol{U}$ and $\boldsymbol{V}$ is set equal to 0.1, with entries drawn from a Dirichlet distribution with parameter $\alpha = 0.1$. The affinity matrix $\boldsymbol{W^1}$ exhibits an assortative block structure with main probabilities $p_1 = \langle k \rangle K/N$ and secondary probabilities $p_2 = 0.1\, p_1$. Conversely, the affinity matrix $\boldsymbol{W^2}$ is generated using a disassortative block structure with off-diagonal probabilities $p_1 = \langle k \rangle K/N$ and diagonal probabilities $p_2 = 0.1\, p_1$. Self-loops are removed, and sparsity is preserved.

The resulting networks depict a simpler scenario characterized by homogeneous layers with nonnegative discrete weights and a single categorical attribute. Such scenario is crucial for testing our model against existing methods.

Experiment details

For comparison, we use MTCOV Contisciani et al., 2020, a probabilistic model that assumes overlapping communities as the main mechanism governing both interactions and node attributes. This model is specifically tailored to handle categorical attributes and nonnegative discrete weights, and employs an EM algorithm for parameter inference. We run MTCOV 50 times with different random initializations, maintaining the same tolerance as in our implementation. In addition, we set the maximum number of EM steps before termination at 500, and the threshold for declaring convergence based on the consecutive updates respecting the tolerance equal to 15. Lastly, we fix the scaling hyperparameter $\gamma = 0.5$, reflecting the matching constraint imposed in the synthetic data generation.

We assess the performance of PIHAM and MTCOV in both prediction and community detection tasks. Specifically, we evaluate their predictive capabilities using a 5-fold cross-validation routine, in which the dataset is split into five equal-size groups (folds), selected uniformly at random. The models are then trained on four of these folds (training set), covering 80% of the triples (i, j, ℓ) and 80% of the categorical vector entries, to learn their parameters. Next, we evaluate the models' performance on the

held-out group (test set). This process is repeated five times by varying the test set, resulting in five trials per iteration. As performance metrics, we use the Area Under the Curve (AUC) for the edge prediction, which represents the probability that a randomly selected edge has a higher expected value than a randomly selected non-existing edge, and accuracy for covariate prediction.

To evaluate the methods' performance in recovering communities, we first need to transform the inferred memberships to match the parameter space of the planted communities. Indeed, the ground truth mixed-memberships are represented as normalized vectors summing to 1, whereas our inferred parameters belong to the real-space. As discussed in the section "Parameter space and transformations" of the main text, two approaches can be adopted for such alignment: i) applying a softmax transformation to the point estimates $\hat{\boldsymbol{\mu}}^{\theta}$, or ii) employing the LM technique to obtain Dirichlet posterior distributions and utilizing a suitable statistic of these, where the mean serves as a viable option. In this experiment, we use both methods. For assessing performance, we utilize the Cosine Similarity (CS), a metric adept at capturing both hard and mixed-membership communities, ranging from 0 (indicating no similarity) to 1 (denoting perfect recovery). We compute the average cosine similarities of both membership matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, and then average them across the nodes.

Results

The results of both models in prediction and community detection tasks are depicted in Figure C.1. While MTCOV is expected to exhibit better performance due to its close alignment with the generative process underlying the synthetic data, PIHAM demonstrates comparable performance across all tasks despite its broader framework. This similarity is particularly notable in scenarios featuring denser networks. Indeed, our model, by treating all information equally, may face challenges in very sparse networks when relying only on a single covariate. Conversely, this is not an issue for MTCOV as it utilizes a linear combination of node and edge information, leveraging node covariates to address network sparsity.

An additional consideration is the choice of transformation used to compare the inferred communities with the planted ones. Both options yield similar results, as shown in Figure C.1C. The slight discrepancy between the two arises from the Dirichlet mean providing slightly more mixed-memberships.

Overall, these findings collectively suggest that PIHAM remains a valid approach even in less heterogeneous scenarios, demonstrating its ability to effectively compete with ad-hoc existing methods.

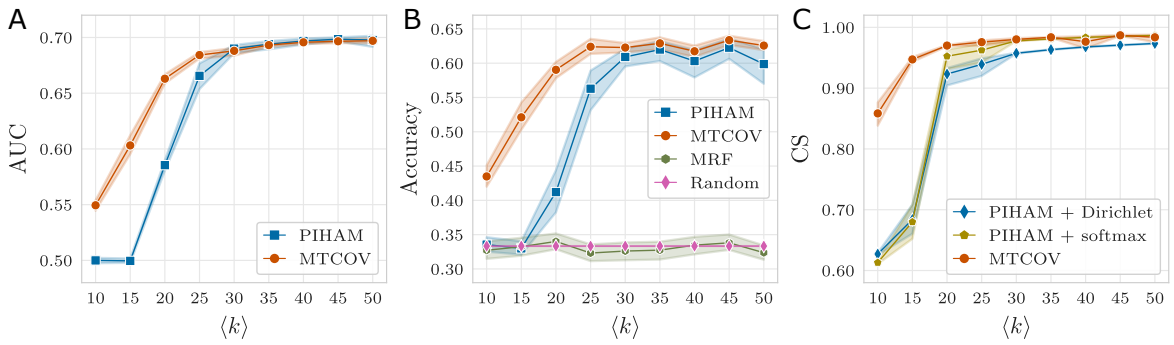


Figure C.1: Prediction and community detection performance on synthetic data. We analyze synthetic attributed multilayer networks with $N = 500$ nodes, $L = 2$ layers (one being assortative and the other disassortative), a categorical attribute with $Z = 3$ categories, $K = 3$ overlapping communities, and increasing average degrees $\langle k \rangle \in \{10, 15, 20, \ldots, 50\}$. The results represent averages and confidence intervals over 20 independent samples. For prediction tasks, we employ a 5-fold cross-validation procedure. The evaluation metrics include (A) the AUC for edge prediction, with a baseline of 0.5 corresponding to random choice, and (B) accuracy for covariate prediction. Here, MRF represents a baseline given by the predictions obtained from the maximum frequency in the training set, while Random denotes the uniform random probability over Z. (C) Community detection performance is assessed using CS. As inferred point estimates, we consider both the mean of transformed Dirichlet posterior distributions and the softmax transformation of $\hat{\boldsymbol{\mu}}^{\boldsymbol{\theta}}$. Overall, PIHAM exhibits comparable performance to MTCOV across all tasks despite its broader framework, especially in scenarios involving denser networks.

