SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Thesis title

Author

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Thesis title

Titel der Abschlussarbeit

Author:	Author
Examiner:	Supervisor
Supervisor:	Advisor
Submission Date:	Submission date

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, Submission date Author

Abstract

Contents

Notation	Explanation
θ	Parameters of a neural network
$(x, y) \in \mathcal{D}$	Dataset with inputs x and labels/targets y
v_t	Time-dependent vector field
J_v	Jacobian of the vector field v
ϕ_t	Flow map from time 0 to time t

Table 1: Summary of notation used throughout the thesis.

1 Introduction

1.1 Section

2 Bayesian Deep Learning

A primary use case for a generative model over neural network weights is in Bayesian deep learning, where it can allow efficient inference by transporting the prior distribution to the posterior. Thus, to motivate the rest of the discussion, we first give an overview of concepts from Bayesian deep learning. Section 2.1 is a general introduction. It is followed by a review of inference methods (Laplace approximations, variational inference, MCMC-based methods) typically used for Bayesian neural networks, and we conclude with a review of literature around Bayesian deep learning particularly relevant for our work in Section 2.4.

2.1 General Concepts

In typical Bayesian fashion, Bayesian deep learning (refer to (MacKay, 1992; Neal, 1996) for foundational work and (Goan and Fookes, 2020; Arbel et al., 2023) for more recent reviews) aims to quantify the uncertainty in neural networks through probability distributions over their parameters, rather than obtaining a single solution by an SGD-like optimization method. Then, given the *posterior* distribution $p(\theta|\mathcal{D})$ over weights θ conditioned on the dataset $\mathcal{D}$, predictions are obtained via *Bayesian model averaging*:

$$p(y|x,\mathcal{D}) = \mathbb{E}_{\theta\sim p(\theta|\mathcal{D})}\left[p(y|x,\theta)\right] = \int p(y|x,\theta)p(\theta|\mathcal{D})d\theta, \tag{2.1}$$

where the prediction is averaged over the posterior over the weights. Note that since the forward pass through the model $p(y|x,\theta)$, is deterministic, the uncertainty in predictions results solely from the uncertainty over parameters. To bring things together, Bayesian inference over neural network weights consists of three steps:

1. Specify prior $p(\theta)$.
2. Compute/sample posterior $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$.
3. Average predictions over the posterior.

The last step only requires forward passes through the model and thus is straightforward. The first two steps require deeper consideration.

2.2 Priors

Specifiying a prior mainly consists of two choices: specifying an architecture, and specifying a probability distribution over the weights. The distribution is typically taken to be an isotropic Gaussian, which is an uninformative prior but straightforward to work with.

Different architectural decisions also result in different functions, even if the flattened weight vectors are identical, meaning that the choice of an architecture further specifies a prior in function space. As a simple example, keeping the depth and width of a neural network constant, even just changing the activation function from a ReLU to a sigmoid results in a differnet distribution of functions. The functional distribution can also be specified in a more deliberate way; e.g. a translation-invariant convolutional neural network, or a group-equivariant network (Cohen and Welling, 2016) puts probability mass only on functions satisfying certain equivariance constraints depending on the task at hand.

We keep this discussion short since the choice of a prior has tangential impact in the rest of the presentation, and we refer to recent reviews such as (Fortuin, 2022) for a more detailed treatment of Bayesian neural network priors.

2.3 Inference

- Variational methods
- MCMC-based methods
- Transformation-based methods (normalizing flow)

Laplace
VI
DE
MCMC introduction
HMC
Highlight distinction between chain-based and transformation based sampling, using normalizing flows and connections to optimal transport etc
– Stochastic –
SG-MCMC
SG-HMC

2.4 Properties of Bayesian Neural Network Posteriors

How they are like Boltzmann distributions, not arbitrary.

3 Flow Models

3.1 Computing Likelihoods

In earlier normalizing flows that aim to learn a static mapping between the two distributions (Rezende and Mohamed, 2015), given source samples $x_0 sim p_0$, likelihoods of the generated samples $z = f(x_0) \approx p_1$ can be computed exactly via the change of variables formula

$$\log p_1(z) = \log p_0(z) - \log \det \left| J_f(z) \right| \tag{3.1}$$

where J_f is the Jacobian of f. Thus, we can obtain exact likelihoods for the generated samples by taking the determinant of the Jacobian of the normalizing flow. Since Jacobian computations can be costly, this has motivated work on designing normalizing flows with easier to compute Jacobians, such as RealNVP (Dinh et al., 2017).

