



CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering



Estimating Sparse Parameterization of Neural Networks

Metody odhadu řídké parametrizace neuronových sítí

Master's Thesis

Author: **Bc. Lukáš Kulička**
Supervisor: **doc. Ing. Václav Šmídl, Ph.D.**
Consultant: **Ing. Milan Papež, Ph.D.**
Academic year: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Lukáš Kulička
Studijní program: Aplikované matematicko-stochastické metody
Název práce (česky): Metody odhadu řídké parametrizace neuronových sítí
Název práce (anglicky): Estimating Sparse Parameterization of Neural Networks

Pokyny pro vypracování:

- 1) Představte problém hledání řídké parametrizace modelu jako nástroj pro tvorbu interpretovatelných pravděpodobnostních modelů. Vypracujte přehled metod používaných pro tento problém, zvláštní pozornost věnujte metodám využívajícím apriorní rozdělení parametrů (shrinkage prior).
- 2) Vypracujte přehled modelů shrinkage prior a metod používaných pro jejich odhad. Zvláštní pozornost věnujte variačním metodám, například variační metodě ADAM. Navrhněte model lineární regrese, na němž bude možné studovat analytické řešení řídké parametrizace. Na tomto modelu srovnajte chování různých metod a diskutujte o jejich vlastnostech.
- 3) Aplikujte testované metody na vybrané architektury neuronových sítí a demonstруйте jejich vlastnosti v porovnání s používanými metodami, jako je variační dropout.
- 4) Aplikujte testované metody na architektury zpracovávající množinová data. Ověřte shodu metod odhadu řídké parametrizace s metodami výběru příznaků.
- 5) Všechny experimenty provádějte na volně dostupných datech z databáze UCI, případně reálných datech z praxe.

Doporučená literatura:

- 1) E. Nalisnick, J. M. Hernández-Lobato, P. Smyth, Dropout as a Structured Shrinkage Prior. In 'International Conference on Machine Learning', PMLR, 2019, 4712-4722.
- 2) C. Louizos, K. Ullrich, M. Welling, Bayesian Compression for Deep Learning. In 'Advances in Neural Information Processing Systems', 2017, 3288-3298.
- 3) T. Pevny, P. Somol, Discriminative Models for Multi-instance Problems with Tree Structure. In 'Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security', 2016, 83-91.
- 4) K. Mohammad, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, A. Srivastava, Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In 'International Conference on Machine Learning', PMLR, 2018, 2611-2620.

Jméno a pracoviště vedoucího diplomové práce:

doc. Ing. Václav Šmídl, Ph.D.

Ústav teorie informace a automatizace, Pod Vodárenskou věží 4, 182 00, Praha 8

Jméno a pracoviště konzultanta:

Ing. Milan Papež, Ph.D.

Katedra počítačů FEL ČVUT v Praze, Karlovo náměstí 13, 121 35 Praha 2

Datum zadání diplomové práce: 31.10.2021

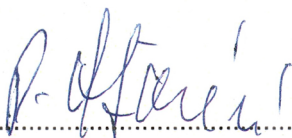
Datum odevzdání diplomové práce: 2.5.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 01.11.2021



garant oboru



vedoucí katedry




děkan

Acknowledgment:

I would like to thank doc. Ing. Václav Šmídl, Ph.D. for his expert guidance, endless support and patience he has shared with me over the last three years. It was a privilege to study and work under his supervision. I would also like to thank Ing. Milan Papež, Ph.D. for his consultancy knowledge and advice especially in the programming part.

Last but not least, let me thank my parents, Mgr. Jiří Kulička, Ph.D. and Simona Kuličková, my girlfriend Lucie Kadrmasová and my friends for their never ending support during my studies at the Faculty of Nuclear Sciences and Physical Engineering, CTU in Prague.

Author's declaration:

I declare that this Master's Thesis is entirely my own work and I have listed all the used sources in the bibliography.

Prague, May 2, 2022

Bc. Lukáš Kulička



Název práce:

Metody odhadu řídké parametrizace neuronových sítí

Autor: Bc. Lukáš Kulička

Program: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: doc. Ing. Václav Šmídl, Ph.D.

ÚTIA Akademie věd ČR, Pod vodárenskou věží 4, 180 00 Praha 8

Konzultant: Ing. Milan Papež, Ph.D.

Katedra počítačů FEL ČVUT v Praze, Karlovo náměstí 13, 121 35 Praha 2

Abstrakt: Diplomová práce se zabývá metodami odhadu řídké parametrizace neuronových sítí, jimiž je možné prořezávat přeparametrizované neuronové sítě a snížit tak jejich komplexitu ve snaze odhalit pouze relevantní parametry, čímž lze zvýšit celkovou interpretabilitu modelu. V rámci práce jsou popsány klasické a variační způsoby, jakými lze tyto parametrizace odhadovat. K tomu převážně poslouží přehled speciálních apriorních distribucí, zde označovaných jako utahující se apriorna, díky kterým dokážeme do modelu vnést informaci o preferenci řídké parametrizace. Variačními metodami poté lze aproximovat aposteriorní distribuci parametrů modelu. Pomocí této aposteriorní distribuce je možné lépe kvantifikovat neurčitost těchto parametrů. Na závěr práce jsou tyto metody aplikovány na různé modely včetně lineární a logistické regrese, neuronových sítí a multi-istančního učení. Experimenty jsou prováděny jak na syntetických, tak reálných datech.

Klíčová slova: Bayesovské metody, multi-istanční učení, neuronové sítě, řídkost, utahující se apriorna

Title:

Estimating Sparse Parameterization of Neural Networks

Author: Bc. Lukáš Kulička

Abstract: The Master's thesis deals with methods for estimating sparse parameterization of neural networks, which can be used to prune overparameterized neural networks and reduce their complexity in an attempt to reveal only relevant parameters, thus increasing the overall interpretability of the model. In this thesis, the classical and the variational methods, which allow these parameterizations to be estimated, are described. This is achieved by reviewing special prior distributions, here referred to as shrinkage priors, which allow us to incorporate our preferences about sparse parameterizations into the model. Variational methods then help us to approximate the posterior distribution for model parameters. Using this posterior distribution, it is possible to better quantify the uncertainty of the parameters. Finally, the methods are applied to various models, including linear and logistic regression, neural networks, and are also utilized in the concept of multi-instance learning. The experiments are carried out on both synthetic and real data.

Key words: Bayesian methods, multi-instance learning, neural networks, shrinkage priors, sparsity

Contents

Introduction	8
1 Theoretical Background	10
1.1 Exponential Family	10
1.2 Conjugate Exponential Family Priors	13
1.3 Non-informative Priors	14
1.4 Shrinkage Priors	14
1.4.1 Automatic Relevance Determination	15
1.5 Variational Bayesian Framework	17
1.5.1 Setup and Goal of Variational Bayes	17
1.5.2 KL Divergence and Evidence Lower Bound	17
1.6 Bayesian Neural Networks	18
1.6.1 Activation Functions	19
1.6.2 Point Estimates	20
1.6.3 Posterior Distribution	20
1.7 Bayesian Ridge Regression	21
1.7.1 Hyperparameter ψ Prior	21
1.7.2 ARD in Bayesian Ridge Regression	23
1.8 Bayesian Logistic Regression	23
1.8.1 Likelihood Function and Formation of Posterior	24
1.9 Multi-Instance Learning	24
1.9.1 Embedded-Space Paradigm	25
2 Variational Optimization	27
2.1 Natural-Gradient Variational Inference	27
2.1.1 Implementation	27
2.2 Mean-Field Variational Inference	28
2.2.1 Natural-Gradient Updates for Model Parameters	28
2.2.2 Natural-Gradient Updates for Hyperparameters	29
2.3 Variational Online-Newton	31
2.3.1 Mean-Field Variant of Variational Online Newton	32
2.3.2 Reparameterization Trick in Hessian	32
2.4 Variational Online Gauss-Newton	33
2.5 Variational RMSprop	33
2.6 Variational ADAM	34
2.7 Variational ADAM with ARD Prior	34

3 Experiments	36
3.1 Sparse Linear Regression	36
3.1.1 Model Architecture	36
3.1.2 Maximum Likelihood Estimation in Linear Regression	37
3.1.3 L_1 Penalization in Linear Regression	38
3.1.4 Variational ADAM in Linear Regression	41
3.1.5 VADAM with ARD Prior in Linear Regression	44
3.1.6 Evaluation of Methods	46
3.2 Sparse Logistic Regression	47
3.2.1 Model Architecture	47
3.2.2 Maximum Likelihood Estimation in Logistic Regression	48
3.2.3 L_1 Penalization in Logistic Regression	48
3.2.4 Variational ADAM with ARD Prior in Logistic Regression	51
3.2.5 Evaluation of Methods	52
3.3 Sparse MIL	52
3.3.1 Model Architecture	53
3.3.2 L_1 Penalization in MIL	54
3.3.3 Variational ADAM with ARD Prior in MIL	56
3.3.4 Evaluation of Methods	58
Conclusion	59
Appendix A	61
Bibliography	63