C.4 Validation on heterogeneous data

Data generation

We construct directed, heterogeneous, and attributed multilayer networks using the framework of PIHAM. Initially, we draw the latent variables $\boldsymbol{\Theta} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{H})$ from Gaussian distributions with specified hyperparameters, and then generate $\boldsymbol{A}$ and $\boldsymbol{X}$ according to the data types, following Eqs. 5.2-5.7 in the main text. We configure the networks with $L = 3$ heterogeneous layers: one with binary interactions, the second with nonnegative discrete weights, and the third with real values. Additionally, each node is associated with three covariates: one categorical with $Z = 4$ categories, one with nonnegative discrete values, and the last with real values. Networks are generated with increasing number of nodes $N \in \{100, 200, \ldots, 1000\}$, and varying number of overlapping communities $K \in \{3, 4, 5\}$. For each combination (N, K), we generate 20 different samples. To generate the membership matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, we assign equal-size group memberships and draw the entries of the matrices from distributions with different means, according to the group the nodes belong to. Specifically, $u_{ik} \sim \mathcal{N}(2, 0.04)$ if i is associated with group k, otherwise $u_{ik} \sim \mathcal{N}(-1, 0.04)$. Similarly, $v_{ik} \sim \mathcal{N}(2, 0.09)$ if i is associated with group k, otherwise $v_{ik} \sim \mathcal{N}(-1, 0.09)$. The affinity tensor $\boldsymbol{W}$ exhibits an assortative block structure in each layer, with diagonal entries following normal distributions with zero mean and $\sigma = 0.45$, and off-diagonal entries drawn from normal distributions with $\mu = -4$ and $\sigma = 0.45$. The community-covariate matrix $\boldsymbol{H}$ is set to maintain coherence between node interactions and covariates, avoiding additional noise in the data. For the categorical variable, $H_{kxz} \sim \mathcal{N}(0.5 + k, 0.04)$ if $k = z$, otherwise $H_{kxz} \sim \mathcal{N}(0, 0.04)$. When $K = 3$ or $K = 5$, we set $H_{kx4} \sim \mathcal{N}(0.2, 0.04)$ or $H_{5xz} \sim \mathcal{N}(0.2, 0.04)$, respectively. The nonnegative discrete attribute is generated according to $H_{kx} \sim \mathcal{N}(1.5 \times \frac{k+2}{3}, 0.01)$, while the covariate with real values is constructed with $H_{kx} \sim \mathcal{N}(4 + (1 - k) \times 3, 0.04)$.

The resulting networks represent a general scenario featuring heteroge-

neous interactions and node covariates, which is essential to validate our approach and demonstrate its flexibility.

Experiment details

We assess the performance of PIHAM by evaluating its predictive capabilities through a 5-fold cross-validation routine. In this approach, the dataset is randomly divided into five equal-sized groups (folds), and the model is trained on four of them (the training set), which include 80% of the triples (i, j, ℓ) and 80% of the entries of each attribute vector, to learn its parameters. The performance of the model is then evaluated on the remaining fold (the test set). This process is repeated five times, each time with a different fold as the test set, resulting in five trials per iteration. For performance metrics, we use different measures depending on the type of information being evaluated. For binary interactions, we use the Area Under the Curve (AUC), which represents the probability that a randomly selected edge has a higher expected value than a randomly selected non-existing edge. The AUC ranges from 0 to 1, with 0.5 representing the random baseline. For nonnegative discrete data, we use the Maximum Absolute Error (MAE), and for real values, we use the Root Mean Squared Error (RMSE). In both cases, lower values indicate better performance. Additionally, for categorical attribute predictions, we use accuracy, which ranges from 0 to 1, with 1 indicating perfect recovery.

C.5 Interpretation of posterior estimates

Experiment details

Interpreting posterior distributions can be challenging, especially with large datasets. To address this, we propose two different approaches to summarize the inferred results.

First, we employ a metric to quantify the area of overlap between distributions. We use the method proposed in Wand et al., 2011, which defines the integrated absolute error (IAE) between two distributions as:

$$\text{IAE} = \int_{-\infty}^{\infty} \left| p_1(x) - p_2(x) \right| dx \,, \tag{C.2}$$

with IAE $\in [0, 2]$. When p_1 and p_2 completely overlap, the difference between them is a line in zero, and the IAE is 0. Conversely, if there is no overlap, the IAE is 2, which is the sum of the integrals of the two distributions. To normalize this metric to the range $[0, 1]$, we define:

$$\text{Overlap} = 1 - \frac{1}{2}\,\text{IAE}\,. \tag{C.3}$$

In this case, an Overlap of 0 indicates no overlap between the distributions, while an Overlap of 1 represents perfect matching. We compute this measure between every pair of distributions for each node and then calculate the average.

Computing the Overlap for many communities can be computationally expensive due to the need to evaluate all pairwise combinations. As an alternative, we use the L_2-barycenter distribution, which represents a weighted average of the node-community distributions Benamou et al., 2015; Coz et al., 2023. This approach simplifies the problem by focusing on a single distribution per node instead of K different ones. We calculate this distribution using the `POT Python package` **flamary2021pot**, and we quantify it by computing its variance (σ^2) using the trapezoidal rule to approximate the integral. Higher values indicate nodes with harder memberships, as the barycenter is more spread due to the individual distributions being more distant from each other. Conversely, lower variance suggests more overlap among the distributions, indicating a more mixed-membership scenario.

C.6 Analysis of a social support network of a rural Indian village

Data pre-processing

We analyze a real-world dataset describing a social support network within a village in Tamil Nadu, India, referred to as "Aḷakāpuram" Power, 2015; Power, 2017. The data were collected in 2013 through surveys, in which adult residents were asked to nominate individuals who provided various types of support. In our analysis, we consider six different binary support questions, each forming a layer in the network, and we exclude individuals without any of these interactions. Details on these layers, including the number of edges and the average degree, are provided in Table C.1, while a visual representation can be found in Figure C.2. Additionally, we construct a seventh layer that incorporates the geographical distance between individuals' households. Specifically, we define the entries of this layer as $A^7_{ij} = \frac{1}{\sqrt{1+d_{ij}}}$, where d_{ij} is the distance between the households of individuals i and j, and we set $A^7_{ii} = 0$. Note that higher values indicate closer proximity, while lower values represent greater distances. The resulting adjacency tensor is then represented as $\boldsymbol{A} = \{\boldsymbol{A}^{\boldsymbol{\ell}} \in \{0,1\}^{N\times N}\ \forall \ell \in [1,6], \boldsymbol{A}^{\boldsymbol{7}} \in \mathbb{R}^{N\times N}\}$.

In addition, several attributes were collected, including information like gender, age, and caste, among others. For our analysis, we focus on caste, religion, and years of education, as ethnographic work and previous analyses Power, 2017; Power and Ready, 2018 suggest these attributes

Table C.1: Summary statistics for the first six binary layers of the "Aḷakāpuram" social support network. E denotes the number of edges, while $\langle k \rangle$ represents the average degree. A visual representation of these layers can be found in Figure C.2.

