M.Sc. Thesis
Master of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Effortless Bayesian Deep Learning

Tapping into the Potential of Modern Optimizers

Silas Brack (s174433)

Kongens Lyngby 2022

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

The first move is optional and contains general background information about your key research variable or variables. The second move is the statement of the problem indicating your research hypothesis/ question. The third move is the methodology move representing your participants, if any ; your materials/instruments, procedures, and data analysis. The fourth move portrays your findings. Finally, in the last optional move, you talk about the likely implications of the study. (Maybe 2x this in terms of size.)

Preface

This Master's thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Master's degree in Mathematical Modelling and Computation.

Kongens Lyngby, October 17, 2022

Silas Brack (s174433)

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

Summary i

Preface iii

Acknowledgements v

Contents vii

List of Figures ix

List of Tables ix

1 Introduction 1

1.1 The Problem . . . 1
1.2 The State of Research in the Area . . . 1
1.3 My Solution . . . 1
1.4 Research Objectives . . . 1

2 Theory 3

2.1 Optimization in Deep Learning . . . 3
2.2 The Hessian . . . 4
2.3 The Generalized Gauss-Newton Approximation . . . 4
2.4 Fisher Information . . . 5
2.5 The Natural Gradient . . . 6
2.6 Conjugate Gradient Descent . . . 7

3 Literature 9

3.1 Adam . . . 9
3.2 K-FAC . . . 9
3.3 Noisy K-FAC . . . 9
3.4 Gauss-Newton Optimization . . . 9
3.5 Variational Adaptive-Newton . . . 9
3.6 Natural Gradient Variational Inference . . . 9
3.7 Laplace Approximation . . . 10
3.8 Linear Response Variational Bayes . . . 10

3.9 Online Laplace 10

4 Method 11

5 Results 13

6 Experiments 15

7 Conclusion 17

7.1 Conclusion and Outlook 17

A An Appendix 19

Bibliography 21

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 The Problem

Bayesian methods allow for estimates of uncertainty which enable more efficient usage of data (i.e., via active learning) and avoid overfitting. Furthermore, these uncertainty estimates improve model interpretability and assessment of model predictive confidence.

Many different methods have been proposed for Bayesian inference, but

1.2 The State of Research in the Area

Currently, approximate Bayesian methods are either expensive to compute (Markov Chain Monte Carlo), are significantly more difficult to implement (such as variational inference), or simply perform poorly and are limited in their Bayesian interpretation (MC dropout). As such, there is demand for a method which exhibits the same computational cost as the optimization algorithms for deterministic neural networks while providing accurate posterior approximations and working out-of-the-box for any given architecture.

1.3 My Solution

The ideas presented in this article essentially constitute a compilation of advice given by academics from multiple fields and institutions,

1.4 Research Objectives

This should be discussed in a bit more detail in a thesis, since there are certain objectives discussed in the project plan. In an article, a quick sentence which summarises the rest of the introduction and states succinctly exactly which problem will be tackled and which approach will be made should suffice.

In this project, we strive to develop an effortless Bayesian method. More specifically, this project investigates the connections between the Laplace approximation and the approximate second-order derivatives used in modern optimizers, such as

Adam. We use a sampling-based training procedure, where a sample is first drawn from a Gaussian weight-posterior, a gradient step is performed on this sampled neural network, and the variance of the weight-posterior is updated with an approximate Hessian. Approximating the Hessian is the most time consuming and painful-to-engineer step of this training procedure. In this project, we tap into the potential of modern machine learning frameworks to efficiently approximate the Hessian.

CHAPTER 2

Theory

2.1 Optimization in Deep Learning

Suppose a neural network is a real-valued function $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^o$ parameterized in $\boldsymbol{\theta}$ which maps an input $\boldsymbol{x}$ to an output $f(\boldsymbol{x};\boldsymbol{\theta}) \equiv f_{\boldsymbol{\theta}}(\boldsymbol{x})$. Our goal is to find the optimal parameters $\boldsymbol{\theta}^*$ which best model the observed data.

From a frequentist perspective, we can define a loss function $\mathcal{L}(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ such that the optimal parameters $\boldsymbol{\theta}^*$ minimize this loss function. Often the loss function is defined as the negative log-likelihood of the data under the model, $\mathcal{L}(\boldsymbol{\theta}) = -\log p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right)$. In this case, the goal of optimization is to find the parameters $\boldsymbol{\theta}^*$ which maximize the likelihood of the data [1].