Introduction

Today, there are no longer many obstacles to train complex models, such as deep neural networks, due to the enormous computing power of graphical processing units (GPU) or tensor processing units (TPU). Whether one chooses to invest in their own computer or take advantage of cloud environments, getting access to high-quality and powerful hardware is easier than ever. However, many state-of-the-art deep learning models operate as so-called *blackbox models* – one provides an input and the model returns an output based on learned parameters, often without you knowing what is going on inside, and why that particular output was thrown out by the model. Applications of such models can nowadays be found in many fields, including natural language processing, image processing or time series analysis, where they face the challenge of processing real-world queries of evergrowing size and dimensionality of input data. Not only large data with many observations, which usually require a lot of computing power, but also data whose dimensionality is often larger than the number of observations. Moreover, the models may suffer from the overparameterization in the production-ready stage, which may prolong their response time and puts an unnecessarily high demand on the computing resources. Another challenge to these models is how to quantify the uncertainty of their outputs. Therefore, having an estimate of the entire distribution of the unknown parameters (in contrast to the point estimate), i.e., capturing their uncertainty, might be more preferable. Not only will this increase the interpretability of the models, but it will also make it possible to embed some prior knowledge about the parameters, which may crucially force a sparser parameterization of the model and thus remove redundant parameters.

Hence, this thesis should give the reader an insight into methods that can deal with the overparameterization in deep learning models using the Bayesian approach and search for the entire distribution of the parameters instead of their point estimates at the output.

The text is divided into three main chapters. The first one is devoted to the necessary theoretical background. This chapter will begin by introducing various probabilistic distributions from the exponential family called priors. Most attention will be paid to the class of shrinkage priors, which allows us to embed our preferences about sparse parameterization. This will be followed by a description of a framework called variational Bayes (VB) introducing the reader to the alternative to finding point estimates of parameters. In particular, the relation between Kullback-Leibler (KL) divergence and the evidence lower bound (ELBO) will be defined, which is the key tool in designing methods to approximate the posterior distribution of the parameters. A special case of neural networks, i.e., linear and logistic regression, and a way to approximate the corresponding posterior distributions using a shrinkage prior will be illustrated. The chapter will conclude with a brief insight into multi-instance learning, especially its structure and its difference from conventional machine learning.

The second chapter discusses variational optimization and inference and offers a wide range of methods to approximate the posterior distribution using natural gradients. It will also be shown how efficiently the expensive computation of Hessian matrix or the inverse Fisher information matrix can be avoided with the help of the VB framework. Furthermore, variational alternatives to classical optimizers, such as variational RMSprop and variational ADAM, will be introduced. At the end of the chapter, the author's

proposed algorithm that performs variational inference with the shrinkage prior and helps to find a sparse parameterization of the model will be described.

In the third chapter, the theoretical knowledge and proposed methods from the previous chapters will be applied to three experiments. All of them will be concerned with the search for a sparse parameterization of the models relying on neural networks, assuming the smallest possible increase in the model error. Or, alternatively, in the case of overparameterized neural networks, prune them so that the model error is even more reduced. Each experiment will include a comparison of various methods.

The Julia programming language (version 1.6) is used exclusively for numerical calculations in this thesis and all source code will be available at <https://github.com/Kulda16/Sparsity.jl>.

Chapter 1

Theoretical Background

Let us assume, we observe some data as a particular realization $(x_{n,1}, \dots, x_{n,K})$ of a random vector, $\mathbf{X}_{n,\cdot} = (X_{n,1}, \dots, X_{n,K})$, consisting of K variables (*predictors*) and a corresponding realization y_n of the random variable, Y (response variable), which we want to model based on the predictors. Then we repeat this process N -times and store the values in a matrix form, $\mathbf{X} \in \mathbb{R}^{N \times K}$, (data matrix) and vector form, $\mathbf{y} \in \mathbb{R}^N$, until we get a dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$. We are interested in obtaining the distribution of parameters θ , after we have observed the dataset \mathcal{D} , which we model with Bayes' theorem [4]:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}, \theta)d\theta} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}. \quad (1.1)$$

We will describe the individual objects in Eq. (1.1) and their meaning in more detail. The distribution $p(\theta)$ is called the *prior* distribution (shortly prior) and it represents our prior knowledge of the parameters in model. The distribution $p(\mathcal{D}|\theta)$, otherwise written as $p(\mathbf{y}|\mathbf{X}, \theta)$, is viewed as a probabilistic model which assigns a probability to each possible output, \mathbf{y} , given an input, \mathbf{X} , using the set of parameters, θ [6]. We call it *likelihood*. The normalization constant, $p(\mathcal{D})$, in the denominator denotes the *evidence* (or marginal likelihood), which can be computed via marginalizing the parameters out of the joint distribution of data and the parameters, and it is an immutable constant normalizing the *posterior* distribution. The posterior, $p(\theta|\mathcal{D})$, is the distribution which takes both prior knowledge and data into account [45].

Obtaining the posterior will be vital for this work as it allows us to describe the uncertainty of all parameters in the probabilistic model. As it turns out, we cannot calculate the posterior analytically in the vast majority of cases because of the integral in Eq. (1.1), so we need to choose procedures and approximations to estimate this distribution as accurately as possible. It will also be shown how this calculation relates to sparse parameterization, which will lead to less complex models and increase their interpretability.

1.1 Exponential Family

Consider K i.i.d.¹ random variables, $\mathbf{X} = (X_1, \dots, X_K)$, and parameter $\theta = (\theta_1, \dots, \theta_J)$ assumed to belong to an open set $\Theta \subset \mathbb{R}^J$. Exponential family is a set of specific distributions whose probability density function (or probability mass function, in case of a discrete random variable) is expressed in the form

$$p(\mathbf{x}|\theta) = k(\mathbf{x}) \exp \left\{ \sum_{j=1}^J t_j(\mathbf{x}) \cdot \eta_j(\theta) - \phi(\theta) \right\}, \quad (1.2)$$

¹Independent and identically distributed.

where $k(\mathbf{x})$ is a non-negative real function called *base measure*, $\boldsymbol{\eta}(\boldsymbol{\theta})$ stands for the *natural parameter*, $\phi(\boldsymbol{\theta})$ is referred to as a *log-normalizer*, and $\mathbf{t}(\mathbf{x})$ is *sufficient statistic* [10]. If $\boldsymbol{\eta}(\boldsymbol{\theta}) = \boldsymbol{\theta}$, then the exponential family is said to be in the *canonical form*. Moreover, if $\mathbf{t}(\mathbf{x}) = \mathbf{x}$ holds, then we are talking about a *natural exponential family*. Exponential family includes many of well-known and common distributions, such as normal (Gaussian), exponential, gamma, Bernoulli, Wishart, binomial, chi-squared, etc.

Let us derive three simple examples of distributions in the form of the Eq. (1.2) to obtain their natural parameters. We include them because the insights from these examples will be used extensively later.

Bernoulli distribution

Let $K = 1$, $J = 1$ and $p(x|\pi)$ be Bernoulli distribution with unknown parameter $\pi \in [0, 1]$:

$$\begin{aligned} p(x|\theta = \pi) &= \pi^x(1 - \pi)^{1-x} \\ &= \exp \{x \log(\pi) + (1 - x) \log(1 - \pi)\} \\ &= \underbrace{1}_{k(x)} \cdot \exp \left\{ \underbrace{x}_{t(x)} \cdot \underbrace{\log\left(\frac{\pi}{1 - \pi}\right)}_{\eta(\theta)} - \underbrace{(-\log(1 - \pi))}_{\phi(\theta)} \right\}. \end{aligned} \quad (1.3)$$

In Eq. (1.3), one can notice that the natural parameter for the Bernoulli distribution is the so-called *logit function*: $\eta(\theta = \pi) = \log\left(\frac{\pi}{1 - \pi}\right) = \text{logit}(\pi)$. By the inverse parameter mapping we obtain *logistic function*: $\theta(\eta) = \frac{1}{1 + \exp(-\eta)} = \text{logistic}(\eta)$. These two functions are shown for illustration in Fig. 1.1, as they will appear in later chapters.

