Computer Science

Advances in distributed Bayesian inference and graph neural networks

Diego Mesquita

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 166/2021

Advances in distributed Bayesian inference and graph neural networks

Diego Mesquita

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, Remote connection link https://aalto.zoom.us/j/6031768727, on 24th November 2021 at 12:00 Noon.

Aalto University
School of Science
Computer Science
Probabilistic Machine Learning (PML)

Aalto University publication series
DOCTORAL DISSERTATIONS 166/2021

ISBN 978-952-64-0608-4 (printed)
ISBN 978-952-64-0609-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-0609-1

Unigrafia Oy
Helsinki 2021

Finland

NORDIC SWAN ECOLABEL
Printed matter
4041-0619

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author Diego Mesquita	
Name of the doctoral dissertation Advances in distributed Bayesian inference and graph neural networks	
Publisher School of Science	
Unit Computer Science	
Series Aalto University publication series DOCTORAL DISSERTATIONS 166/2021	
Field of research	
Manuscript submitted 18 August 2021	**Date of the defence** 24 November 2021
Permission for public defence granted (date) 19 October 2021	**Language** English
☐ **Monograph** [✓ ☒ **Article dissertation**]	☐ **Essay dissertation**

Abstract

Bayesian statistics and graph neural networks comprise a bag of tools widely employed in machine learning and applied sciences. The former rests on solid theoretical foundations, but its application depends on techniques that scale poorly as data increase. The latter is notorious for large-scale applications (e.g., in bioinformatics and natural language processing), but is largely only based on empirical intuitions. This thesis aims to i) broaden the scope of applications for Bayesian inference, and ii) deepen the understanding of core design principles of graph neural networks.

First, we focus on distributed Bayesian inference under limited communication. We advance the state-of-the-art of embarrassingly parallel Markov chain Monte Carlo (MCMC) with a novel method that leverages normalizing flows as density estimators. On the same front, we also propose an extension of stochastic gradient Langevin dynamics for federated data, which are inherently distributed in a non-IID manner and cannot be centralized due to privacy constraints.

Second, we develop a methodology for meta-analysis which allows the combination of Bayesian posteriors from different studies. Our approach is agnostic to study-specific complexities, which are all encapsulated in their respective posteriors. This extends the application of Bayesian meta-analysis to likelihood-free posteriors, which would otherwise be challenging. Our method also enables us to reuse posteriors from computationally costly analyses and update them post-hoc, without rerunning the analyses.

Finally, we revisit two popular graph neural network components: spectral graph convolutions and pooling layers. Regarding convolutions, we propose a novel architecture and show that it is possible to achieve state-of-the-art performance by adding a minimal set of features to the most basic formulation of polynomial spectral convolutions. On the topic of pooling, we challenge the need for intricate pooling schemes and show that they do not play a role in the performance of graph networks in relevant benchmarks.

Keywords

ISBN (printed) 978-952-64-0608-4	**ISBN (pdf)** 978-952-64-0609-1	
ISSN (printed) 1799-4934	**ISSN (pdf)** 1799-4942	
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2021
Pages 127	**urn** http://urn.fi/URN:ISBN:978-952-64-0609-1	

Preface

This thesis is the product of my work in the Probabilistic Machine Learning (PML) group at Aalto university, between the years 2017 and 2021. During this time, I was mainly funded by the Helsinki Doctoral Education Network in Information and Communications Technology (HICT). This work was also supported by the Academy of Finland (Flagship programme: Finnish Center for Artificial Intelligence FCAI). Additionally, I am grateful to the Aalto Science-IT Project, whose computational resources were crucial to this thesis' developments.

I am grateful to Prof. Samuel Kaski, my supervisor. His guidance was paramount to my academic development. Working with him has undoubtedly changed the way I plan and conduct research. He also actively helped expand my collaboration network.

I salute my collaborators Amauri Souza, Paul Blomstedt, Hojin Kang, Jou-hui Ho and Jorge Peréz. All of you motivate me to seek ambitious goals and I hope we will have long-lived collaborations.

Espoo, November 8, 2021,

Diego Mesquita

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Embarrassingly Parallel MCMC with deep invertible transformations. In *Uncertainty in Artificial Intelligence*, Tel-Aviv, Israel, July 2019.

II Khaoula el Mekkaoui, Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Federated stochastic gradient Langevin dynamics. In *Uncertainty in Artificial Intelligence*, Online, July 2021.

III Paul Blomstedt, Diego Mesquita, Jarno Lintuusari, Tuomas Sivula, Jukka Corander, and Samuel kaski. Meta-analysis of Bayesian analyses. Submitted to *a journal*, 2020.

IV Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. In *Advances in neural information processing systems*, Online, December 2020.

V Hojin Kang, Jou-hui Ho, Diego Mesquita, Amauri Souza, Jorge Pérez, and Samuel Kaski. Spectral Graph Networks with Constrained Polynomials. Submitted to *a journal*, 2021.

Author's Contribution

Publication I: "Embarrassingly Parallel MCMC with deep invertible transformations"

I come up with the concept for the project and implemented it. I also developed the theory for the work, which was latter checked by Paul Blomstedt. After I wrote an initial version of the manuscript, all authors iterated to refine the manuscript.

Publication II: "Federated stochastic gradient Langevin dynamics"

The idea for this came up after a series of discussions between the authors. The theory for this work was developed by myself and Paul Blomstedt. I also implemented the illustrative examples and the ones for Bayesian metric learning. Writing was a joint task for all authors.

Publication III: "Meta-analysis of Bayesian analyses"

Paul Blomstedt and Samuel Kaski had the original idea for this work. In my first year of PhD, I was invited to join the project. I contributed implementing experiments and helped with some of the theoretical results. I also contributed drawing connections to works in the meta-analysis literature and with writing.

Publication IV: "Rethinking pooling in graph neural networks"

Myself and Amauri Souza had the idea for this project. Together with Amauri Souza, I implemented the experiments and interpreted the results. After this, all authors worked on the text to until we reached the final version.

Publication V: "Spectral Graph Networks with Constrained Polynomials"

Hojin Kang, Jou-hui Ho had the initial idea for this graph neural network architecture. Hojin and Jou-hui were mostly responsible for implementation, with little help from me and other authors. Myself, Amauri and Jorge refined the formulation of the method and suggested a series of experiments to validate and better understand why and how the model works. Writing was a joint effort.

1. Introduction

This thesis covers selected topics in Bayesian statistics and graph neural networks (GNNs). In the realm of Bayesian statistics, we aim to extend the applicability of Bayesian methods by addressing computational issues and other fundamental limitations. In the scope of GNNs, our goal is to revisit well-established concepts to better understand which design choices are beneficial to build state-of-the-art GNNs.