From a Bayesian perspective, we define a prior distribution $p(\boldsymbol{\theta})$ and a likelihood function $p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right)$ such that the goal of optimization is to find the parameters $\boldsymbol{\theta}^*$ which maximize the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y})$. This way, we can quantify the uncertainty in our model parameters $\boldsymbol{\theta}$ by computing the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

2.1.1 Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a first-order optimization algorithm which iteratively updates the parameters $\boldsymbol{\theta}$ in the direction of the negative gradient of the loss function $\mathcal{L}(\boldsymbol{\theta})$. The update rule is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)$$

where η_t is the learning rate at iteration t. The learning rate η_t can be constant or adaptive, and is often chosen to be a decreasing function of t. The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)$ is computed using a single training example $\boldsymbol{x}_t, y_t$. The gradient is computed using the chain rule as

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} \log p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right) (\boldsymbol{x}_t, y_t \mid \boldsymbol{\theta}_t) \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t)$$

where $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_t)$ is the gradient of the neural network with respect to its parameters $\boldsymbol{\theta}_t$.

[1] Maximizing the likelihood is equivalent to minimizing the negative log likelihood, since the log function is strictly monotonically increasing.

Note that η_t can be interpreted as the step size of the gradient descent algorithm. It can either be constant throughout training ($\eta_t = \eta$) or adaptive (if it varies throughout training).

2.2 The Hessian

The Hessian is a matrix of second-order partial derivatives of a scalar function. From now on, we will use the notation $\nabla^2_{\boldsymbol{\theta}} f$ to represent the Hessian matrix of f, where each element is given by $\left(\nabla^2_{\boldsymbol{\theta}} f\right)_{i,j} = \frac{\partial^2 f}{\partial\theta_i \partial\theta_j}$. Suppose we have a function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$. Then the Hessian can be written as $J_{\boldsymbol{\theta}}(\nabla_{\boldsymbol{\theta}} \mathcal{L})$.

We are interested in the Hessian of the loss function $\mathcal{L}(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$. For a neural network with d parameters, the Hessian is therefore a $d \times d$ square matrix. Furthermore, if all of the neural network's second partial derivatives are continuous, then the Hessian is symmetric and positive definite. In this case, the Hessian is generally dominated by the block-diagonal [MG15].

2.3 The Generalized Gauss-Newton Approximation

From the equation of the Hessian we can apply the chain rule twice and the product rule once to obtain a simpler expression

$$\begin{aligned}\nabla^2_{\boldsymbol{\theta}} \mathcal{L} &= J_{\boldsymbol{\theta}}(\nabla_{\boldsymbol{\theta}} \mathcal{L}) = J_{\boldsymbol{\theta}}\left(\nabla_f \mathcal{L} \cdot \nabla_{\boldsymbol{\theta}} f\right) \\ &= J_{\boldsymbol{\theta}}(\nabla_{\boldsymbol{\theta}} f) \cdot \nabla_f \mathcal{L} + J_{\boldsymbol{\theta}}\left(\nabla_f \mathcal{L}\right) \cdot \nabla_{\boldsymbol{\theta}} f \\ &= \nabla^2_{\boldsymbol{\theta}} f \cdot \nabla_f \mathcal{L} + \nabla_{\boldsymbol{\theta}} f^{\mathsf{T}} \cdot \nabla^2_f \mathcal{L} \cdot \nabla_{\boldsymbol{\theta}} f \qquad (2.1)\end{aligned}$$

eq:ggn-whole-hess

with ∇_f representing the gradient with respect to $f(\boldsymbol{x})$.

Assume now we will approximate f using a first-order Taylor expansion. This approximation is given by

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) \cdot (\boldsymbol{\theta} - \boldsymbol{\theta}_0). \qquad (2.2)$$

eq:nn-taylor

Notice that $\nabla^2_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) = 0$

In this way, we are linearizing our neural network function. By combining (2.1) and (2.2) we then obtain the Gauss-Newton Hessian approximation,

$$\nabla^2_{\boldsymbol{\theta}} \mathcal{L} \approx (\nabla_{\boldsymbol{\theta}} f)^{\mathsf{T}} (\nabla^2_f \mathcal{L})(\nabla_{\boldsymbol{\theta}} f)$$

For mean square error (MSE) loss, we have one output variable ($o = 1$), so $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$. From the expression of MSE loss, we get the GGN-approximate

Hessian

$$
\begin{aligned}
\nabla_f^2 \mathcal{L} =& \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\boldsymbol{x}_n))^2 = 1 \\
\implies \nabla_{\boldsymbol{\theta}}^2 \mathcal{L} =& (\nabla_{\boldsymbol{\theta}} f)^{\mathsf{T}} (\nabla_{\boldsymbol{\theta}} f)
\end{aligned}
$$