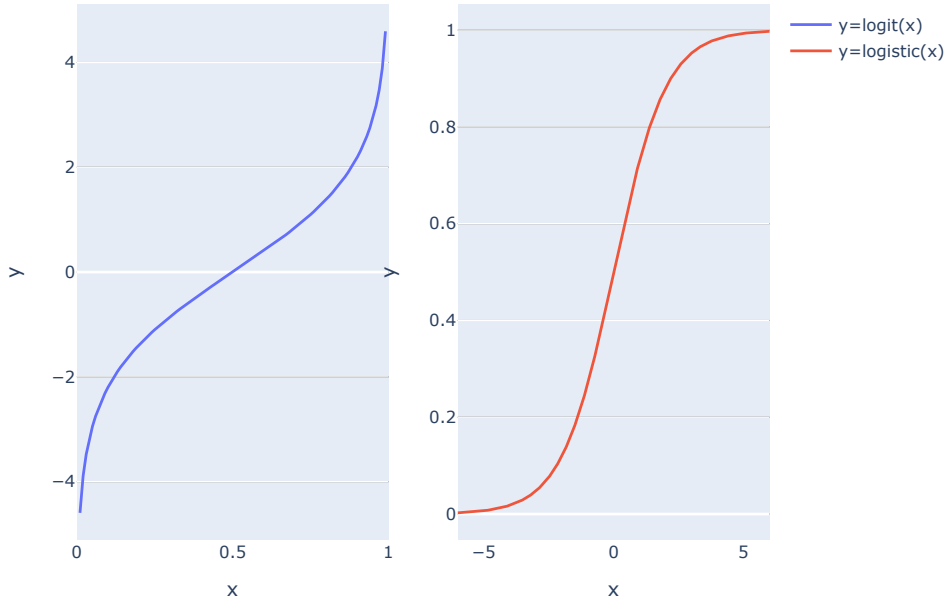


Figure 1.1: The logit and logistic functions.

Gamma distribution

Let $K = 1$, $J = 2$ and $p(x|\alpha, \beta)$ be Gamma distribution with unknown parameters α and β :

$$\begin{aligned}
 p(x|\theta = (\alpha, \beta)) &= \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp\{-\beta x\} \\
 &= 1 \cdot \exp\{(\alpha - 1) \log(x) - \beta x - (\log(\Gamma(\alpha)) - \alpha \log(\beta))\} \\
 &= \underbrace{1}_{k(x)} \exp\left\{ \underbrace{(\log(x), x)}_{\mathbf{t}(x)} \cdot \underbrace{\begin{pmatrix} \alpha - 1 \\ -\beta \end{pmatrix}}_{\boldsymbol{\eta}(\theta)} - \underbrace{(\log(\Gamma(\alpha)) - \alpha \log(\beta))}_{\phi(\theta)} \right\}.
 \end{aligned} \tag{1.4}$$

The gamma distribution also contains a *gamma function* represented by the capital letter gamma from the Greek alphabet $\Gamma(\cdot)$, which is defined as follows:

$$\Gamma(\alpha) = \int_0^{+\infty} x^{\alpha-1} \exp(-x) dx.$$

Univariate Gaussian distribution

Let $K = 1$, $J = 2$ and $p(x|\mu, \sigma^2)$ be Gaussian distribution with the unknown mean parameter μ and unknown variance parameter σ^2 :

$$\begin{aligned}
 p(x|\theta = (\mu, \sigma^2)) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x - \mu)^2}{2\sigma^2}\right\} \\
 &= \frac{1}{\sqrt{2\pi}} \exp\left\{-\log(\sigma) - \frac{x^2}{2\sigma^2} + \frac{\mu x}{\sigma^2} - \frac{\mu^2}{2\sigma^2}\right\} \\
 &= \frac{1}{\sqrt{2\pi}} \exp\left\{ \underbrace{\left(x, x^2\right)}_{\mathbf{t}(x)} \cdot \underbrace{\begin{pmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{pmatrix}}_{\boldsymbol{\eta}(\theta)} - \underbrace{\left(\log(\sigma) + \frac{\mu^2}{2\sigma^2}\right)}_{\phi(\theta)} \right\}.
 \end{aligned} \tag{1.5}$$

Multivariate Gaussian distribution

For general K and J let $p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be multivariate Gaussian distribution with the unknown mean vector parameter $\boldsymbol{\mu}$ and the unknown covariance matrix $\boldsymbol{\Sigma}$:

$$\begin{aligned}
 p(\mathbf{x}|\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})) &= \frac{1}{(2\pi)^{N/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\} \\
 &= (2\pi)^{-N/2} \exp\left\{-\frac{1}{2} (\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - 2\mathbf{x}^T \boldsymbol{\mu}) - \frac{1}{2} \log(|\boldsymbol{\Sigma}|)\right\} \\
 &= \underbrace{(2\pi)^{-N/2}}_{k(\mathbf{x})} \exp\left\{ \boldsymbol{\mu}^T \mathbf{x} - \frac{1}{2} \text{Tr}(\boldsymbol{\Sigma}^{-1} \mathbf{x} \mathbf{x}^T) - \underbrace{\left(\frac{1}{2} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \frac{1}{2} \log(|\boldsymbol{\Sigma}|)\right)}_{\phi(\theta)} \right\}.
 \end{aligned} \tag{1.6}$$

The derivation, which can be found in more detail in [11], takes advantage of a typical property of the trace operator, i.e., it is invariant under cyclic permutations. The final step is to express sufficient statistics

and the natural parameter:

$$\mathbf{t}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \mathbf{xx}^T \end{pmatrix}, \quad \boldsymbol{\eta}(\boldsymbol{\theta}) = \begin{pmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\frac{1}{2}\boldsymbol{\Sigma}^{-1} \end{pmatrix}.$$

1.2 Conjugate Exponential Family Priors

As stated above, to obtain the exact posterior within the Bayes' theorem in Eq. (1.1) is nearly an impossible task. To properly normalize this distribution, we need to compute the integral of the joint distribution, $p(\mathbf{x}, \boldsymbol{\theta})$, over the unknown parameters, which in many cases (mostly in higher dimensions) is intractable. To avoid the numerical integration or approximations [21], we study the *conjugate systems* that allow for closure of the Bayes' theorem under an analytically tractable solution. In other words, if the likelihood and prior distributions are in the same probability distribution family, e.g., in *exponential family*, then they are called *conjugate distributions* and prior a *conjugate prior*. More specifically, if the likelihood function has a functional form that holds structural similarities to the functional form of the prior distribution, then the posterior distribution which preserves the form of the prior exists. The main benefit of conjugate probability distribution systems is the existence of a closed-form solution for updating the sufficient statistics of the posterior, which also belongs to the particular family [12]. Let's give two examples.

Exponential likelihood and Gamma prior

As mentioned before, both the exponential and gamma distributions belong to the exponential family. Consider K i.i.d. observations with the exponential likelihood function, $p(\mathbf{x}|\lambda) = \prod_{k=1}^K \lambda \exp(-\lambda x_k)$, and the gamma prior, $p(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda)$, where $\lambda \in \mathbb{R}^+$ is the *rate* and α, β are the prior hyperparameters. Then by inserting these expressions into the Eq. (1.1) and using proportionality up to the normalizing factor, i.e., the \propto -sign, we obtain the unnormalized posterior in the form of the gamma distribution,

$$\begin{aligned} p(\lambda|\mathbf{x}) &= \frac{\lambda^K \exp\left(-\lambda \sum_{k=1}^K x_k\right) \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda)}{\int_0^{+\infty} \lambda^K \exp\left(-\lambda \sum_{k=1}^K x_k\right) \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda) d\lambda} \\ &\propto \lambda^{K+\alpha-1} \exp\left(-\lambda \left(\sum_{k=1}^K x_k + \beta\right)\right). \end{aligned} \quad (1.7)$$

Thus, the hyperparameters of the gamma posterior of λ are

$$\begin{aligned} \alpha &\rightarrow \alpha + K, \\ \beta &\rightarrow \beta + \sum_{k=1}^K x_k. \end{aligned}$$

Gaussian likelihood and Inverse gamma prior

Other members that, too, belong to the exponential family are the Gaussian (normal) and Inverse gamma distributions. Again, consider K i.i.d. observations drawn from the Gaussian likelihood function with the known μ parameter and the unknown variance, $p(\mathbf{x}|\mu, \sigma^2) = \prod_{k=1}^K \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_k-\mu)^2}{2\sigma^2}\right)$. Now we choose the prior on the variance as the inverse gamma distribution, $p(\sigma^2) = \frac{\beta^\alpha}{\Gamma(\alpha)} (\sigma^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\sigma^2}\right)$, where $\sigma^2 \in \mathbb{R}^+$ and α, β are hyperparameters.

After inserting these expressions into Eq. (1.1) and using proportionality sign, \propto , we obtain the unnormalized posterior in the form of the inverse gamma distribution:

$$p(\sigma^2|\mathbf{x}, \mu) = \frac{\frac{1}{(2\pi\sigma^2)^{K/2}} \exp\left(-\frac{1}{2} \sum_{k=1}^K \frac{(x_k - \mu)^2}{\sigma^2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} (\sigma^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\sigma^2}\right)}{\int_0^{+\infty} \frac{1}{(2\pi\sigma^2)^{K/2}} \exp\left(-\frac{1}{2} \sum_{k=1}^K \frac{(x_k - \mu)^2}{\sigma^2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} (\sigma^2)^{-\alpha-1} \exp\left(-\frac{\beta}{\sigma^2}\right) d\sigma^2} \quad (1.8)$$

$$\propto (\sigma^2)^{-\alpha - \frac{K}{2} - 1} \exp\left(-\frac{1}{\sigma^2} \left(\frac{\sum_{k=1}^K (x_k - \mu)^2}{2} + \beta\right)\right).$$

Thus, the hyperparameters of the inverse gamma posterior of σ^2 are

$$\alpha \rightarrow \alpha + \frac{K}{2},$$

$$\beta \rightarrow \beta + \frac{\sum_{k=1}^K (x_k - \mu)^2}{2}.$$

1.3 Non-informative Priors

In some applications, it is possible to have prior knowledge about the parameters of the model, often acquired via an expert's opinion on the real-world system. For example, if the prior assigns zero probability to a certain value of a random variable, then the posterior distribution necessarily assigns zero probability to that value, irrespective of any subsequent observations of data [4]. This forces a sparse parameterization of the model (but more on that later).