The contributions of this thesis are organized into three self-contained chapters, addressing individual research questions. Each chapter motivates our research directions, briefly covers the necessary background, and addresses the main contributions therein. Each chapter also contains a section assessing the extent to which we answered the research questions. The thesis closes with a discussion of broader impacts and directions for future works. The remainder of this introduction presents our research questions and provides an overview of their respective chapters.

Research Question 1: *Real-world computing systems are often linked by unreliable communication networks, making communication costs a bottleneck for many applications. Can we design efficient methods for distributed Bayesian inference under severe communication constraints?*

Chapter 2 focuses on using distributed computations to scale-up Bayesian inference while adhering to communication constraints, covering Publication I and Publication II. Publication I employs normalizing flows [Papamakarios et al., 2021] to improve embarrassingly parallel Markov chain Monte Carlo, which uses a divide-and-conquer strategy to speed-up posterior sampling. Publication II proposes an extension of stochastic gradient Langevin dynamics [Welling and Teh, 2011] for federated learning, a novel setting in which data are inherently distributed and privacy concerns prevent its disclosure to a centralizing server.

Research Question 2: *The natural outcome of a Bayesian analysis is a posterior distribution. However, Bayesian meta-analysis traditionally depends on* *summaries extracted from data. Can we combine the results of multiple Bayesian studies, i.e. posterior distributions, into a meta-analysis in a principled manner?*

Chapter 3 presents a framework for meta-analysis that allows the combination of Bayesian posteriors, proposed in Publication III. Despite posteriors being the natural outcome of a Bayesian study, Bayesian meta-analysis traditionally combines results summarized as point estimates. To the best of our knowledge, Publication III is the first work addressing the combination of full posteriors.

Research Question 3: *GNNs are often based on intuitions from traditional deep learning models (e.g. Convolutional neural networks). As a consequence, they might inherit unnecessary complexities, which may not be useful for graph domains. Can simple GNNs, with minimalist designs, perform as well as state-of-the-art models?*

Chapter 4 revisits two basic concepts in GNNs: spectral graph convolutions and graph pooling. We first cover Publication IV, which shows it is possible to achieve state-of-the-art performance by adding a minimal set of features to the most basic formulation of polynomial spectral GNNs. Subsequently, we cover Publication V which revisits popular pooling methods and shows they usually do not contribute to the performance of successful GNNs.

2. Scalable Bayesian inference

Markov Chain Monte Carlo (MCMC) algorithms are a cornerstone of practical Bayesian analysis. Nonetheless, accommodating large and distributed data is still a challenge, especially when communication is a premium. Publication I proposes an embarrassingly parallel MCMC strategy using normalizing flows. Publication II develops a stochastic gradient Langevin dynamics (SGLD) sampler for federated learning.

The remaining of this chapter is organized as follows. Section 2.1 provides a brief background of embarrassingly parallel MCMC [Neiswanger et al., 2014] and an overview of Publication I. Section 2.2 reviews serial and distributed SGLD and shows how Publication II adapts it for federated data.

2.1 Embarrassingly parallel MCMC using normalizing flows

Embarrassingly parallel MCMC methods employ a divide-and-conquer strategy to obtain samples from the Bayesian posterior

$$p(\theta|\mathscr{D}) \propto p(\theta)p(\mathscr{D}|\theta), \tag{2.1}$$

where $p(\theta)$ is a prior, $p(\mathscr{D}|\theta)$ is a likelihood function and $\mathscr{D}$ denotes the data. The general idea is to break the global inference into smaller tasks and combine their results, requiring coordination only in the final aggregation stage. First, we partition $\mathscr{D}$ into S shards $\mathscr{D}_1, \ldots, \mathscr{D}_S$. Then, we factorize our target posterior as

$$p(\theta|\mathscr{D}) \propto \prod_{s=1}^{S} p(\theta)^{1/S} p(\mathscr{D}_s|\theta), \tag{2.2}$$

and sample from the right-hand-side factors, referred to as *subposteriors*, in parallel using an MCMC algorithm of choice. Subsequently, we send the sub-posterior samples to a coordinating server for an aggregation step. The core challenge in this framework is devising combination/aggregation strategies that are both accurate and computationally efficient.

The pioneering work of Scott et al. [2016] uses weighted averages of subposterior samples to approximate the target posterior. Neiswanger et al. [2014] propose parametric, semi-parametric and non-parametric strategies, the two former depend on fitting kernel density estimators to subposterior samples. Wang et al. [2015] use random partition trees to approximate the posterior with a multidimensional histogram. Nemeth and Sherlock [2018] fit Gaussian process approximations to the log-subposteriors and take the product of their expected values. Except for the parametric method, which imposes overly simplistic local approximations that generally result in poor approximations of the posterior, all of the aforementioned approaches require the subposterior samples to be centralized, incurring extensive communication costs. In fact, communication costs have been altogether ignored in the literature before Publication I. Furthermore, sampling from the approximate posterior can become difficult, requiring expensive additional MCMC steps to obtain samples from the combined posterior.

Contribution. To alleviate these problems, Publication I proposes i) using real non-volume-preserving transformations (NVP) [Dinh et al., 2014] to approximate the subposteriors, and ii) a simple importance sampling scheme to combine subposteriors. Notably, since we only send the real NVPs to the server, our strategy results in communication costs that are constant in the number of subposterior samples. This is an especially appealing feature when communication between machines holding data shards and the server is expensive or limited.

Real NVPs are a special case of normalizing flows [Papamakarios et al., 2021], a family of invertible generative models that can also be used for density estimation. While normalizing flows are easy to sample from, evaluating densities can be arbitrarily expensive. Real NVPs are specially useful to us since they are both easy to sample from and to evaluate.

Once we obtain the sets of samples $\mathscr{M}_1, \ldots, \mathscr{M}_S$ from each of the S subposteriors, we use them to fit a series of normalizing flows $\widehat{p}_1, \ldots, \widehat{p}_S$. Then, we use their product $\widehat{p} = \prod_{s=1}^{S} \widehat{p}_s$ to approximate the posterior. While normalizing flows are easy to sample from, there is no straightforward recipe to sample from their product. Therefore, we propose an Importance Sampling (IS) algorithm [Geweke, 1989] to compute the expectation of an arbitrary test function h. More specifically, given a set of T samples drawn from any $\widehat{p}_s$, we build our IS estimate as:

$$\overline{h}(\theta) = \sum_{t=1}^{T} w_t h(\theta_t), \tag{2.3}$$

where $w_1, \ldots, w_T$ are self-normalized importance weights such that

$$w_t \propto \frac{\prod_{s'=1}^{S} \widehat{p}_{s'}(\theta_t)}{\widehat{p}_s(\theta_t)} \quad \forall t = 1 \ldots T. \tag{2.4}$$

Our IS framework exploits two key properties of real NVP transformations — ease of evaluation and sampling — and avoids the overhead of running still more MCMC chains to sample from the aggregated posterior $\widehat{p}(\theta)$, which might be a challenging target due to the neural networks parameterizing the normalizing flows.