2.4 Fisher Information

sher-information⟩ Consider the gradients of the log-likelihood optimization objective, given by the expression $\nabla_{\boldsymbol{\theta}} \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})$. We can show that the expectation of these gradients is zero:

$$
\begin{aligned}
\mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\right] =& \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{\theta})} \left[\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\right] \\
=& \int \nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\, p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\; d\boldsymbol{y} \\
=& \int \frac{\nabla p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})}{p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})} p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\; d\boldsymbol{y} \\
=& \int \nabla p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\; d\boldsymbol{y} \\
=& \nabla \int p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\; d\boldsymbol{y} \\
=& \nabla 1 \\
=& 0
\end{aligned}
$$

Let us now analyze the covariance of these gradients. It is given by

$$
\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{\theta})} \left[\left(\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - \mathbb{E}\left[\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\right]\right) \left(\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - \mathbb{E}\left[\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\right]\right)^{\mathsf{T}} \right] \\
&\qquad = \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{\theta})} \left[\left(\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - 0\right) \left(\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - 0\right)^{\mathsf{T}} \right] \\
&\qquad = \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{\theta})} \left[\nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})\, \nabla \log p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})^{\mathsf{T}} \right] \\
&\qquad \equiv \mathcal{F}(\boldsymbol{\theta})
\end{aligned}
\tag{2.3}
$$

eq:gradient-c

where we have defined the Fisher information matrix $\mathcal{F}(\boldsymbol{\theta})$ as the expectation of the outer product of the score function under our model. This product is usually intractable. It can, however, be approximated empirically by using samples from an empirical distribution $q(\boldsymbol{y})$ which is close to the true likelihood $p\,(\boldsymbol{y} \,|\, \boldsymbol{\theta})$ (2.4).

$$
\mathcal{F} \approx \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(y_n \,|\, \boldsymbol{\theta}) \nabla \log p(y_n \,|\, \boldsymbol{\theta})^{\mathsf{T}}. \tag{2.4}
$$

eq:empirical-

Note, however, that this matrix is distinct from the Fisher information matrix [Mar20; KHB19], since

Another nontrivial property of the Fisher information matrix is that it is equivalent to the negative expectation of the Hessian of the log-likelihood.

$$
\begin{aligned}
H_{\boldsymbol{\theta}}\left(\log p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right)\right) =& \nabla^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right) \\
=& \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \log p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right) \\
=& \nabla_{\boldsymbol{\theta}} \frac{\nabla_{\boldsymbol{\theta}} p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right)}{p\left(\boldsymbol{y} \mid \boldsymbol{\theta}\right)} \\
=&
\end{aligned}
$$

The Fisher information matrix can therefore be interpreted as the curvature of the log-likelihood.

RELATIONSHIP WITH GAUSS-NEWTON APPROXIMATION for common loss functions, the GGN is equivalent to the Fisher information matrix [Mar20]

RELATIONSHIP WITH KL DIVERGENCE

2.5 The Natural Gradient

[Ama98] introduced the natural gradient as a way to optimize a function f parameterized by $\boldsymbol{\theta}$ by following the direction of steepest descent in the Fisher information metric. It can be interpreted as The algorithm can be seen in Algorithm 1. In prac-

Algorithm 1 Natural Gradient Descent

ural-gradient⟩

```
1: for $i = 1$ to $T$ do
2:     Calculate $\mathcal{L}(\boldsymbol{\theta})$
3:     Calculate the gradient of the loss $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
4:     Calculate the Fisher information matrix $\mathcal{F}(\boldsymbol{\theta})$
5:     Calculate the natural gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\mathcal{F}(\boldsymbol{\theta})^{-1}$
6:     Update the parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\mathcal{F}(\boldsymbol{\theta})^{-1}$
7: end for
8: return $\boldsymbol{\theta}$
```

tice, deep learning models have millions (or billions) or parameters, and the Fisher information matrix is consequently often too large to be computed and stored. This therefore limits the applicability of the natural gradient. Instead, it is often approximated by the diagonal of the Hessian.

[KB14] [MG15] [BRB17]

For a recent, detailed discussion of the natural gradient, see [Mar20]. [Wu+19]

2.6 Conjugate Gradient Descent

If allocation of memory to store a matrix is infeasible, but the computational graph for this matrix can be determined, then the conjugate gradient method can be used to approximate the inverse of this matrix [GBJ18; NW99].