On the other hand, if we do not know anything about the parameters, θ , and do not want to introduce new information into the model, we can use non-informative priors. For example, when nothing is known about θ in advance, we choose $p(\theta)$ in the form of the uniform distribution, which assigns the same probability to all possible outcomes of θ [49]. Because the domain of the parameter, θ , may be unbounded, the problem with proper normalization of $p(\theta)$ arises, i.e., the integral over θ may diverge. Such priors are called *improper*, and, in practise, they are used provided that the corresponding posterior can be normalized [4].

In such cases, the so-called *Jeffreys prior* is widely adopted. It is defined as follows:

$$p(\theta) \propto \sqrt{\det(\mathbf{F}(\theta))}, \quad (1.9)$$

where \mathbf{F} stands for Fisher information matrix. Its key feature is the invariance under any differentiable transformation [44].

1.4 Shrinkage Priors

The main focus of this thesis is to find parameterizations of probabilistic models, as this allows us to achieve less complexity and better interpretability of its outputs. Moreover, in high-dimensional models with many predictors, the *overfitting* is often encountered. Penalization techniques are then used to counteract this phenomenon. Specifically, the so-called *shrinkage priors* aim to shrink small effects to zero while maintaining dominant and large effects [53]. In other words, we want as many non-significant parameters as possible to be zero, which will reduce model complexity and increase its interpretability.

1.4.1 Automatic Relevance Determination

Automatic Relevance Determination (ARD) is a technique in Bayesian framework, which requires the use of a certain type of prior on a parameter whose relevance needs to be determined. We usually place Gaussian priors on parameters and then hyper-priors on their scales [30]. If we set the prior as the Gaussian distribution with the zero mean vector, then the main benefit of ARD is that any unnecessary parameters are automatically forced to zero [55]. We can write the prior for all the parameters in the probabilistic model, θ , including their possible hyperparameters, α , as the integral over their joint distribution

$$p(\theta, \alpha) \longrightarrow p(\theta) = \int p(\theta, \alpha) d\alpha = \int p(\theta|\alpha)p(\alpha) d\alpha, \quad (1.10)$$

where $p(\theta|\alpha)$ is chosen as Gaussian. ARD priors are also referred to as *Gaussian scale mixtures* (GSM) [3], which is defined to be a zero mean Gaussian conditional on its scales with fixed variance σ^2

$$p(\theta_j) = \int \underbrace{\mathcal{N}(0, \alpha_j^2 \sigma^2)}_{\text{hierarchical parameterization}} p(\alpha_j) d\alpha_j, \quad (1.11)$$

where $j = 1, \dots, J$. This parametrization is also called *hierarchical*.

Spike and Slab prior

Mixture priors with spike and slab components are widely used for predictor selection. The spike component, which concentrates its mass around values close to zero, allows us to shrink the small effects to zero, whereas the slab component has its mass spread over a wide range of plausible values of the parameters [32]. Let $\theta \in \mathbb{R}^J$ be parameters with hyperparameters α acting element-wisely on fixed variance σ^2 and

$$\begin{aligned} (\theta_j|\alpha_j, \sigma^2) &\sim \mathcal{N}(0, \alpha_j^2 \sigma^2), \\ \alpha_j &\sim \text{Bernoulli}(\pi). \end{aligned} \quad (1.12)$$

By substituting Eq. (1.12) into the Eq. (1.11), we obtain

$$p_{\text{spike and slab}}(\theta_j) = \sum_{\alpha_j \in \{0,1\}} \mathcal{N}(\theta_j|\alpha_j^2 \sigma^2) p(\alpha_j) = \pi \mathcal{N}(\theta_j|0, \sigma^2) + (1 - \pi) \delta(\theta_j), \quad (1.13)$$

where $\delta(\cdot)$ denotes the Dirac function. Proof of convergence $\mathcal{N}(\theta_j|0, \alpha_j^2 \sigma^2) \rightarrow \delta(\theta_j)$ for $\alpha_j \rightarrow 0$ in space of generalized functions (or distributions) can be found in [26]. The random variable $\alpha_j^2 \sim \text{Bernoulli}(\pi)$ implies the prior on Gaussian's variance to be also Bernoulli [30].

Laplace prior

One of the most famous shrinkage priors is the Laplace prior. Using the exponential mixing with prior α_j^2 on the variance σ^2 , $\alpha_j^2 \sim \text{Exp}(2)$,

$$\begin{aligned} (\theta_j|\alpha_j^2, \sigma^2) &\sim \mathcal{N}(0, \alpha_j^2 \sigma^2), \\ \alpha_j^2 &\sim \text{Exp}(2), \end{aligned} \quad (1.14)$$

we can obtain the corresponding prior on the standard deviation α_j to express the Laplace prior in the form of the GSM. Based on the proof in [27], such a standard-deviation prior is the Rayleigh distribution with its scale parameter equal to 1, $p(\alpha_j) = \alpha_j \exp\left(-\frac{\alpha_j^2}{2}\right)$. Eventually, we get the Laplace prior formula

$$p_{\text{Laplace}}(\theta_j) = \int \mathcal{N}(0, \alpha_j^2 \sigma^2) \text{Rayleigh}(1) d\alpha_j. \quad (1.15)$$

Student-t prior

Another prior distribution we are concerned with is the Student-t prior [3] with zero mean, ν degrees of freedom and the scale parameter $\frac{\sigma^2}{\lambda}$:

$$\begin{aligned} (\theta_j | \alpha_j^2, \sigma^2) &\sim \mathcal{N}(0, \alpha_j^2 \sigma^2), \\ (\alpha_j^2 | \nu, \lambda) &\sim \text{Inverse-Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2\lambda}\right), \\ \lambda &\sim \text{Half-Cauchy}(0, 1). \end{aligned} \quad (1.16)$$

For the inverse-gamma prior on the variance σ^2 , there exists a corresponding prior on the standard deviation in the form of the inverse-Nakagami distribution [29]. Larger values of λ result in more shrinkage to the mean. Student-t prior can also be expressed in the form of GSM

$$p_{\text{Student-t}}(\theta_j | \nu, 0, \frac{\sigma^2}{\lambda}) = \int \mathcal{N}(0, \alpha_j^2 \sigma^2) \text{Inverse-Nakagami}\left(\frac{\nu}{2}, \frac{\nu^2}{4\lambda}\right) d\alpha_j. \quad (1.17)$$

Horseshoe prior

We introduce the horseshoe prior originally proposed in [9] and according to our notation, put it into the concept of GSM. We write

$$\begin{aligned} (\theta_j | \alpha_j, \sigma^2) &\sim \mathcal{N}(0, \alpha_j^2 \sigma^2) \\ \alpha_j &\sim \text{Half-Cauchy}(0, 1), \end{aligned} \quad (1.18)$$

where Half-Cauchy distribution describes the standard deviation α_j . The corresponding prior on the variance cannot be analytically expressed but, there is no problem to write horseshoe prior as GSM:

$$p_{\text{horseshoe}}(\theta_j) = \int \mathcal{N}(0, \alpha_j^2 \sigma^2) \text{Half-Cauchy}(0, 1) d\alpha_j. \quad (1.19)$$

The prior distributions on the standard deviation or the variance discussed in subsection, including their corresponding GSM priors, are summarized in Table 1.1.

Standard deviation prior $p(\alpha_j)$	Variance prior $p(\alpha_j^2)$	Marginal prior $p(\theta_j)$
Bernoulli	Bernoulli	Spike and Slab
Rayleigh	Exponential	Laplace
Inverse-Nakagami	Inverse-Gamma	Student-t
Half-Cauchy	Unnamed	Horseshoe

Table 1.1: Standard deviation priors $p(\alpha_j)$ and their corresponding GSM priors $p(\theta_j)$.

In practice, and, in the case of the following Bayesian regression example, it is common to choose the so-called *precision* hyperparameter as a multiplicative factor in front of the variance with the following, simple reparameterization:

$$\alpha_j^{-2} = \psi_j, j = 1, \dots, J. \quad (1.20)$$

For example, the inverse-Gamma prior then switches to the Gamma precision prior.