While importance sampling estimates can be unreliable if their variance is very high or infinite, we prove that, under mild assumptions, our importance sampling scheme is stable, i.e., $\overline{h}(\theta)$ has finite variance. Additionally, we can use the same importance weights to sample from the approximate posterior using Sampling Importance Resampling [SIR Rubin, 1987].

Experimental results show that our method outperforms the previous state-of-the-art in several settings, including heterogeneous subposteriors and intricate-shaped, multi-modal or high-dimensional posteriors.

2.2 Stochastic-gradient MCMC for federated data

Langevin dynamics [Neal, 2011] is a family of MCMC methods which utilizes the gradient of the log-posterior

$$\nabla \log p(\theta|\mathscr{D} = (x_1, \ldots, x_N)) = \nabla \log p(\theta) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta), \tag{2.5}$$

to generate proposals in a Metropolis-Hastings sampling scheme. A problem with this approach, however, is that the exact computation of the gradient in Equation 2.5 can be prohibitive for large datasets. To mitigate this problem, Welling and Teh [2011] propose stochastic gradients Langevin dynamics (SGLD), a method that uses a mini-batch $\mathscr{B}_t$ of size m to approximate the full-data gradient at each time-step t.

SGLD draws samples from the target distribution using a stochastic gradient update of the form

$$\theta_{t+1} = \theta_t + \frac{h_t}{2} v(\theta_t) + \eta_t, \tag{2.6}$$

in which h_t is the step size, η_t is a noise variable sampled from $\mathscr{N}(0, h_t I)$ and where the velocity function v is given by

$$v(\theta_t) = \nabla \log p(\theta_t) + \frac{N}{m} \nabla \log p(\mathscr{B}_t|\theta_t). \tag{2.7}$$

To extend SGLD to distributed settings – where $\mathscr{D}$ is partitioned into shards

$\mathscr{D}_1, \ldots, \mathscr{D}_S$ – Ahn et al. [2014] propose an extension of SGLD coined distributed SGLD (DSGLD).

The main idea in DSGLD is to sample each mini-batch *within* a single shard $\mathscr{D}_{s_t}$ at each time-step t. The shard index s_t itself is sampled by a scheduler with probability f_s, with $\sum_{s=1}^{S} f_s = 1$ and $f_s > 0$ for all s. This results in the update

$$\theta_{t+1} = \theta_t + \frac{h_t}{2} v_{s_t}(\theta_t) + \eta_t, \tag{2.8}$$

in which v_{s_t} is an unbiased gradient estimator given by

$$v_{s_t}(\theta_t) = \nabla \log p(\theta_t) + \frac{N_{s_t}}{f_{s_t} m} \nabla \log p(\mathscr{B}_t | \theta_t), \tag{2.9}$$

and where N_{s_t} denotes the size of shard $\mathscr{D}_{s_t}$, chosen at time t. Intuitively, if we choose the mini-batch $\mathscr{B}_t$ of m data points uniformly at random from $\mathscr{D}_{s_t}$, then N_{s_t}/m scales $\nabla \log p(\mathscr{B}_t|\theta_t)$ to be an unbiased estimator for $\nabla \log p(\mathscr{D}_{s_t}|\theta_t)$. In turn, $f_{s_t}^{-1}$ further scales this gradient to be an unbiased estimator of $\nabla \log p(\mathscr{D}|\theta_t)$.

It is worth noting that, while data are distributed, we can still understand DSGLD chains as entirely serial procedures. In practice, however, distributed settings are naturally amenable to running multiple chains simultaneously.

A downside of this approach is that the chain state θ_t needs to be transferred between workers at each time-step t. To avoid this constant communication, Ahn et al. [2014] propose taking multiple update steps within the same shard before moving to another computing node. Nonetheless, this reduction in communication costs comes at the expense of some loss in asymptotic accuracy.

Contribution. In Publication II, we propose an extension of SGLD for federated scenarios [Konecný et al., 2016]. Notably, federated data arise independently in different devices, and communication or privacy constraints prevent it from being disclosed to a server. As a consequence, data are often partitioned in a non-IID fashion.

We show that distributed methods such as vanilla DSGLD are inappropriate for the federated non-IID regime. In practice, this regime significantly amplifies the variance of stochastic gradients (e.g. Figure 2.1). In turn, this can lead to poor mixing rates and slow convergence[Dubey et al., 2016]. Additionally, in federated settings, we want to avoid frequent communication, which can cause DSGLD to diverge from the target posterior even for very simple models [Ahn et al., 2014].

To alleviate both these problems, we propose *conducive gradients*, a simple mechanism that combines local likelihood approximations to correct gradient updates. Given the approximations $q_1(\theta), \ldots, q_S(\theta)$ for the likelihood contributions

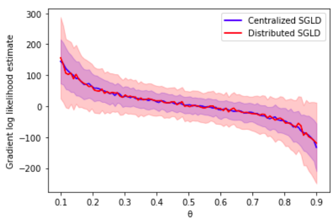

Figure 2.1. Comparison between gradient estimators using centralized (SGLD) and distributed data (DSGLD) for a model with Bernoulli likelihood and uniform prior. We computed gradients using 5 samples from a total of 30 observations generated from fair coin tosses. For DSGLD, we simulate the federated non-IID regime splitting data between 3 equally-available shards of same size but distinct means – 0.1, 0.5 and 0.9. The confidence bars correspond to one standard deviation. DSGLD shows higher variance than SGLD even for this simple case.

$p(\mathscr{D}_1|\theta),\dots,p(\mathscr{D}_S|\theta)$, we define the conducive gradient for a shard s as

$$g_s(\theta) = \nabla \log q(\theta) - \frac{1}{f_s} \nabla \log q_s(\theta), \tag{2.10}$$

where $q(\theta) = \prod_s q_s(\theta)$ can be seen as a crude approximation of the posterior.