Layer	Description	E	$\langle k \rangle$
L_1	Talk about important matters	880	4.2
L_2	Help finding a job	437	2.1
L_3	Help with physical tasks	758	3.6
L_4	Borrow household items from	876	4.2
L_5	Ask for money	386	1.8
L_6	Talk to for pleasure	824	3.9

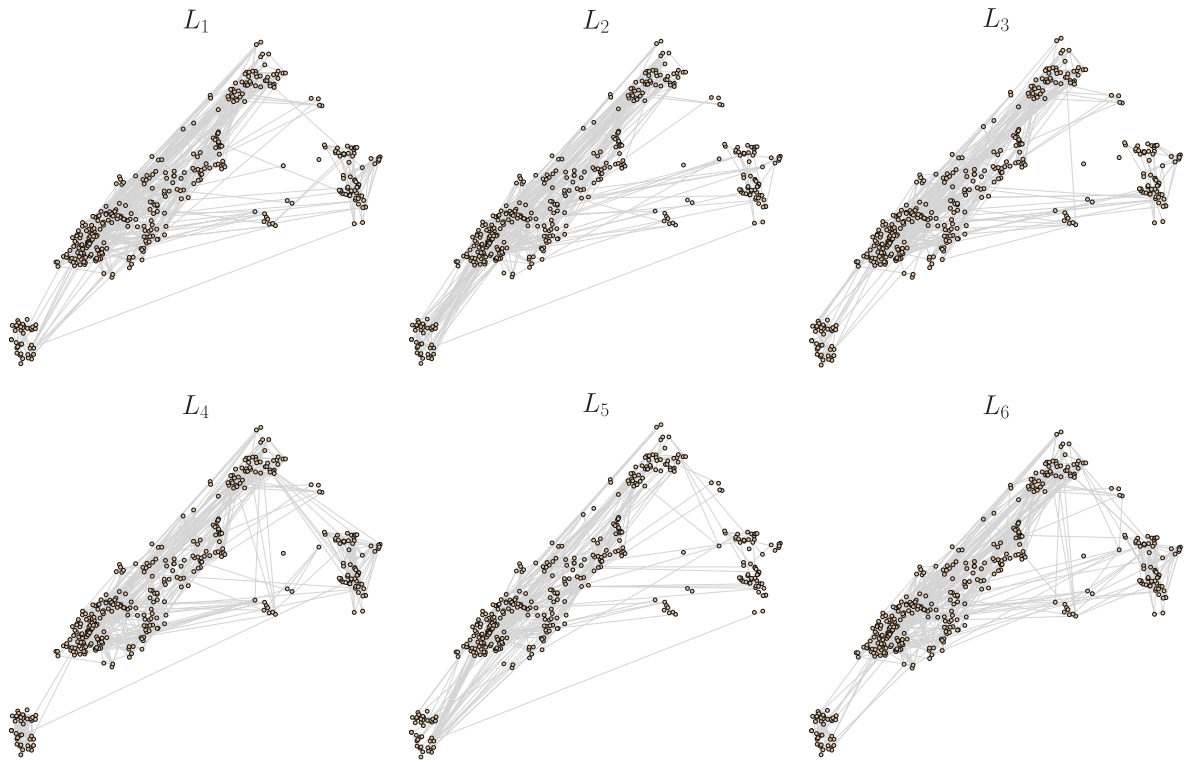


Figure C.2: Visual representation of the first six binary layers of the "Alakāpuram" social support network. Details on these layers, including the information encoded, the number of edges, and the average degree, are provided in Table C.1. The position of the nodes reflects the geographical distance between individuals' households.

significantly influence how villagers relate to one another. Specifically, caste is a categorical attribute with $Z_{caste} = 14$ categories, religion has $Z_{religion} = 3$ categories, and years of education are represented as nonnegative discrete values $\boldsymbol{X}_{\cdot 3} \in \mathbb{N}_0^N$.

The resulting heterogeneous attributed multilayer network comprises $N = 419$ nodes, $L = 7$ layers, and $P = 3$ node attributes.

Results

To determine the number of communities K, we employ a 5-fold cross-validation procedure for $K \in [1, 10]$ and select the value that exhibits the optimal performance. Similar to the synthetic experiments, for a given K, we train the model on four folds (training set), which include 80% of the triples (i, j, ℓ) and 80% of the entries of each attribute vector, to learn its parameters. The model's performance is then evaluated on the remaining fold (the test set). This process is repeated five times, each with a different fold as the test set, resulting in five trials per iteration. We evaluate prediction performance in the first six binary layers using the Area Under the Curve (AUC), representing the probability that a randomly selected edge has a higher expected value than a randomly selected non-existing edge. The AUC ranges from 0 to 1, with 0.5 indicating the random baseline. For the seventh layer containing real values, we use the Root Mean Squared Error (RMSE), where lower values indicate better performance. Additionally, we assess prediction performance for the attributes using accuracy for caste and religion, which ranges from 0 to 1 (with 1 indicating perfect recovery), and the Maximum Absolute Error (MAE) for years of

Table C.2: Prediction performance on the "Alakāpuram" social support network. For a given number of communities K, we employ a 5-fold cross-validation procedure and report averages and standard deviations over the five trials, each using a different fold as the test set. We evaluate prediction performance in the first six binary layers using the Area Under the Curve (AUC) and for the seventh layer containing real values using the Root Mean Squared Error (RMSE). Additionally, we assess prediction performance for the attributes using accuracy for caste ($\boldsymbol{X}_{\cdot 1}$) and religion ($\boldsymbol{X}_{\cdot 2}$), and the Maximum Absolute Error (MAE) for years of education ($\boldsymbol{X}_{\cdot 3}$), represented as nonnegative discrete values. The baselines are omitted for brevity. In our experiments, we set $K = 6$ as it achieves the best performance across most prediction metrics, and overall, PIHAM demonstrates robust outcomes with the chosen fixed value of K.

K	AUC ($[\boldsymbol{A}^{\ell}]_{\ell\in[1,6]}$)	RMSE ($\boldsymbol{A}^{7}$)	Accuracy ($\boldsymbol{X}_{\cdot 1}$)	Accuracy ($\boldsymbol{X}_{\cdot 2}$)	MAE ($\boldsymbol{X}_{\cdot 3}$)
1	0.575 ± 0.007	0.096 ± 0.003	0.55 ± 0.05	0.84 ± 0.04	4.3 ± 0.3
2	0.717 ± 0.009	0.096 ± 0.003	0.55 ± 0.04	0.84 ± 0.04	4.3 ± 0.2
3	0.73 ± 0.01	0.096 ± 0.003	0.55 ± 0.05	0.84 ± 0.04	**4.3 ± 0.1**
4	**0.77 ± 0.01**	0.093 ± 0.003	0.58 ± 0.05	0.84 ± 0.03	4.4 ± 0.1
5	0.77 ± 0.02	0.088 ± 0.005	0.63 ± 0.06	0.87 ± 0.06	4.4 ± 0.1
6	0.76 ± 0.01	**0.086 ± 0.003**	**0.70 ± 0.07**	**0.87 ± 0.02**	4.4 ± 0.1
7	0.74 ± 0.01	0.089 ± 0.003	0.63 ± 0.04	0.85 ± 0.04	4.4 ± 0.1
8	0.73 ± 0.01	0.095 ± 0.003	0.27 ± 0.06	0.83 ± 0.04	4.4 ± 0.1
9	0.72 ± 0.01	0.095 ± 0.003	0.20 ± 0.04	0.84 ± 0.04	4.4 ± 0.1
10	0.72 ± 0.01	0.095 ± 0.003	0.2 ± 0.2	0.84 ± 0.04	4.4 ± 0.2

education, with lower values indicate better performance. The results are displayed in Table C.2. In our experiments, we set $K = 6$ as it achieves the best performance across most prediction metrics. Note that, summarizing and evaluating results in a heterogeneous setting using a single metric is challenging, as discussed in the section "Validation on heterogeneous data" of the main text. The results in Table C.2 further demonstrate that PIHAM achieves robust outcomes with the chosen fixed value of K.