1.5 Variational Bayesian Framework

As already mentioned in Eq. (1.1) we can not always find the normalization constant for posterior (especially in higher dimensions). However, we still want to estimate it in some way so that we can capture as much uncertainty in the system as possible. Variational inference (or *Variational Bayes*) is therefore widely used to approximate the posterior for Bayesian models [5].

1.5.1 Setup and Goal of Variational Bayes

Let $\mathcal{D} = (\mathbf{y}, \mathbf{X})$ be an available dataset. Within Bayesian framework, the vector of latent variables \mathbf{z} has been adopted as a designation for the parameters known from above sections as θ . This is because we want to estimate not only the mapping between \mathbf{X} and \mathbf{y} , that are observed directly, but also hidden variables that are not but rather inferred from others [7]. Given the probabilistic model $p(\mathcal{D}, \mathbf{z})$ to model data \mathcal{D} using the latent variable vector \mathbf{z} . The goal of Variational Bayes is computing (or rather inferring) the posterior $p(\mathbf{z}|\mathcal{D})$ using Bayes's theorem.

1.5.2 KL Divergence and Evidence Lower Bound

Let us now modify and put the latent variables into Eq. (1.1):

$$p(\mathbf{z}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{z})p(\mathbf{z})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\mathbf{z})p(\mathbf{z})}{\int p(\mathcal{D}, \mathbf{z})d\mathbf{z}} = \frac{p(\mathcal{D}|\mathbf{z})p(\mathbf{z})}{\int p(\mathcal{D}|\mathbf{z})p(\mathbf{z})d\mathbf{z}}. \quad (1.21)$$

The problem here is the marginal, $p(\mathcal{D})$. In Eq. (1.7) or Eq. (1.8) we worked in terms of the proportional and later on we found normalization constants (or we recognized the kernel of distribution). In general, the dimensionality of the space and complexity of the integrand may prohibit numerical integration and not always we have a closed-form analytical solution [4]. We consider the integral simply intractable and if we do not have the marginal, then we do not have a full $p(\mathbf{z}|\mathcal{D})$.

We introduce the surrogate $q(\mathbf{z})$, with which we might be able to capture most of the posterior information and still be able to handle it:

$$q(\mathbf{z}) \approx p(\mathbf{z}|\mathcal{D}). \quad (1.22)$$

To find as good surrogate as possible we want to optimize over functions². We also need some metric to evaluate a goodness of the fit of $p(\mathbf{z}|\mathcal{D})$ and our surrogate. There are many such metrics (see [41] for more), but in this case we use the Kullback-Leibler (KL) divergence [19]:

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathcal{D})) = - \int q(\mathbf{z}) \log \left(\frac{p(\mathbf{z}|\mathcal{D})}{q(\mathbf{z})} \right) d\mathbf{z}. \quad (1.23)$$

The KL divergence result tells us how similar the two distributions are.

²The name of *Variational Bayes* consists of *variational* – optimization over functions (comes from variational calculus) and *Bayes* – within framework we mostly use Bayes's theorem.

Note. Technically speaking, KL divergence is not a true metric, as it generally does not satisfy the metric symmetry axiom, i.e.:

$$KL(q||p) \neq KL(p||q), \quad \forall p, q.$$

That is why it is called a divergence. But other metric axioms are fulfilled.

The problem then moves on to minimization of

$$q^*(\mathbf{z}) = \arg \min_{q(\mathbf{z}) \in \mathcal{Q}} KL(q(\mathbf{z})||p(\mathbf{z}|\mathcal{D})), \quad (1.24)$$

where \mathcal{Q} is some family of distribution (e.g. exponential). But we still do not have a solution to the problem in Eq. (1.21), because if we knew the posterior $p(\mathbf{z}|\mathcal{D})$, then we would not have to solve the minimization of the KL divergence. By expanding the definition (full proof can be found here [18]):

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathcal{D})) = - \underbrace{\int q(\mathbf{z}) \log \left(\frac{p(\mathbf{z}, \mathcal{D})}{q(\mathbf{z})} \right) d\mathbf{z}}_{\text{Evidence Lower Bound, } \mathcal{L}(q(\mathbf{z}))} + \log(p(\mathcal{D})) \quad (1.25)$$

we obtain the KL divergence and Evidence Lower Bound relationship. Logarithm of $p(\mathcal{D})$ is called *log-evidence* and although we do not have direct access to it, its value is always smaller or equal to zero. On the other hand, KL divergence is always a non-negative quantity. It follows that the ELBO is always smaller or equal to the log-evidence. The equality $\mathcal{L}(q(\mathbf{z})) = \log(p(\mathcal{D}))$ holds if and only if

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathcal{D})) = 0, \quad (1.26)$$

which means, that our surrogate is a true posterior. Now we can solve the variational Bayes task, because we found equivalent solveable problem to Eq. (1.24):

$$q^*(\mathbf{z}) = \arg \max_{q(\mathbf{z}) \in \mathcal{Q}} \mathcal{L}(q(\mathbf{z})). \quad (1.27)$$

This variation task is extensive just because the families can be really large and searching them all is a lengthy process. Therefore, in our case, we select one particular distribution from a particular family and numerically optimize its parameters. So, in Eq. (1.27) let $\mathcal{Q} = \{q_{\text{opt}}\}$, where q_{opt} is our chosen distribution (such as Gaussian, etc.) parameterized by some parameters we would like to optimize (for example $\boldsymbol{\eta}$) and the optimization problem then transitions to the problem of:

$$\boldsymbol{\eta}^* = \arg \max_{\boldsymbol{\eta}} \mathcal{L}(q_{\text{opt}}(\mathbf{z}|\boldsymbol{\eta})). \quad (1.28)$$

1.6 Bayesian Neural Networks

Given an observed dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$ and based on [42] we introduce a fully connected artificial neural network (ANN) with L hidden layers in the following way:

$$\begin{aligned} \mathbf{h}^{(0)} &= \mathbf{X}, \quad \mathbf{h}^{(l)} = \mathbf{a}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \dots, L \\ \mathbf{y} &= \mathbf{a}^{\text{out}}(\mathbf{W}^{(L+1)}\mathbf{h}^{(L)}), \end{aligned} \quad (1.29)$$

where $\mathbf{a}(\cdot)$ represents an *activation function*, $\mathbf{a}^{\text{out}}(\cdot)$ an *output activation function* and $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L+1}$ is a set of all parameters (weights and biases) in an ANN. The goal of using such an ANN described in Eq. (1.29)

is to find the proper mapping $\mathbf{y} = \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})$ from \mathbf{X} to \mathbf{y} and learn the parameters $\boldsymbol{\theta} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L+1}$ based on training epochs. However, within Bayesian framework, it is possible to model neural networks probabilistically and view the parameters as random variables for which we want to estimate uncertainty given the data. It is a supervised-learning task in machine learning, since we have labeled dataset.

Bayesian Neural Networks (BNN) utilize probabilistic layers with which we can capture uncertainty over all parameters in model (weights, biases and hyperparameters) and they are trained using Bayesian inference [20]. First we select the prior $p(\boldsymbol{\theta})$, which represents our prior belief of the parameters configuration (to force sparsity in BNN it could be used one of the shrinkage priors), choose the model likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ (depending on regression or classification task) and via variational inference infer the posterior of all parameters.

1.6.1 Activation Functions

In every hidden and output layer of neural networks the activation function can be found. This is a transformation of the weighted inputs, the output of which follows to the next layer. If we use other than linear activation function, it can help the network to learn complex data, compute and learn almost any dependence between inputs and output [31]. The activation functions are here written in form of vector functions, since their output is generally a vector or even matrix, not a single number. But each of its components is the same in the particular layer.

Here are five examples of activation functions (taken mostly from [4]) used in this work:

1. Identity

The simplest activation function is the linear identity function, which is defined as $f_{\text{Id}}(x) = x$. We list it here only for interest, since if we choose it as an activation to the output layer in a network without any hidden layer, we get a classical linear regression problem in relation to Eq. (1.29):

$$\mathbf{y} = \mathbf{a}^{\text{out}}(\mathbf{X}\boldsymbol{\theta}), \quad \text{where } a_n^{\text{out}}(\cdot) = f_{\text{Id}}(\cdot) \quad \forall n = 1, \dots, N. \quad (1.30)$$

2. Rectified Linear Unit (ReLU)

ReLU is widely used in hidden layers in deep networks, because of its computational efficiency, non-linearity and simplicity [1]. We define the ReLU activation in each its component as

$$f_{\text{ReLU}}(x) = \max\{0, x\}, \quad \text{collectively as } \mathbf{a}(\cdot) = \mathbf{f}_{\text{ReLU}}(\cdot). \quad (1.31)$$

We just need to pick the maximum and not to perform expensive exponential operations as in *sigmoid*.