We then add the conducive gradient g_s to the unbiased estimate in Equation 2.8 to a novel method, which we call federated SGLD (FSGLD). More specifically the FSGLD update equation is defined as:

$$\theta_{t+1} = \theta_t + \frac{h_t}{2}\left(v_{s_t}(\theta_t) + g_{s_t}(\theta_t)\right) + \eta_t. \tag{2.11}$$

To avoid computational overhead, we must choose tractable approximations q_s for which gradient computations are inexpensive. Exponential family distributions are especially convenient for this purpose. Furthermore, they are closed under product operations, enabling us to compute $\nabla \log q(\theta_t)$ in a single gradient evaluation instead of S. This keeps the additional cost of our method negligible even when $S \gg m$.

In Publication II, we compute q_s by first drawing from $p_s(\theta) \propto p(\mathscr{D}_s|\theta)$ locally using SGLD, and then using the resulting samples to compute the parameters of an exponential family approximation. To avoid communication overhead, we compute $q_1,\dots,q_S$ in parallel for each data shard and send the approximations to the coordinating server once, before the FSGLD iterations take place.

Our experiments show that FSGLD outperforms DSGLD in federated non-IID scenarios, and that it converges to the true posterior in cases where DSGLD fails. We also provide convergence bounds for FSGLD and use these results to gain further insight on how to weight the quality of the local likelihood approximations. Furthermore, we provide analysis for DSGLD since no formal analysis is available in the literature. We use well-established analyses of convergence for SGLD in serial settings [Chen et al., 2015, Nagapetyan et al., 2017, Baker et al., 2019, Teh et al., 2016, Vollmer et al., 2016] as a starting point due to their relatively straightforward formulation.

2.3 Revisiting research question I

— Can we design efficient methods for distributed Bayesian inference under severe communication constraints?

The contributions in this chapter show two efficient inference schemes for settings with strong communication constraints. Besides making embarrassingly parallel MCMC capable of handling more complex posteriors, Publication I further reduces its communication cost. More specifically, we only send normalizing flows to the server — instead of the entire subposterior chains — detaching the communication cost from the number of subposterior samples. Also, Publication II extends the well-known SGLD to cope with federated settings, where data is often distributed in non-IID fashion and communication is a premium.

3. Bayesian meta-analysis

Meta-analysis encompasses a collection of approaches that aim to combine results from multiple related statistical analyses. In the standard formulation, these results are summary statistics computed from data, a typical example being point estimates for some treatment's effect size. For the combination of point estimates, there exists a well-established Bayesian methodology and a large body of literature [see e.g. Higgins et al., 2009, and references therein]. However, while a Bayesian analysis' natural outcome is a posterior distribution, the analogous task of combining posteriors has received little attention.

Publication III presents a principled framework for meta-analysis of Bayesian posteriors. From a broader perspective, this contribution can also be seen as a strategy to handle uncertain or 'soft' evidence [Diaconis and Zabell, 1982, Jeffrey, 2004, Smets, 1993, Zhao and Osherson, 2010]. Section 3.1 provides a review of Bayesian *random effects meta-analysis* (REMA) and shows how we extend it to combine posterior distributions instead of summary statistics.

3.1 Meta-analysis of Bayesian analyses

In standard Bayesian REMA, we observe a summary statistic D_j, designed to provide information about the local effect θ_j, for each study $j = 1, \dots, J$. Furthermore, it is common to translate the relatedness between studies as exchangeability between local effects [Gelman et al., 2013], conditioning them on the overall effect φ. Denoting by Q the prior distribution over φ, this leads to the following hierarchical model:

$$\varphi \sim Q$$

$$\theta_j \sim P_\varphi$$

$$D_j \sim F_{\theta_j},$$

where P_φ is the conditional distribution of θ_j given φ, and F_{θ_j}'s is typically modeled as $\mathcal{N}(\theta_j, \hat{\sigma}_j^2)$, with $\hat{\sigma}_j^2$ estimated from data. One of the primary goals of the above model is to estimate the overall effect φ, for which the marginal posterior density is given by

$$q(\varphi|D_1,\dots,D_J) \propto \prod_{j=1}^{J} \left[\int f(D_j|\theta_j) p(\theta_j|\varphi) d\theta_j \right] q(\varphi). \tag{3.1}$$

Contribution. Publication III considers the setting where, instead of summary statistics D_j, we have posterior distributions with densities $\pi_j(\theta_j)$ from each of J studies, based on which we wish to update our prior knowledge about the global effect φ, in analogy with Equation (3.1). Since π_j informs us directly about the local effect θ_j, we propose marginalizing the uncertainty around these effects as they appear in the likelihood, which leads us to update $q(\varphi)$ as

$$q^*(\varphi) \propto \prod_{j=1}^{J} \left[\int p(\theta_j|\varphi) \pi_j(\theta_j) d\theta_j \right] q(\varphi), \tag{3.2}$$

which differs from Equation (3.2) only in the likelihood $f(D_j|\theta_j)$ being replaced by the study-specific posterior $\pi_j(\theta_j)$.

To derive an update equation for the local effects, we first note that Equation (3.2) induces a joint distribution all effects:

$$p^*(\theta_1,\dots,\theta_J,\varphi) \propto \prod_{j=1}^{J} \left[p(\theta_j|\varphi) \pi_j(\theta_j) d\theta_j \right] q(\varphi), \tag{3.3}$$

and then marginalize with respect to all but the effect $\theta_{j'}$ we are interested to get the update

$$\pi_{j'}^*(\theta_{j'}) \propto \int p(\theta_{j'}|\varphi) \pi_{j'}(\theta_{j'}) \prod_{j=1, j\neq j'}^{J} \left[\int p(\theta_j|\varphi) \pi_j(\theta_j) d\theta_j \right] q(\varphi) d\varphi. \tag{3.4}$$

A major advantage of this meta-analysis framework is its composability. In particular, the study-specific inferences resulting in $\pi_j(\theta_j)$'s can be carried independently of the combination model, $q(\varphi)\prod_{j=1}^{J} p(\theta_j|\varphi)$, which is designed by the meta-analyst. This means that, unlike in conventional meta-analysis, all study-level complexities are hidden 'under the hood' and need not explicitly be included in the meta-analysis. For instance, in likelihood-free models [e.g. Lintusaari et al., 2017, Marin et al., 2012], the data can typically be summarized by a number of different statistics but there is no closed-form likelihood to relate these to the parameter of interest. In our framework, likelihood-free inferences can be conducted separately for each study using *approximate Bayesian computation*, after which the resulting posteriors are directly combined in a meta-analysis.

With this, we are able to reuse results from computationally costly analyses and update them without having to rerun the analyses themselves.