To provide a qualitatively interpretation of the inferred results, Fig. 4 in the main text shows the inferred out-going communities $\hat{\boldsymbol{U}}$. Here, we present additional visualizations for other model parameters. In particular, Figure C.3 displays the inferred $K \times Z_{caste}$-dimensional matrix $\hat{\boldsymbol{H}}_{\cdot 1}$, which explains the contributions of each caste category to the formation of the k-th community. Panel B presents the inferred posterior distributions $\hat{H}_{k1z} \sim \mathcal{N}(\hat{H}_{k1z}; \hat{\mu}^{H}_{k1z}, (\hat{\sigma}^{H}_{k1z})^2)$, with different colors representing distinct caste categories. Panel A, instead, shows the softmax transformation of the MAP estimates $\hat{\boldsymbol{\mu}}^{\boldsymbol{H}}_{\boldsymbol{k1}}$ for easier interpretation. Note that, the matrix is transposed in the plot, so that each column sums to 1, and the y-axis lists the caste categories. From this figure, we observe that the first and second communities predominantly consist of nodes from to the Yatavar and Paṟaiyar castes, respectively. Similarly, K_3 comprises nodes from the Kulalar and Maṟavar castes, while communities K_4, K_5, and K_6 are predominantly composed by nodes from the Palḷar caste, which is also the most represented caste in the dataset. Furthermore, Figure C.4 shows the inferred posterior distributions of the community-covariate vector related to years of education, $\hat{H}_{k3} \sim \mathcal{N}(\hat{H}_{k3}; \hat{\mu}^{H}_{k3}, (\hat{\sigma}^{H}_{k3})^2)$. From this figure, it is evident that this attribute plays a more significant role in determining K_6, as its distribution has a notably higher mean compared to the distributions of the other communities. Lastly, Figure C.5 displays the affinity tensor $\hat{\boldsymbol{W}}$, which explains the edge density between different community pairs in the various layers. To improve visualization clarity, we apply a logistic transformation to the MAP estimates $[\hat{\mu}^{\ell}_{kq}]^{W}$. This figure suggests that the different layers predominantly exhibit an assortative structure, where nodes tend to interact more with individuals within the same community than with those from different communities.

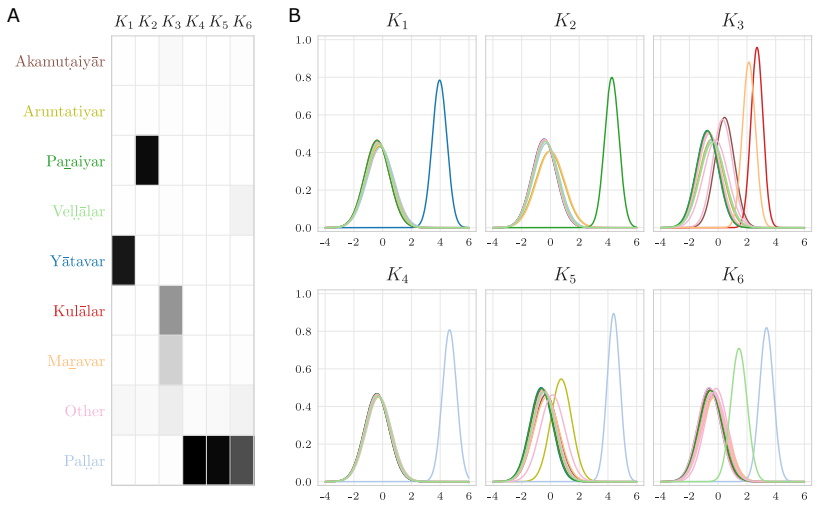


Figure C.3: Inference of the community-caste parameter in the "Aḷakāpuram" social support network. We display the inferred $K \times Z_{caste}$-dimensional matrix $\hat{\boldsymbol{H}}_{\cdot \mathbf{1}}$, which explains the contributions of each caste category to the formation of the k-th community. For privacy reasons, nodes belonging to castes with fewer than five individuals are aggregated into an "Other" category. (A) Transformation of the MAP estimates $\hat{\boldsymbol{\mu}}^{\boldsymbol{H}}_{\boldsymbol{k1}}$ inferred by PIHAM using the softmax function. In this plot, the matrix is transposed, so that each column sums to 1, and the y-axis lists the caste categories. (B) Inferred posterior distributions $\hat{H}_{k1z} \sim \mathcal{N}(\hat{H}_{k1z}; \hat{\mu}^{H}_{k1z}, (\hat{\sigma}^{H}_{k1z})^2)$, with different colors representing distinct caste categories. The first and second communities predominantly consist of nodes from to the Yatavar and Paṟaiyar castes, respectively. K_3 comprises nodes from the Kulalar and Maṟavar castes, while communities K_4, K_5, and K_6 are predominantly composed by nodes from the Paḷḷar caste.

However, we notice some variations for certain layers. For instance, L_2 (help finding a job) has few non-zero diagonal values, suggesting that this type of support sometimes requires seeking out individuals in different communities. Moreover, L_7, corresponding to the geographical distance between nodes, has several off-diagonal entries, particularly for communities K_4, K_5, and K_6, suggesting a weakened effect for physical proximity for those communities.

Taken together, these findings suggest that PIHAM utilizes all the input information to infer partitions that effectively integrate all of them in a meaningful manner. In addition, the inferred affinity matrices illustrate how different layers can exhibit different community structures, a diversity that can be captured by our model.