3. Sigmoid (Logistic function)

Logistic function is used as an activation mostly in output layer in logistic regression, when we would like to classify the output into two classes. It was already proposed in Eq. (1.3) and plotted in Fig. 1.1. We define the logistic activation function as

$$f_{\text{sigmoid}}(x) = \frac{1}{1 + \exp(-x)}, \quad \text{collectively as } \mathbf{a}^{\text{out}}(\cdot) = \mathbf{f}_{\text{sigmoid}}(\cdot). \quad (1.32)$$

Its good feature is output scaling between zero and one, which can also be interpreted as a probability.

4. Hyperbolic tangent

Hyperbolic tangent activation is often used as an alternative to logistic function, because of its same S-shape. It is defined, as follows

$$f_{\tanh}(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \tanh(x), \text{ collectively as } \mathbf{a}^{\text{out}}(\cdot) = \mathbf{f}_{\tanh}(\cdot). \quad (1.33)$$

Unlike the logistic function, the output of this activation function is scaled between minus one and one [4].

5. Softmax

A generalized version of the logistic function is the softmax function, which is often found in the output layer of multi-class classification up to K classes. This extends the properties of the logistic function, so that each value of the output component in softmax is restricted to the interval zero to one and their total sum equals to one. Softmax function $\mathbf{f}_{\text{softmax}} : \mathbb{R}^K \rightarrow (0, 1)^K$ is defined, as follows

$$f_{\text{softmax}}(\mathbf{x})_n = \frac{\exp(x_n)}{\sum_{k=1}^K \exp(x_k)} \quad \forall n = 1, \dots, N, \text{ collectively as } \mathbf{a}^{\text{out}}(\cdot) = \mathbf{f}_{\text{softmax}}(\cdot). \quad (1.34)$$

1.6.2 Point Estimates

We assume that the observed data is i.i.d with the likelihood

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^N p(y_n|\mathbf{X}_n, \boldsymbol{\theta}), \quad (1.35)$$

where \mathbf{X}_n represents the n -th row of data matrix \mathbf{X} (i.e. one observation of K predictors). The parameters can be learnt by maximum likelihood estimation (MLE) principle

$$\boldsymbol{\theta}^{\text{MLE}} = \arg \max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) = \arg \min_{\boldsymbol{\theta}} -\log p(\mathcal{D}|\boldsymbol{\theta}). \quad (1.36)$$

If we add a shrinkage prior $p(\boldsymbol{\theta})$ from Table 1.1 to the likelihood, we obtain a regularization, which penalizes the parameters during the training [6]:

$$\boldsymbol{\theta}_{\text{penalized}}^{\text{MLE}} = \arg \max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \arg \min_{\boldsymbol{\theta}} -(\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})). \quad (1.37)$$

The objective in Eq. (1.36) and Eq. (1.37) is called *loss function* (or *cost function*) [4], which we would like to optimize. This task is typically achieved by gradient-based methods (*optimizers*). The disadvantage of this approach is only the point estimate of the parameters, which does not tell us much about their uncertainty.

1.6.3 Posterior Distribution

To capture the uncertainty in the whole network, posterior is then computed via Bayes' theorem described in Eq. (1.1) or approximated via maximizing the ELBO and represents the updated belief of how likely the network parameters are given the observations. We can then calculate its moments and it also can be used to predict the new response y_{N+1} of an unseen input \mathbf{X}_{N+1} using the predictive distribution [42]

$$p(\mathbf{y}_{N+1}|\mathbf{X}_{N+1}, \mathcal{D}) = \int p(y_{N+1}|\mathbf{X}_{N+1}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}, \quad (1.38)$$

or

$$p(\mathbf{y}_{N+1}|\mathbf{X}_{N+1}, \mathcal{D}) \approx \int p(y_{N+1}|\mathbf{X}_{N+1}, \boldsymbol{\theta})q(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \quad (1.39)$$

1.7 Bayesian Ridge Regression

In this section, an analytically solvable posterior search problem as an alternative to basic linear regression and its subsequent shrinkage priors upgrade will be presented. It will also be outlined how this problem is related to BNN.

Given the dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$, setting $L = 0$ (no hidden layers), $\mathbf{a}^{\text{out}}(\cdot) = \mathbf{f}_{\text{Id}}(\cdot)$ in Eq. (1.29) and adding a Gaussian *noise* \mathbf{e} of the mean $\boldsymbol{\mu}_e$ and variance σ_e^2 will give us the regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \mathbf{e}, \quad (1.40)$$

where $\mathbf{y} \in \mathbb{R}^N$ and the data matrix \mathbf{X} is $(N \times K + 1)$ -dimensional matrix in case with bias (or intercept in the regression model) and \mathbf{X} is $(N \times K)$ -dimensional without it. Based on this fact, we can estimate up to $K + 1$ (or K in case without bias) parameters in the model. Vector $\boldsymbol{\theta}$ denotes the model parameters (weights and bias) and \mathbf{e} represents independent model noise with distribution $p(\mathbf{e}) = \mathcal{N}(\mathbf{0}, \sigma_e^2 \cdot \mathbf{I})$. Thanks to that, the conditional distribution for \mathbf{y} will be

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \sigma_e^2) = \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \sigma_e^2 \cdot \mathbf{I}) \propto \frac{1}{(\sigma_e^2)^{N/2}} \exp\left(-\frac{1}{2}\sigma_e^{-2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right). \quad (1.41)$$

We will now not consider bias, whereby we can estimate up to K parameters (purely just weights) in the model. We introduce the prior distribution for parameters with known precision ψ and known common measurement precision $\omega = \frac{1}{\sigma_e^2}$:

$$\begin{aligned} p(\boldsymbol{\theta}|\psi) &= \mathcal{N}(\mathbf{0}, \psi^{-1} \cdot \mathbf{I}) \propto \psi^K \exp\left(-\frac{1}{2}\psi\boldsymbol{\theta}^T\boldsymbol{\theta}\right), \\ p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \omega) &= \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \omega^{-1} \cdot \mathbf{I}) \propto \omega^{N/2} \exp\left(-\frac{1}{2}\omega(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right). \end{aligned} \quad (1.42)$$

In this such a simple example, when ψ and ω are known, the posterior for weights can be calculated analytically (inspiration for proof comes from [50]):

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \psi, \omega) = \mathcal{N}\left(\left(\mathbf{X}^T\mathbf{X} + \psi \cdot \mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}, \omega\left(\mathbf{X}^T\mathbf{X} + \psi \cdot \mathbf{I}\right)^{-1}\right). \quad (1.43)$$

1.7.1 Hyperparameter ψ Prior

Let's slightly modify Eq. (1.42), when ψ will be the unknown hyperparameter, to which we need to assign the prior

$$\begin{aligned} (\boldsymbol{\theta}|\psi) &\sim \mathcal{N}(\mathbf{0}, \psi^{-1} \cdot \mathbf{I}), \\ \psi &\sim \text{Gamma}(\gamma_0, \delta_0). \end{aligned} \quad (1.44)$$

We build the model likelihood according to Eq. (1.35):

$$\begin{aligned} p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}, \omega, \psi) &= p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}, \omega)p(\boldsymbol{\theta}|\psi) = \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \omega^{-1} \cdot \mathbf{I})\mathcal{N}(\mathbf{0}, \psi^{-1} \cdot \mathbf{I}) \\ &\propto \omega^{N/2}\psi^{K/2} \exp\left(-\frac{1}{2}\omega(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\right) \exp\left(-\frac{1}{2}\psi\boldsymbol{\theta}^T\boldsymbol{\theta}\right) \\ &\propto \omega^{N/2}\psi^{K/2} \exp\left(-\frac{1}{2}\omega\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 - \frac{1}{2}\psi\|\boldsymbol{\theta}\|_2^2\right). \end{aligned} \quad (1.45)$$

Note. In Eq. (1.45) the symbol $\|\cdot\|_2$ is introduced. Let's clarify, what it means. We define L_p norm [2] of vector $\mathbf{x} \in \mathbb{R}^N$ as

$$\|\mathbf{x}\|_p = \left(\sum_{n=1}^N |x_n|^p \right)^{1/p},$$

with special case of $p = 2$: $\|\mathbf{x}\|_2 = \left(\sum_{n=1}^N |x_n|^2 \right)^{1/2} = \sqrt{x_1^2 + \dots + x_N^2}$, which is an *Euclidian norm* on \mathbb{R}^N space.