Instead of computing

$$\begin{aligned} \boldsymbol{\theta}^* &= H^{-1}\boldsymbol{\epsilon} \\ \Rightarrow H\boldsymbol{\theta}^* &= HH^{-1}\boldsymbol{\epsilon} \\ \Rightarrow H\boldsymbol{\theta}^* &= \boldsymbol{\epsilon}. \end{aligned}$$

EXPLANATION *When even calculating or instantiating H is prohibitively time-consuming, one can use conjugate gradient algorithms to approximately compute H−1 g (Wright and Nocedal, 1999, Chapter 5). The advantage of conjugate gradient algorithms is that they approximate H−1 g using only the Hessian-vector product H g , which can be computed efficiently using automatic differentiation without ever forming the full Hessian H . See, for example, the hessian vector product method of the Python autograd package (Maclaurin et al., 2015). Note that a separate conjugate gradient problem must be solved for each column of g / , so if the parameter of interest g ( ) is high-dimensional it may be faster to pay the price for computing and inverting the entire matrix H . See 5.3.2 for more discussion of a specific example. In Theorem 2, we require 0∗ to be at a true local optimum. Otherwise the estimated sensitivities may not be reliable (e.g., the covariance implied by Eq. (14) may not be positive definite). We find that the classical MFVB coordinate ascent algorithms (Blei et al. (2016, Section 2.4)) and even quasi-second order methods, such as BFGS (e.g., Regier et al., 2015), may not actually find a local optimum unless run for a long time with very stringent convergence criteria. Consequently, we recommend fitting models using second-order Newton trust region methods. When the Hessian is slow to compute directly, as in Section 5, one can use the conjugate gradient trust region method of Wright and Nocedal (1999, Chapter 7), which takes advantage of fast automatic differentiation Hessian-vector products without forming or inverting the full Hessian.*

CHAPTER 3
Literature

3.1 Adam

[Sut+13] [KB14]

Note that Adam approximates the second moment of the gradient of the loss, that this corresponds to the variance of this gradient, and that this is the diagonal of the Fisher information matrix, as we defined in Section 2.4 ((2.3))!

3.2 K-FAC

Another alternative to approximating the Fisher information matrix is by calculating the Kronecker-factored Approximate Curvature (K-FAC). [MG15]

3.3 Noisy K-FAC

[Zha+18]

3.4 Gauss-Newton Optimization

[BRB17]

3.5 Variational Adaptive-Newton

[Kha+17]

3.6 Natural Gradient Variational Inference

Natural gradient VI (VOGN) [Osa+19] approximates the posterior as

$$p(\boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{0}, \boldsymbol{I}/\delta)$$

with an update of the natural parameter

$$\boldsymbol{\lambda}_{t+1} = (1+\tau\rho)\boldsymbol{\lambda}_t - \rho\nabla_\mu \mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta} + 1/2\tau\delta\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta})\right]$$

3.7 Laplace Approximation

In the Laplace approximation (LA) [Dax+21], the posterior is approximated by a Gaussian, similarly to mean-field VI. However, instead of finding the optimal Gaussian distribution locations and scales by maximising the ELBO, the LA finds the location by computing the MAP solution $\boldsymbol{\theta}_\text{MAP}$ and the scale by approximating the log posterior with a second degree Taylor expansion around this solution ($\boldsymbol{\theta}_0 = \boldsymbol{\theta}_\text{MAP}$) and determining the curvature via its Hessian matrix $\boldsymbol{H} = \nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \boldsymbol{y})\big|_{\boldsymbol{\theta}_\text{MAP}}$. Since the Taylor expansion is performed around the MAP solution, the first order derivative is zero, and the expansion is simply given by

$$\begin{aligned}
\ln p(\boldsymbol{\theta} \mid \boldsymbol{y}) &\approx \ln p(\boldsymbol{\theta}_\text{MAP} \mid \boldsymbol{y}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP})^\mathsf{T}\left(\nabla^2_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta} \mid \boldsymbol{y})\big|_{\boldsymbol{\theta}_\text{MAP}}\right)(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP}) \\
\Rightarrow \tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{y}) &\approx p(\boldsymbol{\theta}_\text{MAP} \mid \boldsymbol{y}) \exp\left(\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP})^\mathsf{T}\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP})\right) \propto p(\boldsymbol{\theta} \mid \boldsymbol{y})
\end{aligned}$$

where $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{y})$ corresponds to the unnormalized posterior. Normalizing it yields

$$\begin{aligned}
p(\boldsymbol{\theta} \mid \boldsymbol{y}) &\approx \sqrt{\frac{\det(-\boldsymbol{H})}{(2\pi)^D}} \exp\left(\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP})^\mathsf{T}\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_\text{MAP})\right) \\
&= \mathcal{N}(\boldsymbol{\theta}_\text{MAP}, -\boldsymbol{H}^{-1}).
\end{aligned}$$