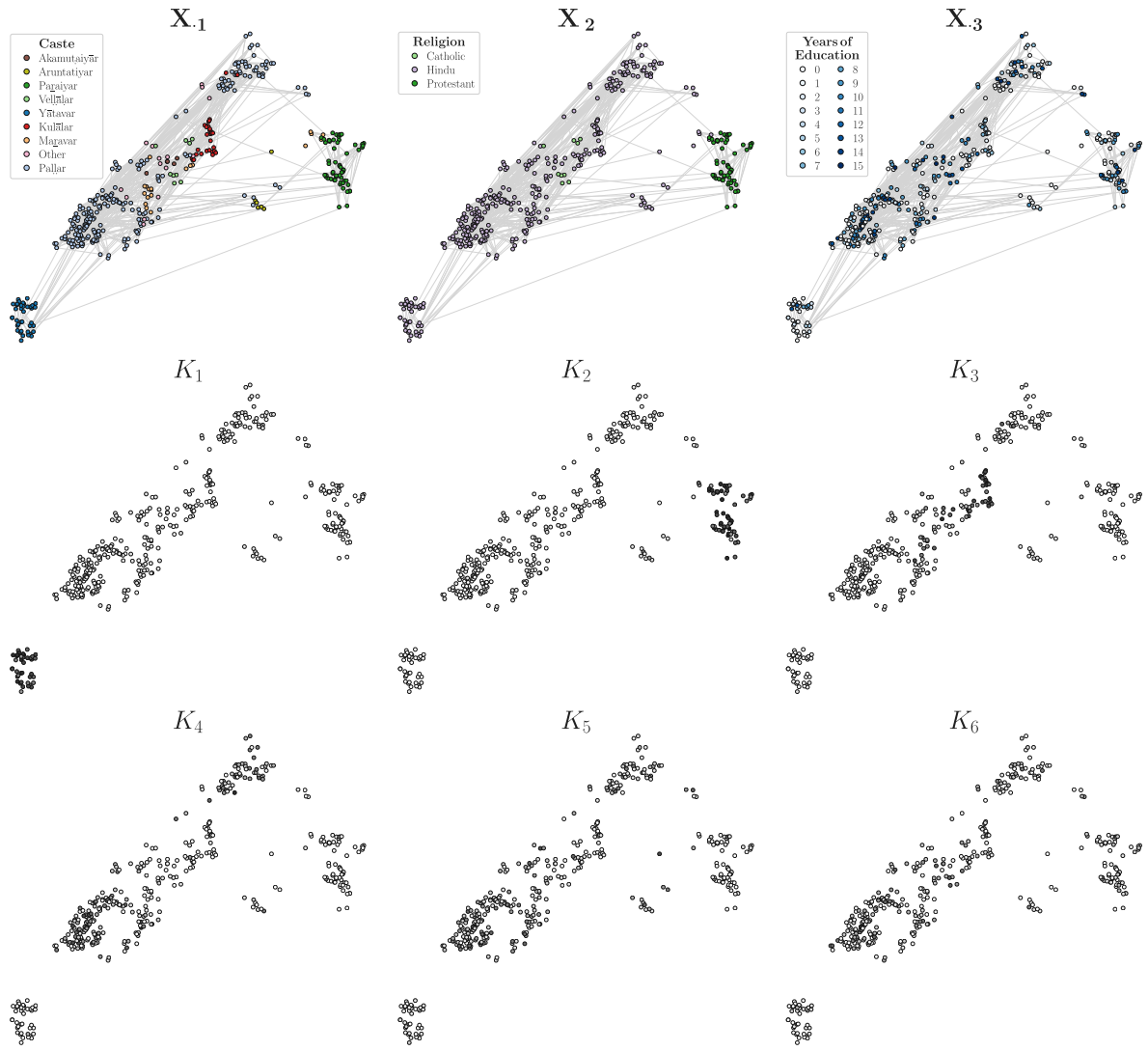
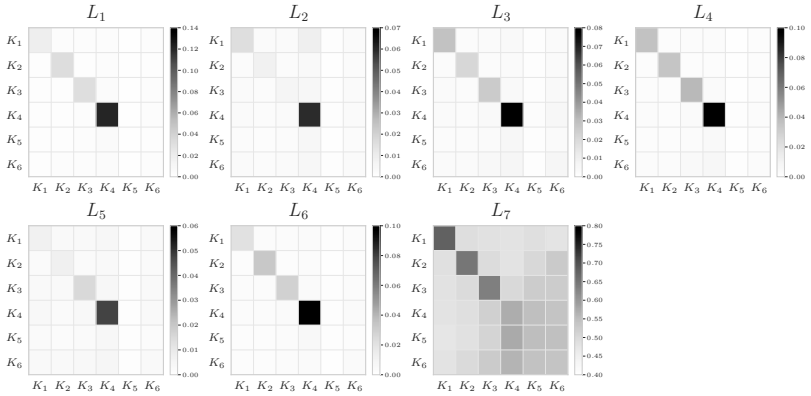


Figure C.4: Inference of the community-education parameter in the "Aḷakāpuram" social support network. We display the inferred posterior distributions $\hat{H}_{k3} \sim \mathcal{N}(\hat{H}_{k3}; \hat{\mu}^H_{k3}, (\hat{\sigma}^H_{k3})^2)$, which explain how the attribute related to years of education is distributed among the K communities. The distribution of the attribute in K_6 has a notably higher mean compared to the distributions of the other communities, suggesting that $\boldsymbol{X}_{\cdot 3}$ plays a significant role in determining this community.

Figure C.5: Inference of the affinity tensor in the "Aḷakāpuram" social support network. We display the MAP estimates of the inferred $\hat{\boldsymbol{W}}$, which explain the edge density between different community pairs in the various layers. To improve visualization clarity, we apply a logistic transformation to the MAP estimates $[\hat{\mu}^\ell_{kq}]^W$. The different layers predominantly exhibit an assortative structure, with some variations for certain layers. For instance, L_2 (help finding a job) has few non-zero diagonal values, suggesting that this type of support sometimes requires seeking out individuals in different communities. Moreover, L_7, corresponding to the geographical distance between nodes, has several off-diagonal entries, particularly for communities K_4, K_5, and K_6, suggesting a weakened effect for physical proximity for those communities.



Bibliography

Stigler, S. M. (1981). "Gauss and the Invention of Least Squares". *The Annals of Statistics* 9.3, pages 465–474. (Visited on 03/21/2023).

Cortes, C. and V. Vapnik (1995). "Support-vector networks". *Machine Learning* 20.3, pages 273–297.

Schölkopf, B., A. Smola, and K.-R. Müller (1997). "Kernel principal component analysis". *Artificial Neural Networks — ICANN'97*. Edited by W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud. *Springer Berlin Heidelberg*.

Hofmann, T., B. Schölkopf, and A. J. Smola (2008). "Kernel methods in machine learning". *The Annals of Statistics* 36.3.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". *nature* 323.6088, pages 533–536.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". *Advances in Neural Information Processing Systems*. Edited by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Volume 25. *Curran Associates, Inc.*

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li (2014). "ImageNet Large Scale Visual Recognition Challenge". *CoRR* abs/1409.0575. arXiv: 1409.0575.

Sevilla, J., L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos (2022). "Compute Trends Across Three Eras of Machine Learning". *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Villalobos, P., J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbhahn (2022). "Machine Learning Model Sizes and the Parameter Gap". arXiv: 2207.02852 [cs.LG].

Villalobos, P., J. Sevilla, L. Heim, T. Besiroglu, M. Hobbhahn, and A. Ho (2022). "Will we run out of data? An analysis of the limits of scaling datasets in Machine Learning". arXiv: 2211.04325 [cs.LG].

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller (2013). "Playing Atari with Deep Reinforcement Learning". arXiv: 1312.5602 [cs.LG].

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning". *Nature* 575.7782, pages 350–354.

Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis (2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". arXiv: 1712.01815 [cs.AI].

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). "Language Models are Few-Shot Learners". *CoRR* abs/2005.14165. arXiv: 2005.14165.

Anthropic (2024). "Introducing the next generation of Claude". URL: https://www.anthropic.com/news/claude-3-family.

Wilson, A. G. and P. Izmailov (2021). "Deep Ensembles as Approximate Bayesian Inference".

Korbak, T., E. Perez, and C. L. Buckley (2022). "RL with KL penalties is better viewed as Bayesian inference". arXiv: 2205.11275 [cs.LG].

MacKay, D. J. C. (1992). "A Practical Bayesian Framework for Backpropagation Networks". *Neural Comput.* 4.3, pages 448–472.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017a). "Variational Inference: A Review for Statisticians". *Journal of the American Statistical Association* 112.518, pages 859–877.