In addition to its intuitive nature, we prove that our update rule — in Equation (3.2) — retains some basic properties of standard Bayesian inference, such as order-invariance in successive updates and posterior concentration as $J \to \infty$. Publication III also demonstrates our framework combining results from likelihood-free Bayesian analyses, which would be difficult to carry out using standard methodology.

3.2 Revisiting reasearch question II

— *Can we combine the results of multiple Bayesian studies, i.e. posterior distributions, into a meta-analysis in a principled manner?*

Publication III provides a positive answer to this question. Our work builds on exchangeability between local effects and guarantees fundamental properties of the Bayesian framework. Furthermore, it allows statisticians to define their meta-analysis model *as if* the local effects, over which the study posteriors are defined, were observed quantities. Notably, Publication III is the first work addressing the meta-analysis of posterior distributions.

4. Graph neural networks

Graph neural networks (GNNs) are the de facto standard for machine learning in graph domains. Similarly to Convolutional Neural Networks, GNNs are primarily composed of convolutions and pooling layers. However, differently from images, there is no unique characterization of these operations on graphs. As a consequence, a flurry of works propose arbitrarily complex designs for GNNs. Both contributions in this chapter strive for simpler architectures.

Publication IV revisits the basic formulation of GNNs based on spectral convolutions and proposes a simple method that outperforms previous spectral GNNs and rivals state-of-the-art models on relevant large-scale benchmarks. Publication V challenges the need for intricate pooling mechanisms and shows that spurious strategies, e.g. random assignments, lead to similar performance.

After introducing notation, the remaining of this chapter is organized as follows. Section 4.1 reviews spectral graph networks and covers Publication IV. Section 4.2 reviews representative pooling methods and discusses the findings from Publication V.

Notation We define a graph $\mathscr{G}$, with nodes $\{1,2,\dots,n\}$, as an ordered pair $(\boldsymbol{A},\boldsymbol{X})$ where $\boldsymbol{A} \in \{0,1\}^{n\times n}$ denotes a symmetric adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is a matrix of d-dimensional node features. The matrix $\boldsymbol{A}$ defines the graph structure: two nodes i,j are connected if and only if $A_{ij} = 1$. We denote by $\boldsymbol{D}$ the diagonal degree matrix of $\mathscr{G}$, i.e., $D_{ii} := \sum_j A_{ij}$. The normalized graph Laplacian $\boldsymbol{\Delta} := \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ is a symmetric positive semidefinite matrix. Naturally, this Laplacian has eigendecomposition $\boldsymbol{\Delta} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ is an orthonormal matrix with eigenvectors $\boldsymbol{u}_1,\dots,\boldsymbol{u}_n$ and the matrix $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1,\dots,\lambda_n)$ comprises the corresponding eigenvalues (or spectrum) of $\mathscr{G}$, with $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

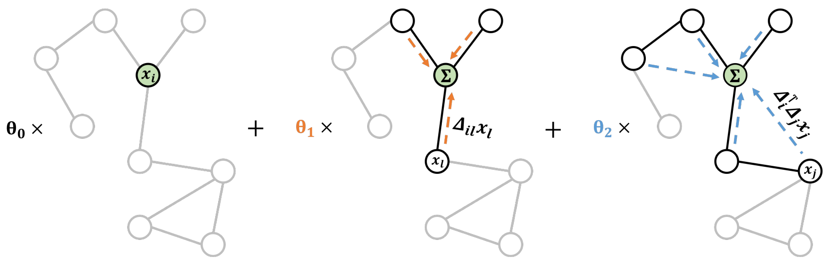

Figure 4.1. Spectral polynomial convolutions. Polynomial convolutions of degree K combines K-hop neighborhood information to update a node's feature/signal. The picture above illustrates how a quadratic polynomial filter $\boldsymbol{g} \star \boldsymbol{x} = \sum_{k=0}^{K} \theta_k \boldsymbol{\Delta}^k \boldsymbol{x}$. — with $K = 2$ — updates the features x_i of a node i. On the node level, the coefficient θ_0 repeats part of the original signal. The first-order term $\theta_1 \boldsymbol{\Delta}$ aggregates information from the immediate neighborhood of node i. Similarly, the second-order $\theta_2 \boldsymbol{\Delta}^2$ term aggregates all nodes with distance at most two.

4.1 Spectral graph networks with constrained polynomials

Let $\boldsymbol{x} \in \mathbb{R}^n$ be a column vector representing a signal on the nodes of $\mathcal{G}$. The graph Fourier transform of $\boldsymbol{x}$ is $\hat{\boldsymbol{x}} = \boldsymbol{U}^\top \boldsymbol{x}$, and its inverse operation is $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$. Furthermore, let $\boldsymbol{g} \in \mathbb{R}^n$ be a graph filter. Using the graph Fourier transform and the convolution theorem [Sandryhaila and Moura, 2013], we can write the convolution between $\boldsymbol{g}$ and a signal $\boldsymbol{x}$ as

$$\boldsymbol{g} \star \boldsymbol{x} = \boldsymbol{U}\left((\boldsymbol{U}^\top \boldsymbol{g}) \odot (\boldsymbol{U}^\top \boldsymbol{x})\right) = \boldsymbol{U}\hat{\boldsymbol{G}}\boldsymbol{U}^\top \boldsymbol{x}, \tag{4.1}$$

where $\hat{\boldsymbol{G}} = \operatorname{diag}(\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_n)$ comprises the spectral filter coefficients $(\hat{g}_i)_{i=1}^n$. The downside of this non-parametric characterization is that it ties the number of filter coefficients to the number of nodes in $\mathcal{G}$. Also, there is no guarantee that $\hat{\boldsymbol{G}}$ leads to localized filters [Bruna et al., 2014].

To circumvent both these problems, it is common to approximate $\hat{\boldsymbol{G}}$ using a polynomial of the eigenvalue matrix $\boldsymbol{\Lambda}$. Without loss of generality, we can represent polynomial spectral filters with base spectrum $\boldsymbol{\Lambda}$ as

$$\hat{\boldsymbol{G}} = \sum_{k=0}^{K} \theta_k \boldsymbol{\Lambda}^k, \tag{4.2}$$

where $\theta_0, \ldots, \theta_K$ are the polynomial coefficients. Under this approximation, we can rewrite the graph convolution in Equation 4.1 as

$$\boldsymbol{g} \star \boldsymbol{x} = \sum_{k=0}^{K} \theta_k \boldsymbol{\Delta}^k \boldsymbol{x}. \tag{4.3}$$

Note that this filtering operation does not explicitly require the eigendecomposition of the graph Laplacian. Hence, this approach is also known as spectrum-free. Figure 4.1 illustrates the result of a polynomial spectral convolution with $K = 2$.