We have now shown that approximating the log-posterior with a second degree Taylor expansion around the $\boldsymbol{\theta}_\text{MAP}$ corresponds to approximating the posterior with a Gaussian distribution given by $\mathcal{N}(\boldsymbol{\theta}_\text{MAP}, -\boldsymbol{H}^{-1})$ where $\boldsymbol{H} = \nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \boldsymbol{y})\big|_{\boldsymbol{\theta}_\text{MAP}}$.

In practice, we can determine the Hessian of the loss with respect to the parameters, where the loss is given by the negative log-posterior. In this case, we get $\boldsymbol{H} = -\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \boldsymbol{y})\big|_{\boldsymbol{\theta}_\text{MAP}} = \nabla^2_{\boldsymbol{\theta}}\mathcal{L}\big|_{\boldsymbol{\theta}_\text{MAP}}$ and the posterior is approximated by $\mathcal{N}(\boldsymbol{\theta}_\text{MAP}, \boldsymbol{H}^{-1})$.

3.8 Linear Response Variational Bayes

[GBJ18]

INLA: Laplace for nested models (hierarchical models) [Rue+17]

3.9 Online Laplace

[Mia+22]

CHAPTER 4

Method

CHAPTER 5
Results

CHAPTER 6

Experiments

CHAPTER 7

Conclusion

7.1 Conclusion and Outlook

7.1.1 Summary and key results

Remember to actually include some results and a quantitative look at the results that were obtained.

7.1.2 Outlook and future developments

Which avenues are most promising for future research? Either in order to solve the problem this paper tables or to solve the next problem.

APPENDIX A

An Appendix

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[Ama98] Shun-Ichi Amari. “Natural gradient works efficiently in learning.” In: *Neural computation* 10.2 (1998), pages 251–276.

[BRB17] Aleksandar Botev, Hippolyt Ritter, and David Barber. “Practical gauss-newton optimisation for deep learning.” In: *International Conference on Machine Learning*. PMLR. 2017, pages 557–565.

[Dax+21] Erik Daxberger et al. “Laplace Redux-Effortless Bayesian Deep Learning.” In: *Advances in Neural Information Processing Systems* (2021).

[GBJ18] Ryan Giordano, Tamara Broderick, and Michael I Jordan. “Covariances, robustness and variational bayes.” In: *Journal of machine learning research* 19.51 (2018).

[KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).

[Kha+17] Mohammad Emtiyaz Khan et al. “Variational adaptive-Newton method for explorative learning.” In: *arXiv preprint arXiv:1711.05560* (2017).

[KHB19] Frederik Kunstner, Philipp Hennig, and Lukas Balles. “Limitations of the empirical Fisher approximation for natural gradient descent.” In: *Advances in neural information processing systems* 32 (2019).

[Mar20] James Martens. “New insights and perspectives on the natural gradient method.” In: *The Journal of Machine Learning Research* 21.1 (2020), pages 5776–5851.

[MG15] James Martens and Roger Grosse. “Optimizing neural networks with kronecker-factored approximate curvature.” In: *International conference on machine learning*. PMLR. 2015, pages 2408–2417.

[Mia+22] Marco Miani et al. “Laplacian Autoencoders for Learning Stochastic Representations.” In: *arXiv preprint arXiv:2206.15078* (2022).

[NW99] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[Osa+19] Kazuki Osawa et al. “Practical deep learning with Bayesian principles.” In: *Advances in neural information processing systems* 32 (2019).

[Rue+17] Håvard Rue et al. “Bayesian computing with INLA: a review.” In: *Annual Review of Statistics and Its Application* 4 (2017), pages 395–421.

[Sut+13] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning." In: *International conference on machine learning*. PMLR. 2013, pages 1139–1147.

[Wu+19] Yan Wu et al. "Logan: Latent optimisation for generative adversarial networks." In: *arXiv preprint arXiv:1912.00953* (2019).

[Zha+18] Guodong Zhang et al. "Noisy natural gradient as variational inference." In: *International Conference on Machine Learning*. PMLR. 2018, pages 5852–5861.