Gilks, W., S. Richardson, and D. Spiegelhalter (1995). "Markov Chain Monte Carlo in Practice". Chapman & Hall/CRC Interdisciplinary Statistics. *Taylor & Francis*.

Hoffman, M. D. and A. Gelman (2011). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". arXiv: 1111.4246 [stat.CO].
Chib, S. and E. Greenberg (1995). "Understanding the Metropolis-Hastings Algorithm". *The American Statistician* 49.4, pages 327–335.
Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin (2013). "Bayesian Data Analysis". 3rd edition. *CRC Press*.
Izmailov, P., S. Vikram, M. D. Hoffman, and A. G. Wilson (2021). "What Are Bayesian Neural Network Posteriors Really Like?" *CoRR* abs/2104.14421. arXiv: 2104.14421.
He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
Krizhevsky, A., V. Nair, and G. Hinton (2014). "The CIFAR-10 dataset". *online: http://www. cs. toronto. edu/kriz/cifar. html* 55.
Kristiadi, A., M. Hein, and P. Hennig (2020). "Being Bayesian, even just a bit, fixes overconfidence in relu networks". *ICML*. PMLR, pages 5436–5446.
Daxberger, E., A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig (2021). "Laplace Redux–Effortless Bayesian Deep Learning". *NeurIPS*.
Charpentier, B., D. Zügner, and S. Günnemann (2020). "Posterior Network: Uncertainty Estimation without OOD Samples via Density-Based Pseudo-Counts". arXiv: 2006.09239 [cs.LG].
Diaconis, P. (1988). "Bayesian Numerical Analysis".
O'Hagan, A. (1991). "Bayes–Hermite quadrature". *Journal of Statistical Planning and Inference* 29, pages 245–260.
Ghahramani, Z. and C. Rasmussen (2002). "Bayesian Monte Carlo". *Advances in Neural Information Processing Systems*. Volume 15. *MIT Press*.
Briol, F.-X., C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic (2017). "Probabilistic Integration: A Role in Statistical Computation?" arXiv: 1512.00933 [stat.ML].
Hennig, P., M. A. Osborne, and H. P. Kersting (2022). "Probabilistic Numerics: Computation as Machine Learning". *Cambridge University Press*.
Tskarvon (2024). URL: https://en.wikipedia.org/wiki/Bayesian_quadrature#/media/File:Bayesian_quadrature_animation.gif.
Gunter, T., M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts (2014). "Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature". arXiv: 1411.0439 [stat.ML].
Kanagawa, M. and P. Hennig (2019). "Convergence Guarantees for Adaptive Bayesian Quadrature Methods". arXiv: 1905.10271 [stat.ML].
Hobbhahn, M. and T. Besiroglu (2022). "Trends in GPU price-performance". *EPOCH. June*.
Hobbhahn, M., L. Heim, and A. Gökçe (2023). "Trends in Machine Learning Hardware". *EPOCH. November*.
Hobbhahn, M., T. Lieberum, and D. Seiler (2022). "Investigating causal understanding in LLMs". https://openreview.net/forum?id=FQI5KxgFRc. Accessed: 2024-04-21.
Hobbhahn, M., E. Landgrebe, and E. Barnes (2022). "Reflection Mechanisms as an Alignment Target: A Survey". https://openreview.net/forum?id=mHitdTIovWL. Accessed: 2024-04-21.
Scheurer, J., M. Balesni, and M. Hobbhahn (2023). "Technical report: Large language models can strategically deceive their users when put under pressure". *arXiv preprint arXiv:2311.07590*.
Hobbhahn, M. (2021a). URL: https://www.mariushobbhahn.com/2021-05-20-Change_of_variable/.
Jacod, J. and P. Protter (2004). "Probability Essentials". Universitext. *Springer Berlin Heidelberg*.
Hobbhahn, M. (2021b). URL: https://www.mariushobbhahn.com/2021-06-10-ExpFam_tutorial/.
Griewank, A. and A. Walther (2008). "Evaluating Derivatives". Second. *Society for Industrial* and *Applied Mathematics*. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717761.
Rasmussen, C. E. (2006). "Gaussian processes for machine learning". *MIT Press*.
Murray, I., R. P. Adams, and D. J. C. MacKay (2009). "Elliptical slice sampling". arXiv: 1001.0175 [stat.CO].
MacKay, D. J. (1998). "Choice of Basis for Laplace Approximation". *Machine Learning* 33.1, pages 77–86.
Nickisch, H. (2012). "Glm-Ie: Generalised Linear Models Inference & Estimation Toolbox". *J. Mach. Learn. Res.* 13.null, pages 1699–1703.
Minka, T. P. (2001). "Expectation Propagation for Approximate Bayesian Inference". *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. UAI'01. *Morgan Kaufmann Publishers Inc.*, pages 362–369.

Neal, R. M. (1993). *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical report.
Seeger, M., H. Nickisch, R. Pohmann, and B. Schölkopf (2009). "Optimization of k-Space Trajectories by Bayesian Experimental Design". 17.2627.
Williams, C. K. I. and D. Barber (1998). "Bayesian Classification With Gaussian Processes". *IEEE Trans. Pattern Anal. Mach. Intell.* 20.12, pages 1342–1351.
Wilson, A. and Z. Ghahramani (2010). "Generalised Wishart Processes". *Uncertainty in Artificial Intelligence (2011)*.
Bishop, C. M. (2006). "Pattern recognition and machine learning". Information science and statistics. Softcover published in 2016. New York, NY: *Springer*.
Milios, D., R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone (2018). "Dirichlet-based Gaussian Processes for Large-scale Calibrated Classification". *CoRR* abs/1805.10915. arXiv: 1805.10915.
Jia, Y., S. Kechagias, J. Livsey, R. Lund, and V. Pipiras (2021). "Latent Gaussian Count Time Series". *Journal of the American Statistical Association*, pages 1–11.
Hennig, P. (2010). "Approximate Inference in Graphical Models". PhD thesis. University of Cambridge.
Challis, E. and D. Barber (2013). "Gaussian kullback-leibler approximate inference". *The Journal of Machine Learning Research* 14.1, pages 2239–2286.
Rue, H., S. Martino, and N. Chopin (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations". *Journal of the Royal Statistical Society Series B* 71, pages 319–392.
Martínez-Minaya, J., F. Lindgren, A. López-Quílez, D. Simpson, and D. Conesa (2019). "The Integrated nested Laplace approximation for fitting models with multivariate response". arXiv: 1907.04059 [stat.CO].
Dua, D. and C. Graff (2017). "UCI Machine Learning Repository". URL: http://archive.ics.uci.edu/ml.
Bruin, J. (2011). *newtest: command to compute new test @ONLINE*. URL: https://stats.idre.ucla.edu/stata/ado/analysis/.
Kleiber, C. and A. Zeileis (2008). "Applied Econometrics with R". ISBN 978-0-387-77316-2. New York: *Springer-Verlag*.
Gretton, A., K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola (2012). "A Kernel Two-Sample Test". *Journal of Machine Learning Research* 13, pages 723–773.
McAllister, R., Y. Gal, A. Kendall, M. van der Wilk, A. Shah, R. Cipolla, and A. Weller (2017). "Concrete Problems for Autonomous Vehicle Safety: Advantages of Bayesian Deep Learning". *IJCAI*.
Michelmore, R., M. Kwiatkowska, and Y. Gal (2018). "Evaluating Uncertainty Quantification in End-to-End Autonomous Driving Control". *CoRR* abs/1811.06817. arXiv: 1811.06817.
Begoli, E., T. Bhattacharya, and D. Kusnezov (2019). "The need for Uncertainty Quantification in machine-assisted medical decision making". *Nat Mach Intell* 1, pages 20–23.
Nguyen, A., J. Yosinski, and J. Clune (2015). "Deep Neural Networks are easily fooled: High confidence predictions for unrecognizable images". *CVPR*.
Hein, M., M. Andriushchenko, and J. Bitterwolf (2019). "Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem". *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Hennig, P., D. Stern, R. Herbrich, and T. Graepel (2012). "Kernel Topic Models". *Fifteenth International Conference on Artificial Intelligence and Statistics*. Volume 22. JMLR Proceedings. *JMLR.org*, pages 511–519.
Snoek, J., O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams (2015). "Scalable Bayesian Optimization Using Deep Neural Networks". *Proceedings of the 32nd ICML*. Edited by F. Bach and D. Blei. Volume 37. Proceedings of Machine Learning Research. Lille, France: *PMLR*, pages 2171–2180.
Ritter, H., A. Botev, and D. Barber (2018). "A Scalable Laplace Approximation for Neural Networks". *International Conference on Learning Representations*.
Martens, J. and R. Grosse (2015). "Optimizing Neural Networks with Kronecker-factored approximate curvature". *ICML*.
Graves, A. (2011). "Practical Variational Inference for Neural Networks". *Advances in Neural Information Processing Systems 24*. Edited by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. *Curran Associates, Inc.*, pages 2348–2356.