Spectral graph neural networks build upon graph convolutions to achieve a multilayer design that resembles CNNs. Recall that the standard CNN layer comprises: i) filtering operations; ii) a summation over channels; and iii) a nonlinear activation function. Consider now a graph signal $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ with d channels. We can derive a generic polynomial spectral GNN using channel-wise independent polynomial filters (Equation 4.2). Let $\boldsymbol{H}^{(0)} := \boldsymbol{X}$. At layer ℓ, the resulting GNN computes

$$\boldsymbol{H}^{(\ell)} = \phi \left(\sum_{k=0}^{K} \boldsymbol{\Delta}^k \boldsymbol{H}^{(\ell-1)} \boldsymbol{\Theta}_k^{(\ell)} \right), \tag{4.4}$$

where ϕ is a nonlinearity, such as the ReLU function, and $\boldsymbol{\Theta}^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ are the coefficients of the spectral filters.

Despite its intuitive nature, the model in Equation 4.4 was only hinted at by Defferrard et al. [2016], but never was explored in detail. The closest model in the literature is ChebNet [Defferrard et al., 2016]. This network replaces the monomials over $\boldsymbol{\Delta}$ by Chebyshev polynomials of a transformed Laplacian $\tilde{\boldsymbol{\Delta}} = (2/\lambda_n)\boldsymbol{\Delta} - \boldsymbol{I}$. The idea behind this choice is to exploit the orthogonality of Chebyshev polynomials to obtain more stable filters.

As a simplification of ChebNets, Graph Convolutional Networks (GCNs) [Kipf and Welling, 2017] use first-order polynomial filters with a single parameter θ each, such that $\theta = \theta_0 = -\theta_1$. A GCN layer can be expressed as

$$\boldsymbol{H}^{(\ell)} = \mathrm{ReLU}\left(\boldsymbol{A}' \boldsymbol{H}^{(\ell-1)} \boldsymbol{\Theta}^{(\ell)} \right), \tag{4.5}$$

where $\boldsymbol{A}' = (\boldsymbol{D} + \boldsymbol{I})^{-1/2} (\boldsymbol{A} + \boldsymbol{I}) (\boldsymbol{D} + \boldsymbol{I})^{-1/2}$ is the normalized adjacency matrix with added self-loops.

Recent works on spectral GNNs aim to improve ChebNets and GCNs by increasing model flexibility with rational filters. Nonetheless, the exact computation of these filters involves matrix inversion. CayleyNets [Levie et al., 2019] avoid this problem by solving a sequence of linear systems with the Jacobi method. Bianchi et al. [2021] propose a design that approximates autoregressive moving average filters based on recursions derived by Isufi et al. [2017].

Contribution. Departing from the search for flexible graph filters, Publication IV pursues a simpler design for polynomial filters. Publication IV introduces *Polynomial Subspace Net* (PSN), a spectral method that builds on the simplest formulation of polynomial spectral GNNs (see Equation 4.4). Each PSN layer filters all input channels using the same coefficients and combines the filtered channels with a fully-connected feedforward network.

To derive convolutions for PSN, we introduce two simple modifications to the general polynomial filters in Equation 4.2. First, we restrict the range of the polynomial coefficients to the interval $[-1,1]$ using the tanh function. Second, we take a convex combination between the constant component (0-th order) and the higher-order terms. The motivation for these choices is, respectively, i) improving training stability, and ii) allowing PSN to learn an importance for the input signal at each layer. Formally, the resulting modified filter is

$$\hat{\boldsymbol{G}} = \sigma(\theta_0)\boldsymbol{I} + (1-\sigma(\theta_0))\sum_{k=1}^{K}\tanh(\theta_k)\boldsymbol{\Lambda}^k, \tag{4.6}$$

where σ denotes the logistic function, and the $\theta_0, \theta_1, \ldots, \theta_K$ are learned filter parameters. At first glance, these constraints seem to reduce the range of polynomials we can express. However, we can overcome this following the filtering operation with, e.g., a linear layer.

Like standard GNNs, PSN interleaves filtering operations and nonlinear mappings in a multilayer fashion. Figure 4.2 shows the block diagram for a PSN layer. The PSN filtering step applies the polynomial filter in Equation 4.6 to each graph input channel, independently. Subsequently, we apply a multilayer perceptron (MLP) to the filtered signals. Given the initial node features $\boldsymbol{H}^{(0)} := \boldsymbol{X}$, the output of the ℓ-th PSN layer can be recursively written as

$$\boldsymbol{S}^{(\ell)} = \sigma(\theta_0^{(\ell)})\boldsymbol{I} + (1-\sigma(\theta_0^{(\ell)}))\sum_{k=1}^{K}\tanh(\theta_k^{(\ell)})\boldsymbol{\Delta}^k \tag{4.7}$$

$$\boldsymbol{H}^{(\ell)} = \mathrm{MLP}_\ell\big(\boldsymbol{S}^{(\ell)}\boldsymbol{H}^{(\ell-1)}\big). \tag{4.8}$$

Despite its simplicity, experiments show that PSN outperforms complex spectral GNNs and rivals state-of-the-art models on relevant large-scale benchmarks, including datasets from the OGB suite Hu et al. [2020]. Notably, PSN can be seen as an implicitly regularized (low-rank) version of the general convolution presented in Equation 4.4. Publication IV also shows ablation studies supporting our design choices.

4.2 Rethinking pooling in graph neural networks

Many GNN architectures interleave pooling operations between convolutional layers. Intuitively, the convolutional layers aggregate neighborhood information to capture local patterns. In turn, the pooling layers reduce the graph representation while ideally preserving important structural information.