By adding the predefined prior from Eq. (1.44) to Eq. (1.45), we obtain the probability model:

$$\begin{aligned} p(\mathbf{y}, \boldsymbol{\theta}, \psi | \mathbf{X}, \omega) &= p(\mathbf{y} | \boldsymbol{\theta}, \mathbf{X}, \omega) p(\boldsymbol{\theta} | \psi) p(\psi) = \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \omega^{-1} \cdot \mathbf{I}) \mathcal{N}(\boldsymbol{\theta}, \psi^{-1} \cdot \mathbf{I}) \text{Gamma}(\gamma_0, \delta_0) \\ &\propto \omega^{N/2} \psi^{K/2} \exp\left(-\frac{1}{2}\omega\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 - \frac{1}{2}\psi\|\boldsymbol{\theta}\|_2^2\right) \frac{\delta_0^{\gamma_0}}{\Gamma(\gamma_0)} \psi^{\gamma_0-1} \exp(-\delta_0\psi) \\ &\propto \omega^{N/2} \psi^{K/2+\gamma_0-1} \exp\left(-\frac{1}{2}\omega\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 - \frac{1}{2}\psi\|\boldsymbol{\theta}\|_2^2 - \delta_0\psi\right). \end{aligned} \quad (1.46)$$

Finally, we use Eq. (1.1) to express the unnormalized posterior of all available parameters:

$$p(\boldsymbol{\theta}, \psi | \mathbf{y}, \mathbf{X}, \omega) \propto \psi^{K/2+\gamma_0-1} \exp\left(-\frac{1}{2}\omega\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 - \frac{1}{2}\psi\|\boldsymbol{\theta}\|_2^2 - \delta_0\psi\right). \quad (1.47)$$

To choose the proper ψ we need to be careful. Values that are too small or too large can be misleading and the model will not be correct. Thus, a several ways how to estimate ψ have been introduced:

1. *k-Fold Cross-validation*

This method is based on dividing the dataset into k equal parts. Then we take one part of them as a test data and the remaining groups as a training data. We fit a model on the training set and evaluate on test data while retaining the evaluation score (or our monitored metrics). Eventually we discard the model and the whole process repeats itself. The proper ψ is then chosen based on our preferences of metrics values results [46].

2. *MAP estimate*

Taking a maximum argument of marginal $p(\psi | \mathbf{y}, \mathbf{X}, \omega)$ leads us to *maximum a posteriori estimate* (MAP):

$$\hat{\psi} = \arg \max_{\psi} \int p(\psi, \boldsymbol{\theta} | \mathbf{y}, \mathbf{X}, \omega) d\boldsymbol{\theta}. \quad (1.48)$$

3. *L-curve*

It is a parametric plot of two functions that measure the size of the regularized solution and its corresponding residual. L-curve has a clear L-shaped corner located exactly, where the solution of ψ changes in nature from being dominated by the regularization to being dominated by the model error [15]. Thanks to that, we can choose such ψ , which makes our parameter $\boldsymbol{\theta}$ sparser, but still keeping an acceptable model error.

1.7.2 ARD in Bayesian Ridge Regression

By revisiting Eq. (1.44) and adding the K -dimensional precision *prior* $\boldsymbol{\psi}$ we rewrite it into a multi-dimensional shape

$$\begin{aligned} (\boldsymbol{\theta}|\boldsymbol{\psi}) &\sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\psi}^{-1}) \cdot \mathbf{I}), \\ \boldsymbol{\psi} &\sim p(\boldsymbol{\psi}) = \prod_{j=1}^K p(\psi_j), \end{aligned} \quad (1.49)$$

with the joint distribution $p(\boldsymbol{\theta}, \boldsymbol{\psi}) = p(\boldsymbol{\theta}|\boldsymbol{\psi})p(\boldsymbol{\psi})$. In this case of the multidimensional hyperparameter $\boldsymbol{\psi}$ a slight change occurs in the probability model in Eq. (1.46):

$$p(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{X}, \omega) = \mathcal{N}(\mathbf{X}\boldsymbol{\theta}, \omega^{-1} \cdot \mathbf{I}) \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\psi}^{-1}) \cdot \mathbf{I}) \prod_{j=1}^K p(\psi_j). \quad (1.50)$$

By replacing $p(\psi_j)$ with the Gamma precision prior distribution according to Eq. (1.20), we force a sparse parameterization. The task of finding the posterior $p(\boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{y}, \mathbf{X}, \omega)$ can also (and more appropriately) be solved within a variational framework with the surrogate distribution $q(\mathbf{z})$.

1.8 Bayesian Logistic Regression

Another simple example of BNN with no hidden layers ($L = 0$) and one output layer with the sigmoid activation $\mathbf{a}^{\text{out}}(\cdot) = \mathbf{f}_{\text{sigmoid}}(\cdot)$ will be outlined as an alternative to the basic logistic regression.

Consider a binary response variable y_n , which is assumed to have a Bernoulli distribution,

$$y_n \sim \text{Bernoulli}(\pi_n), \quad \mathbb{E}[y_n] = \pi_n, \quad n \in 1, \dots, N, \quad (1.51)$$

where $\pi_n \in [0, 1]$. Let the dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$ consist of the response variable $\mathbf{y} \in \{0, 1\}^N$ and the data matrix $\mathbf{X} \in \mathbb{R}^{N \times K+1}$ and $\boldsymbol{\theta} \in \mathbb{R}^{K+1}$ be a vector of the model parameters we would like to estimate. The logistic regression model writes that the *logit* (see also Fig. 1.1) of the probability π_n is a linear function of the predictors $\mathbf{X}_n = (x_{n,1}, \dots, x_{n,K})$ [16]

$$\text{logit}(\pi_n) = \log\left(\frac{\pi_n}{1 - \pi_n}\right) = \theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK}. \quad (1.52)$$

When we put together the equations described above, we obtain

$$\begin{aligned} \mathbb{E}[y_n|\mathbf{x}_n] &= \pi_n = \text{logit}^{-1}(\theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK}) \\ &= \frac{\exp(\theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK})}{1 + \exp(\theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK})}, \end{aligned} \quad (1.53)$$

where inverse mapping of logit denotes the logistic or sigmoid function. Here we can see the connection between the logistic regression and the derived natural parameters of the Bernoulli distribution from Eq. (1.3).

Note. Note that π_n represents $\pi(\mathbf{x}_n)$, where

$$\pi_n := \pi(\mathbf{x}_n) = \frac{\exp(\theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK})}{1 + \exp(\theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \dots + \theta_K x_{nK})}.$$

1.8.1 Likelihood Function and Formation of Posterior

Let y_1, \dots, y_N be random variables sampled from Bernoulli distribution, $p(\mathbf{y}|\boldsymbol{\theta}) = \text{Bernoulli}(\boldsymbol{\pi})$. Since we assume i.i.d. variables, we can write

$$\text{likelihood}(\boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{n=1}^N p(y_n|\boldsymbol{\theta}) = \prod_{n=1}^N \pi(\mathbf{x}_n)^{y_n} (1 - \pi(\mathbf{x}_n))^{1-y_n}. \quad (1.54)$$

By taking a negative logarithm of Eq. (1.54), we obtain the *loss function* (or an objective function to be minimized) for model parameters

$$\text{loss}(\boldsymbol{\theta}) = - \sum_{n=1}^N (y_n \log(\pi_n) + (1 - y_n) \log(1 - \pi_n)), \quad (1.55)$$

also known as *binary cross-entropy function* [13].

Now we can choose the prior to our likelihood, in order to be able to calculate the posterior, since the Bayes' theorem from Eq. (1.1) holds. For example, if we use the Laplace prior on parameters described in Eq. (1.15), we get the equivalent expression of Bayesian LASSO regression [14], which will cause that instead of Eq. (1.55), the optimization task will be

$$\text{loss}_{L1}(\boldsymbol{\theta}) = \text{loss}(\boldsymbol{\theta}) + \lambda \sum_{k=0}^K |\theta_k|, \quad (1.56)$$

with hyperparameter λ , which can be estimated in similar ways as in the previous Section 1.7 on Bayesian linear regression.

Exact Bayesian inference for the logistic regression is intractable. In particular, evaluation of the posterior would require normalization of the product of the prior and the likelihood that itself comprises a product of logistic sigmoid functions, one for every data point. Evaluation of the predictive distribution is similarly intractable [4].

1.9 Multi-Instance Learning

Another, slightly more complex type of supervised-learning task is *multi-instance learning* or MIL. MIL formalism assumes that each object in the real world (or *sample*) is represented by a set (*bag*) of feature vectors (*instances*) of fixed length, where knowledge about that object is available on bag level, but not necessarily on instance level [39]. Let us now set the terminology and graphical explanation of the difference between the conventional machine learning and MIL.

Consider an instance $\mathbf{x}_i \in \mathbb{R}^K$ and a parameter space Θ . We define bag as a set of several instances, i.e., $b = \{\mathbf{x}_i \in \mathbb{R}^K | i = 1, \dots, |b|\}$, which can be arbitrarily large and also empty. Therefore, we denote $|b| \in \mathbb{N}_0$ the size of bag b . We can write the whole space of bags as $\mathcal{B} = \bigcup_{|b| \geq 0} \mathbb{R}^{K \times |b|}$. In general, we do not have to have directly labeled instances, but rather labeled bags. Thus, each bag $b \in \mathcal{B}$ has its own label y [33]. We will use MIL mainly for classification tasks with supervised setting and therefore each label of each bag will come from a finite set \mathcal{C} (for example $\{0, 1\}$).