Blundell, C., J. Cornebise, K. Kavukcuoglu, and D. Wierstra (2015). "Weight uncertainty in neural network". *ICML*. PMLR, pages 1613–1622.

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

Hendrycks, D. and K. Gimpel (2016). "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". *arXiv* abs/1610.02136. arXiv: 1610.02136.

Gibbs, M. N. (1997). "Bayesian Gaussian Processes for Regression and Classification". PhD thesis. University of Cambridge.

Lu, Z., E. Ie, and F. Sha (2020). "Uncertainty Estimation with Infinitesimal Jackknife, Its Distribution and Mean-Field Approximation". *CoRR* abs/2006.07584. arXiv: 2006.07584.

Dangel, F., F. Kunstner, and P. Hennig (2020). "BackPACK: Packing more into Backprop". *International Conference on Learning Representations*.

Spiegelhalter, D. J. and S. L. Lauritzen (1990). "Sequential updating of conditional probabilities on directed graphical structures". *Networks* 20.5, pages 579–605.

MacKay, D. J. (1992). "The evidence framework applied to classification networks". *Neural computation* 4.5, pages 720–736.

Titsias, M. (2016). "One-vs-each approximation to softmax for scalable estimation of probabilities". *NIPS*.

Ahmed, A. and E. Xing (2007). "On tight approximate inference of the logistic-Normal topic admixture model". *Proceedings of the 11th Tenth International Workshop on Artificial Intelligence and Statistics*.

Braun, M. and J. McAuliffe (2010). "Variational inference for large-scale models of discrete choice". *Journal of the American Statistical Association* 105.489, pages 324–335.

Wu, A., S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt (2018). "Fixing Variational Bayes: Deterministic Variational Inference for Bayesian Neural Networks". *arXiv* abs/1810.03958. arXiv: 1810.03958.

Haussmann, M., S. Gerwinn, and M. Kandemir (2019). "Bayesian Evidential Deep Learning with PAC Regularization". arXiv: 1906.00816 [stat.ML].

Malinin, A. and M. Gales (2018). "Predictive Uncertainty Estimation via Prior Networks". *Advances in Neural Information Processing Systems*, pages 7047–7058.

– (2019). "Reverse KL-Divergence Training of Prior Networks: Improved Uncertainty and Adversarial Robustness". *Advances in Neural Information Processing Systems*, pages 14520–14531.

Sensoy, M., L. Kaplan, and M. Kandemir (2018). "Evidential Deep Learning to quantify Classification Uncertainty". *Advances in Neural Information Processing Systems*, pages 3179–3189.

Malinin, A., B. Mlodozeniec, and M. Gales (2019). "Ensemble Distribution Distillation". arXiv: 1905.00076 [stat.ML].

Vadera, M. P., B. Jalaian, and B. M. Marlin (2020). "Generalized Bayesian Posterior Expectation Distillation for Deep Neural Networks". arXiv: 2005.08110 [cs.LG].

Louizos, C. and M. Welling (2016). "Structured and efficient Variational Deep Learning with Matrix Gaussian Posteriors". *ICML*.

Sun, S., C. Chen, and L. Carin (2017). "Learning structured Weight Uncertainty in Bayesian Neural Networks". *Artificial Intelligence and Statistics*, pages 1283–1292.

Newman, M. (2018). "Networks". *Oxford university press*.

Liu, J., J. Wang, and B. Liu (2020). "Community detection of multi-Layer attributed networks via penalized alternating factorization". *Mathematics* 8.2, page 239.

Xu, S., Y. Zhen, and J. Wang (2023). "Covariate-assisted community detection in multi-layer networks". *Journal of Business & Economic Statistics* 41.3, pages 915–926.

Pei, Z., X. Zhang, F. Zhang, and B. Fang (2018). "Attributed multi-layer network embedding". *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, pages 3701–3710.

Cen, Y., X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang (2019). "Representation learning for attributed multiplex heterogeneous network". *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1358–1368.

Cao, J., D. Jin, L. Yang, and J. Dang (2018). "Incorporating network structure with node contents for community detection on large networks using deep learning". *Neurocomputing* 297, pages 71–81.

Park, C., D. Kim, J. Han, and H. Yu (2020). "Unsupervised attributed multiplex network embedding". *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 34. 04, pages 5371–5378.

Han, B., Y. Wei, L. Kang, Q. Wang, and Y. Yang (2022). "Node Classification in Attributed Multiplex Networks Using Random Walk and Graph Convolutional Networks". *Frontiers in Physics* 9, page 763904.

Martirano, L., L. Zangari, and A. Tagarelli (2022). "Co-MLHAN: contrastive learning for multilayer heterogeneous attributed networks". *Applied Network Science* 7.1, pages 1–44.

Goldenberg, A., A. X. Zheng, S. E. Fienberg, E. M. Airoldi, et al. (2010). "A survey of statistical network models". *Foundations and Trends in Machine Learning* 2.2, pages 129–233.

Peel, L., T. P. Peixoto, and M. De Domenico (2022). "Statistical inference links data and theory in network science". *Nature Communications* 13.1, page 6794.