Although strategies for graph pooling come in varied flavors [Murphy et al., 2019, Zhang et al., 2018, Lee et al., 2019, Ying et al., 2018], most GNNs follow a hierarchical scheme in which the pooling regions correspond to graph clusters

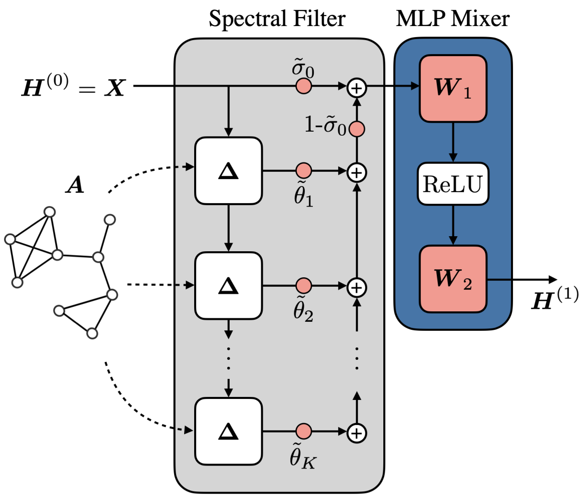

Figure 4.2. Polynomial Subspace Nets (PSNs). Each PSN layer applies the same polynomial spectral filter across all input channels. This filter has constrained coefficients: $\tilde{\sigma}_0 \in [0,1]$ and $\tilde{\theta}_1,\dots,\tilde{\theta}_K \in [-1,1]$. After the filtering, an MLP combines the filtered signals. For clarity, we only denote the first layer ($\boldsymbol{H}^{(0)} = \boldsymbol{X}$) and omit layer indexes in the model parameters.

that are combined to produce a coarser graph [Bruna et al., 2014, Defferrard et al., 2016, Yuan and Ji, 2020, Ying et al., 2018, Khasahmadi et al., 2020, Fey et al., 2018]. Intuitively, these clusters generalize the notion of local neighborhood from traditional CNNs and enable us to coarsen graphs of varying sizes. The cluster assignments can be computed using deterministic clustering algorithms [Bruna et al., 2014, Defferrard et al., 2016] or be learned in an end-to-end fashion [Ying et al., 2018, Khasahmadi et al., 2020]. Also, one can leverage node embeddings [Khasahmadi et al., 2020], graph topology [Dhillon et al., 2007], or both [Ying et al., 2018, Yuan and Ji, 2020], to pool graphs. We refer to these approaches as local pooling.

Alongside attention-based mechanisms [Lee et al., 2019, Knyazev et al., 2019], the notion that clustering is a Vital component of graph pooling has been tremendously influential, resulting in an ever-increasing number of pooling schemes [Ma et al., 2019, Gao and Ji, 2019, Huang et al., 2019, Yuan and Ji, 2020, Khasahmadi et al., 2020]. This reflects the implicit belief that the quality of the cluster assignments, that underlie these pooling methods, is crucial for the performance of GNNs.

Contribution. Publication V studies the influence of local pooling in the success of GNNs. We select a number of models that are popular or claim to achieve state-of-the-art performances and simplify their pooling operators, stripping them of components that enforce clustering. To do so, we either use randomized

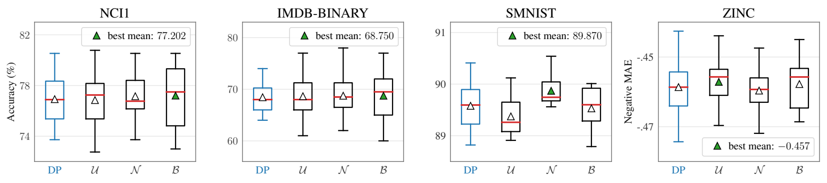

Figure 4.3. Boxplots for DIFFPOOL (DP) and its random variants: Uniform $\mathscr{U}(0,1)$, Normal $\mathscr{N}(0,1)$, and Bernoulli $\mathscr{B}(0.3)$. In all cases, at least one random variants achieves higher average accuracy than DIFFPOOL. Also, random pooling does not consistently lead to higher variance. Strikingly, learned pooling assignments do not contribute to the performance of DIFFPOOL.

cluster assignments or operate on complementary graphs. Surprisingly, our results reveal that the non-local GNN variants perform on par with or better than the original methods.

As an illustrative example, consider DIFFPOOL [Ying et al., 2018], which uses a GNN to learn cluster assignments for graph pooling. At each layer l, DIFFPOOL computes the soft cluster assignment matrix $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l}$ as

$$\mathbf{S}^{(l)} = \text{softmax}\left(\text{GNN}_1^{(l)}(\boldsymbol{A}^{(l-1)}, \boldsymbol{X}^{(l-1)})\right) \quad \text{with} \quad (\boldsymbol{A}^{(0)}, \boldsymbol{X}^{(0)}) = (\boldsymbol{A}, \boldsymbol{X}), \tag{4.9}$$

and then applies $\boldsymbol{S}^{(l)}$ and a second GNN to compute the graph representation at layer l:

$$\boldsymbol{X}^{(l)} = \boldsymbol{S}^{(l)\top}\text{GNN}_2^{(l)}(\boldsymbol{A}^{(l-1)}, \boldsymbol{X}^{(l-1)}) \qquad \text{and} \qquad \boldsymbol{A}^{(l)} = \boldsymbol{S}^{(l)\top}\boldsymbol{A}^{(l-1)}\boldsymbol{S}^{(l)}. \tag{4.10}$$

To evaluate the true benefit of learning cluster assignments, we replace $\boldsymbol{S}^{(l)}$ in Equation 4.9 with a normalized random matrix softmax($\tilde{\boldsymbol{S}}^{(l)}$), which we sample before training starts. We do not propagate gradients thereafter. We experiment with following distributions for the entries of $\tilde{\boldsymbol{S}}^{(l)}$:

$$\text{(Uniform) } \tilde{S}_{ij}^{(l)} \sim \mathscr{U}(a,b) \qquad \text{(Normal) } \tilde{S}_{ij}^{(l)} \sim \mathscr{N}(\mu,\sigma^2) \qquad \text{(Bernoulli) } \tilde{S}_{ij}^{(l)} \sim \mathscr{B}(\alpha) \tag{4.11}$$

Figure 4.3 compares DIFFPOOL against the randomized variants in four well-established datasets [Dwivedi et al., 2020]. Note that a randomized approach achieves the highest average accuracy in all datasets. Nonetheless, there is no clear winner among all methods. Additionally, the variances for the random pooling schemes are not significantly greater than the ones for DIFFPOOL.

Publication V shows similar experiments for graph memory network GMN [Khasahmadi et al., 2020], GRACLUS [Dhillon et al., 2007], and MINCUTPOOL [Bianchi et al., 2020]. For all these methods, results suggest that intricate pooling strategies are often innocuous. The common explanation is that the initial convolutional layers in GNNs tend to learn low-pass filters. As a consequence, it makes little difference how these representations are pooled. Publication V also supports these findings with a series of additional experiments. For example, we repeat experiments for different hyperparamter values and also evaluate the role of unsupervised losses in GMN and DIFFPOOL.

4.3 Revisiting reasearch question III

— *Can simple GNNs, with minimalist designs, perform as well as state-of-the-art models?*

The contributions in this chapter (Publication IV and Publication V) show independent evidence that this is often the case. Publication IV develops a constrained version of classical spectral GNNs which, in many cases, performs as well as state-of-the-art architectures. Publication V shows that we can generally swap intricate pooling methods with random partitioning without decrease in performance.