In a *continuous normalizing flow* on the other hand, the *instantenous change of variables* formula (Chen et al., 2018) defines the change in probability mass through time. Given that the vector field v_t is continuous in t and uniformly Lipschitz continuous in $\mathbb{R}^d$, it holds that

$$\frac{d \log p_t(\phi_t(x))}{dt} = -\operatorname{div}(v_t(\phi_t(x))) \tag{3.2}$$

$$= -\operatorname{Tr}\left(\frac{dv_t(\phi_t(x))}{dt}\right) \tag{3.3}$$

where $\frac{dv_t(\phi_t(x))}{dt} =: J_v(\phi_t(x))$ is the Jacobian of the vector field. Then we integrate over time to compute the full change in probability:

$$\log p_1(\phi_1(x)) = \log p_0(\phi_0(x)) - \int_0^1 \operatorname{Tr}(J_v(\phi_t(x)))dt. \tag{3.4}$$

Then we can integrate the Jacobian trace of the vector field through time (simultaneously with sampling) to obtain exact likelihoods for the generated samples.

3.1.1 Faster Likelihoods Through Trace Estimation

However, materializing the full Jacobian of the vector field can be prohibitively expensive, especially if the task is high dimensional (as in our case) since the log determinant

computation has a time complexity of $O(d^3)$ (Grathwohl et al., 2018) without any restrictions on the structure of the Jacobian.

To alleviate this problem, (Grathwohl et al., 2018) propose to use the *Hutchinson trace estimator* (Hutchinson, 1990) for an unbiased estimate of the Jacobian trace of a square matrix:

$$\mathrm{Tr}(J_v) = \mathbb{E}_{p(\epsilon)}\left[\epsilon^T J_v \epsilon\right] \tag{3.5}$$

where $p(\epsilon)$ is chosen such that $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Cov}(\epsilon) = I$, typically a Gaussian or a Rademacher distribution. Then, we can use this estimator in place of the explicit trace computation in Equation 3.4 and compute the likelihoods as

$$\log p_1(\phi_1(x)) = \log p_0(\phi_0(x)) - \int_0^1 \mathbb{E}_{p(\epsilon)}\left[\epsilon^T J_v(\phi_t(x))\epsilon\right] dt. \tag{3.6}$$

The performance benefit of using the Hutchinson trace estimator results from the fact that the Jacobian-vector product $J_v\epsilon$ can be computed very efficiently by automatic differentiation (Baydin et al., 2018), giving the whole approach a time complexity of $O(d)$ only. Due to this significant performance improvement and being an unbiased estimate, the Hutchinson trace estimator has been widely used in the diffusion/flow model literature (Lipman et al., 2023; Song et al., 2021).

4 Geometry of Neural Networks

4.1 Symmetries of Neural Networks

Also introduce mode connectivity here, before canonical reps

Introduce permutation and scaling symmetries, and maybe other symmetries if we end up using them.

4.2 Canonical Representations of Neural Networks

Mainly based on the deep toroids paper

5 Graph Neural Networks & MetaNets

6 Generative Models in Weight-Space

7 Method & Design Choices

8 Results

9 Discussion

10 Conclusion & Future Work

Abbreviations

List of Figures

List of Tables

Bibliography

Arbel, Julyan et al. (2023). *A Primer on Bayesian Neural Networks: Review and Debates*. DOI: 10.48550/arXiv.2309.16314. arXiv: 2309.16314 [cs, math, stat].

Baydin, Atılım Gunes et al. (2018). "Automatic Differentiation in Machine Learning: A Survey." In: *Journal of Machine Learning Research*.

Chen, Ricky T. Q. et al. (2018). "Neural Ordinary Differential Equations." In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Cohen, Taco and Max Welling (2016). "Group Equivariant Convolutional Networks." In: *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, pp. 2990–2999.

Dinh, Laurent et al. (2017). "Density Estimation Using Real NVP." In: *International Conference on Learning Representations*.

Fortuin, Vincent (2022). "Priors in Bayesian Deep Learning: A Review." In: *International Statistical Review* 90.3, pp. 563–591. ISSN: 0306-7734, 1751-5823. DOI: 10.1111/insr.12502.

Goan, Ethan and Clinton Fookes (2020). "Bayesian Neural Networks: An Introduction and Survey." In: vol. 2259, pp. 45–87. DOI: 10.1007/978-3-030-42553-1_3. arXiv: 2006.12024 [cs, stat].

Grathwohl, Will et al. (2018). *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*. DOI: 10.48550/arXiv.1810.01367. arXiv: 1810.01367.

Hutchinson, M.F. (1990). "A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines." In: *Communications in Statistics - Simulation and Computation* 19.2, pp. 433–450. ISSN: 0361-0918. DOI: 10.1080/03610919008812866.

Lipman, Yaron et al. (2023). *Flow Matching for Generative Modeling*. DOI: 10.48550/arXiv.2210.02747. arXiv: 2210.02747 [cs, stat].

MacKay, David J C (1992). "Bayesian Methods for Adaptive Models." In.

Neal, Radford M. (1996). *Bayesian Learning for Neural Networks*.

Rezende, Danilo and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows." In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, pp. 1530–1538.

Song, Yang et al. (2021). *Score-Based Generative Modeling through Stochastic Differential Equations*. arXiv: 2011.13456 [cs, stat].