Fortunato, S. (2010). "Community detection in graphs". *Physics reports* 486.3-5, pages 75–174.

Ranganath, R., S. Gerrish, and D. Blei (2014). "Black box variational inference". *Artificial intelligence and statistics*. PMLR, pages 814–822.

Tran, D., R. Ranganath, and D. M. Blei (2016). "Variational Gaussian Process". *4th International Conference on Learning Representations, ICLR 2016*.

Valera, I., M. F. Pradier, M. Lomeli, and Z. Ghahramani (2020). "General latent feature models for heterogeneous datasets". *The Journal of Machine Learning Research* 21.1, pages 4027–4075.

Nazabal, A., P. M. Olmos, Z. Ghahramani, and I. Valera (2020). "Handling incomplete heterogeneous data using vaes". *Pattern Recognition* 107, page 107501.

Tallberg, C. (2004). "A Bayesian approach to modeling stochastic blockstructures with covariates". *Journal of Mathematical Sociology* 29.1, pages 1–23.

Yang, J., J. McAuley, and J. Leskovec (2013). "Community detection in networks with node attributes". *2013 IEEE 13th international conference on data mining*. IEEE, pages 1151–1156.

Hric, D., T. P. Peixoto, and S. Fortunato (2016). "Network structure, metadata, and the prediction of missing nodes and annotations". *Physical Review X* 6.3, page 031038.

Newman, M. E. and A. Clauset (2016). "Structure and inference in annotated networks". *Nature communications* 7.1, page 11863.

White, A. and T. B. Murphy (2016). "Mixed-membership of experts stochastic blockmodel". *Network Science* 4.1, pages 48–80.

Stanley, N., T. Bonacci, R. Kwitt, M. Niethammer, and P. J. Mucha (2019). "Stochastic block models with multiple continuous attributes". *Applied Network Science* 4.1, pages 1–22.

Fajardo-Fontiveros, O., R. Guimerà, and M. Sales-Pardo (2022). "Node metadata can produce predictability crossovers in network inference problems". *Physical Review X* 12.1, page 011010.

Contisciani, M., E. A. Power, and C. De Bacco (2020). "Community detection with node attributes in multilayer networks". *Scientific Reports* 10.1, page 15736.

Hobbhahn, M. and P. Hennig (2021). "Laplace Matching for fast Approximate Inference in Latent Gaussian Models". *arXiv preprint arXiv:2105.03109*.

De Bacco, C., E. A. Power, D. B. Larremore, and C. Moore (2017). "Community detection, link prediction, and layer interdependence in multilayer networks". *Physical Review E* 95.4, page 042317.

McCullagh, P. (2019). "Generalized linear models". *Routledge*.

Kucukelbir, A., D. Tran, R. Ranganath, A. Gelman, and D. M. Blei (2017). "Automatic differentiation variational inference". *Journal of machine learning research*.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the royal statistical society: series B (methodological)* 39.1, pages 1–22.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017b). "Variational inference: A review for statisticians". *Journal of the American statistical Association* 112.518, pages 859–877.

Wand, M. P., J. T. Ormerod, S. A. Padoan, and R. Frühwirth (2011). "Mean Field Variational Bayes for Elaborate Distributions". *Bayesian Analysis* 6.4, pages 847–900.

Benamou, J.-D., G. Carlier, M. Cuturi, L. Nenna, and G. Peyré (2015). "Iterative Bregman projections for regularized transportation problems". *SIAM Journal on Scientific Computing* 37.2, A1111–A1138.

Coz, C. L., A. Tantet, R. Flamary, and R. Plougonven (2023). "A barycenter-based approach for the multi-model ensembling of subseasonal forecasts". *arXiv:2310.17933*.

Power, E. A. (2015). "Building bigness: Religious practice and social support in rural South India". *Stanford University*.
– (2017). "Social support networks and religiosity in rural South India". *Nature Human Behaviour* 1.3, page 0057.
Power, E. A. and E. Ready (2018). "Building bigness: Reputation, prominence, and social capital in rural South India". *American Anthropologist* 120.3, pages 444–459.
Newman, M. E. (2002). "Assortative mixing in networks". *Physical review letters* 89.20, page 208701.
Badalyan, A., N. Ruggeri, and C. De Bacco (2023). "Hypergraphs with node attributes: structure and inference". *arXiv preprint arXiv:2311.03857*.
Kingma, D. P. and M. Welling (2013). "Auto-Encoding Variational Bayes". arXiv: 1312.6114 [stat.ML].
Maddison, C. J., A. Mnih, and Y. W. Teh (2017). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". arXiv: 1611.00712 [cs.LG]. URL: https://arxiv.org/abs/1611.00712.
Tokui, S. and I. sato (2016). "Reparameterization trick for discrete variables". arXiv: 1611.01239 [stat.ML]. URL: https://arxiv.org/abs/1611.01239.
Wilson, J. T., R. Moriconi, F. Hutter, and M. P. Deisenroth (2017). "The reparameterization trick for acquisition functions". arXiv: 1712.00424 [stat.ML]. URL: https://arxiv.org/abs/1712.00424.
Johnson, M. J., D. Duvenaud, A. B. Wiltschko, S. R. Datta, and R. P. Adams (2016). "Composing graphical models with neural networks for structured representations and fast inference". arXiv: 1603.06277 [stat.ML].
Kobyzev, I., S. J. D. Prince, and M. A. Brubaker (2019). "Normalizing Flows: An Introduction and Review of Current Methods". arXiv: 1908.09257 [stat.ML].
Bengio, Y., S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio (2021). "GFlowNet Foundations". arXiv: 2111.09266 [cs.LG].
Hu, E. J., M. Jain, E. Elmoznino, Y. Kaddar, G. Lajoie, Y. Bengio, and N. Malkin (2023). "Amortizing intractable inference in large language models". arXiv: 2310.04363 [cs.LG].
Stegle, O., C. Lippert, J. M. Mooij, N. D. Lawrence, and K. Borgwardt (2011). "Efficient inference in matrix-variate Gaussian models with \iid observation noise". *Advances in Neural Information Processing Systems 24*. Edited by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. *Curran Associates, Inc.*, pages 630–638.
Wilson, A. G., Z. Hu, R. Salakhutdinov, and E. P. Xing (2016). "Deep Kernel Learning". *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Edited by A. Gretton and C. C. Robert. Volume 51. Proceedings of Machine Learning Research. Cadiz, Spain: *PMLR*, pages 370–378.
Brosse, N., C. Riquelme, A. Martin, S. Gelly, and É. Moulines (2020). "On Last-Layer Algorithms for Classification: Decoupling Representation from Uncertainty Estimation". *arXiv preprint arXiv:2001.08049*.
LeCun, Y. (1998). "THE MNIST DATABASE of handwritten digits". *http://yann.lecun.com/exdb/mnist/*.
Xiao, H., K. Rasul, and R. Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". *arXiv* abs/1708.07747. arXiv: 1708.07747.
Bulatov, Y. (2011). "notMNIST dataset". URL: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.
Clanuwat, T., M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha (2018). "Deep Learning for Classical Japanese Literature". *arXiv* abs/1812.01718. arXiv: 1812.01718.
Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.