5. Discussion and Conclusions

This thesis covers topics in distributed Bayesian inference, meta-analysis, and graph neural networks. Throughout the thesis, scalability is a cross-cutting motivation. In all cases, the contributions herein are either geared towards accommodating large datasets or avoiding unnecessary computational overheads.

In the context of distributed Bayesian inference, we propose using normalizing flows to improve embarrassingly parallel inference (Publication I). We build on the flexibility and tractability of these normalizing flows to beat the state-of-the-art in a range of challenging scenarios, including high-dimensional and heterogeneous subposteriors. Despite our advances, there is a major question unaddressed in the literature: are subposterior samples reliable? We know this is not the case and that MCMC chains often get stuck to modes, which can cause parallel MCMC to wipe away regions of the combined posterior. We believe this is an important issue and we plan to address it in upcoming work.

We also propose an extension of distributed SGLD for federated learning (Publication II). In this setting, data are usually distributed in a non-iid manner and privacy constraints prevent it from being disclosed to a server. Despite the popularity of federated learning in the deep-learning community, Publication II is the first work dedicated to MCMC for federated settings. With the renewed interested in Bayesian deep learning, we believe future works could explore how to leverage ideas from Publication II to sample from more complex models.

In the context of meta-analysis, we propose a framework to combine Bayesian posteriors from a number of related studies (Publication III). Since it relaxes the need to extract meaningful summary statistics from data, this work significantly broadens the applicability of meta-analysis. For instance, we are able to combine likelihood-free posteriors from arbitrarily complex simulator models. Besides addressing a fundamental problem, this work also allows the reuse of posteriors from computationally costly analyses. We also believe the framework we present to handle uncertain evidence can be used for other applications, such as metalearning or active knowledge elicitation.

In the realm of graph neural networks, our scalability bias translates to simplicity. While the common trend in deep learning is to enrich models with more complex components, we strive for minimal design choices. For example, we show that it is possible to match the state-of-the-art for spectral GNNs with small modifications to the simplest polynomial graph filter (Publication IV). We also show that intricate and usually expensive graph pooling mechanisms are often innocuous (Publication V). Both these works contribute to an ongoing trend of skepticism towards wide-spread intuitions propagated at large in the deep learning community. We hope these contributions help researchers and practitioners better choose in which directions to employ their time and resources to build more accurate GNNs.

References

S. Ahn, B. Shahbaba, and M. Welling. Distributed stochastic gradient MCMC. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, 29(3):599–615, 2019.

F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 874–883, 2020.

F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi. Graph neural networks with convolutional ARMA filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

C. Chen, N. Ding, and L. Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems 29*, 2015.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 30*, 2016.

I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

P. Diaconis and S. L. Zabell. Updating subjective probability. *Journal of the American Statistical Association*, 77(380):822–830, 1982.

L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. *ArXiv preprint*, 1410.8516, 2014.

K. A. Dubey, S. J. Reddi, S. A. Williamson, B. Póczos, A. J. Smola, and E. P. Xing. Variance reduction in stochastic gradient langevin dynamics. In *Advances in Neural Information Processing Systems 30*, 2016.

V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *ArXiv preprint*, 2003.00982, 2020.

M. Fey, J. E. Lenssen, F. Weichert, and H. Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *2018 IEEE / CVF Conference on Computer Vision and Pattern Recognition*, 2018.

H. Gao and S. Ji. Graph U-nets. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis, Third Edition (Chapman & Hall / CRC Texts in Statistical Science)*. Chapman and Hall/CRC, London, third edition, 2013.

J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

P. T. Higgins, S. G. Thompson, and D. J. Spiegelhalter. A re-evaluation of random-effects meta-analysis. *Journal of the Royal Statistical Society A*, 172(1):137–159, 2009.

W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 34*, 2020.

J. Huang, Z. Li, N. Li, S. Liu, and G. Li. Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *2019 IEEE / CVF International Conference on Computer Vision*, 2019.

E. Isufi, A. Loukas, A. Simonetto, and G. Leus. Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, 65(2):274–288, 2017.

R. Jeffrey. *Subjective Probability: The Real Thing*. Cambridge University Press, New York, 2004.

A. H. Khasahmadi, K. Hassani, P. Moradi, L. Lee, and Q. Morris. Memory-based graph networks. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

B. Knyazev, G. W. Taylor, and M. Amer. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems 33*, 2019.

J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *ArXiv preprint*, 1610.02527, 2016.

J. Lee, I. Lee, and J. Kang. Self-attention graph pooling. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

R. Levie, F. Monti, X. Bresson, and M. M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2019.

J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 66 (1):e66–e82, 2017.

Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang. Graph convolutional networks with eigenpooling. In *International Conference on Knowledge Discovery & Data Mining*, 2019.

J. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. Relational pooling for graph representations. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

T. Nagapetyan, A. B. Duncan, L. Hasenclever, S. J. Vollmer, L. Szpruch, and K. Zygalakis. The true cost of stochastic gradient Langevin dynamics. *ArXiv preprint*, 1706.02692, 2017.

R. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, New York, 2011.

W. Neiswanger, C. Wang, and E. P. Xing. Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.

C. Nemeth and C. Sherlock. Merging MCMC subposteriors through Gaussian-process approximations. *Bayesian Analysis*, 13(2):507–530, 06 2018.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

D. B. Rubin. The calculation of posterior distributions by data augmentation: Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The sir algorithm. *Journal of the American Statistical Association*, 82(398): 543–546, 1987.

A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61:1644–1656, 2013.

S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch. Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11:78–88, 2016.

P. Smets. Jeffrey's rule of conditioning generalized to belief functions. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, 1993.

Y. W. Teh, A. H. Thiery, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(1):193–225, 2016.

S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh. Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(1):5504–5548, 2016.

X. Wang, F. Guo, K. A. Heller, and D. B. Dunson. Parallelizing MCMC with random partition trees. In *Advances in Neural Information Processing Systems 29*, 2015.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems 32*, 2018.

H. Yuan and S. Ji. StructPool: Structured graph pooling via random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

J. Zhao and D. Osherson. Updating beliefs in light of uncertain evidence: Descriptive assessment of Jeffrey's rule. *Thinking & Reasoning*, 16(4):288–307, 2010.

Aalto-DD 166/2021

ISBN 978-952-64-0608-4 (printed)
ISBN 978-952-64-0609-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Computer Science
www.aalto.fi