We define MIL models, as follows

$$f : \mathcal{B} \rightarrow \mathcal{C}, \quad (1.57)$$

where the goal is to learn the *classifier* f based on the parameters $\boldsymbol{\theta} \in \Theta$ and available dataset with bags and their labels $\mathcal{D} = \{(b_n, y_n) \in \mathcal{B} \times \mathcal{C} | n = 1, \dots, N\}$.

Note. Thus, in the available dataset we have a total of N bags together containing I instances each of dimension K with N corresponding labels, as in the case of the linear regression, logistic regression or neural networks, but each bag can be a different (but fixed) size. Here is the difference between the conventional machine learning and MIL, which is graphically described in Fig. 1.2 and Fig. 1.3.

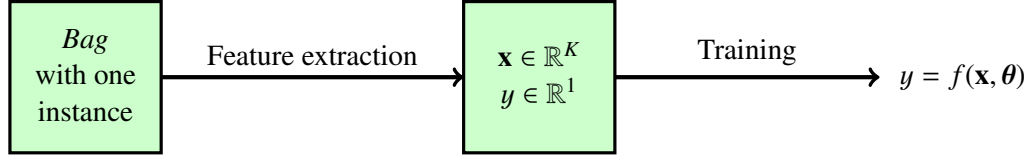


Figure 1.2: By setting $|b| = 1$ the task of MIL switches to the conventional machine learning.

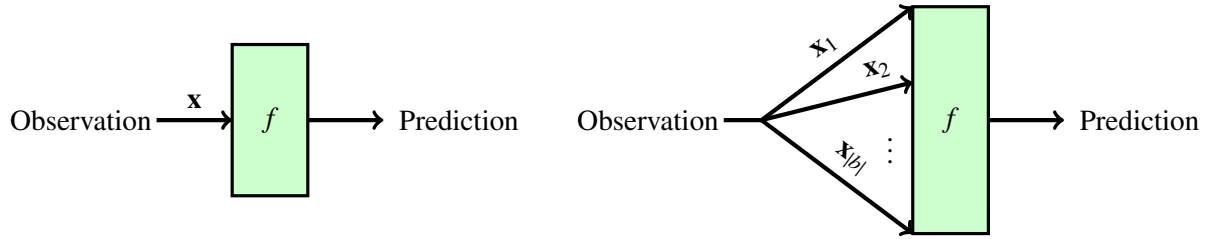


Figure 1.3: Difference between the conventional machine learning and MIL (taken from [33]).

1.9.1 Embedded-Space Paradigm

We propose an approach that embed each bag into a M -dimensional vector space \mathbb{R}^M and represent it in a much more useful form to be able to use standard machine learning techniques [48]. An embedding of the bag b can be written as

$$\varphi : \mathcal{B} \rightarrow \mathbb{R}^M, \quad (1.58)$$

with individual projection $\varphi_m = g(\{d(\mathbf{x}, \theta_m)\}_{\mathbf{x} \in b})$, $m = 1, \dots, M$, where $d : \mathbb{R}^K \times \Theta \rightarrow \mathbb{R}_0^+$ is some suitably distance function parameterized by $\theta \in \Theta$ and g stands for *aggregation function* [38].

The aggregation (or *pooling*) function g can be fixed (such as maximum, minimum or mean aggregation) or any other, to which we are able to compute the gradients with respect to its inputs. With appropriate choice of these functions parameterized by θ we can create a neural network with one or more layers that implement a set of distance functions followed by *pooling* layer. The main feature of the pooling function is that its output is always a vector of fixed dimension. Then we add the output layer (such as softmax, etc. – depends on our task) and optimize the neural network. A sketch of this idea is outlined in Fig. 1.4.

The only thing that changes in the optimization is that we have a slightly different network architecture (by adding the pooling layer) and a different dataset structure. In any case, we are trying to find a mapping based on the optimized parameters, so using sparse methods like L_1 regularization, etc. should work exactly the same.

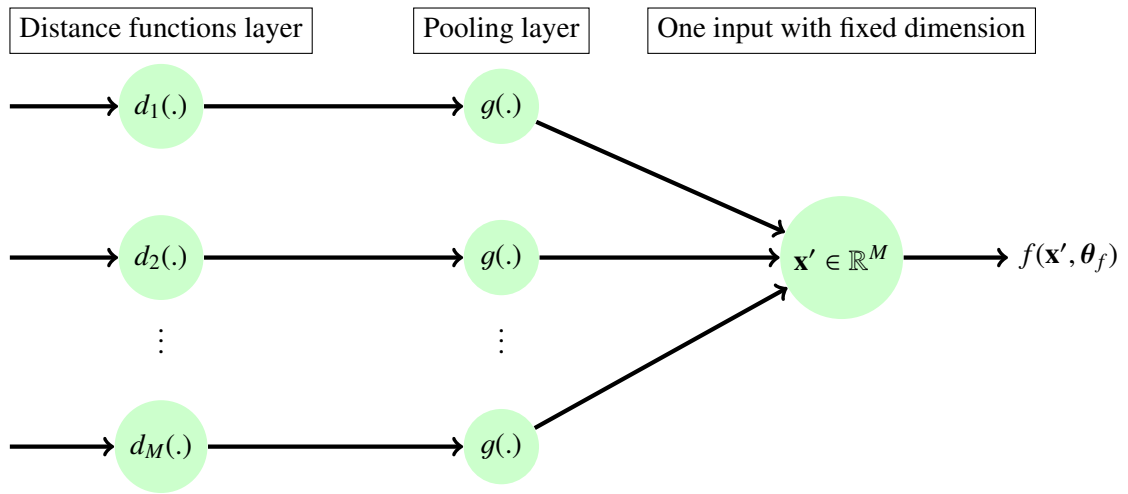


Figure 1.4: Sketch of the neural network optimizing the embedding (taken from [39]).

Chapter 2

Variational Optimization

2.1 Natural-Gradient Variational Inference

Consider likelihood $p(\mathcal{D}|\mathbf{z})$ and prior $p(\mathbf{z})$. As proposed in Eq. (1.2), we choose the surrogate posterior from the exponential family, but now parameterized by its natural parameters in the terms of the latent variables:

$$q_{\boldsymbol{\eta}}(\mathbf{z}) = q(\mathbf{z}|\boldsymbol{\eta}) = k(\mathbf{z}) \exp(\langle \mathbf{t}(\mathbf{z}), \boldsymbol{\eta} \rangle - \phi(\boldsymbol{\eta})). \quad (2.1)$$

Such parameterized approximation written above can be estimated by maximizing the ELBO:

$$\max_{\boldsymbol{\eta}} \mathcal{L}(\boldsymbol{\eta}) = \max_{\boldsymbol{\eta}} \mathbb{E}_q [\log p(\mathcal{D}, \mathbf{z}) - \log q(\mathbf{z}|\boldsymbol{\eta})], \quad (2.2)$$

which can be solved via *gradient descent* that is simple and convenient to implement by using modern automatic-differentiation methods and the reparametrization trick.

We propose an alternative approach and express the gradient descent in natural-parameter space (also called *natural-gradient descent*), which exploits the information geometry of q to speed-up the convergence and accelerate the conventional first-order gradient optimization by scaling the gradient with the Fisher information matrix (FIM), especially when the FIM is well-conditioned [28]. The natural-gradient descent step in the natural-parameter space is given as follows:

$$\boldsymbol{\eta}_{(t+1)} = \boldsymbol{\eta}_{(t)} + \beta_{(t)} [\mathbf{F}(\boldsymbol{\eta}_{(t)})]^{-1} \nabla_{\boldsymbol{\eta}} \mathcal{L}(\boldsymbol{\eta}_{(t)}), \quad (2.3)$$

where $\mathbf{F}(\cdot)$ is the FIM of $q(\mathbf{z}|\boldsymbol{\eta})$, β the learning rate and t is the iteration.

2.1.1 Implementation

The first problem that may occur in the implementation of Eq. (2.3), is the FIM inversion, which can be quite computationally demanding. Nevertheless, for certain types of models, such as the mean-field variational inference in conjugate-exponential family models, using of natural gradients could be simpler than the ordinary gradients [17]. For exponential family approximations we can use the *expectation-parameter*, defined as the function

$$\mathbf{m}(\boldsymbol{\eta}) := \mathbb{E}_q [\mathbf{t}(\mathbf{z})] = \int q(\mathbf{z}|\boldsymbol{\eta}) \mathbf{t}(\mathbf{z}) d\mathbf{z}, \quad (2.4)$$

to compute natural-gradients. From [22] we assume a probabilistic framework, where each data example is sampled independently from the likelihood, $p(\mathcal{D}|\mathbf{z})$, and that the exponential family is in minimal

representation, which ensures that there exists one-to-one mapping between $\boldsymbol{\eta}$ and the expectation parameter \mathbf{m} . That leads us to expressing $\mathcal{L}(\boldsymbol{\eta})$ in term of \mathbf{m} , $\mathcal{L}_\star(\mathbf{m}) = \mathcal{L}(\boldsymbol{\eta})$. We also have to reparameterize $q_\eta \leftrightarrow q_m$ using the following relation

$$\nabla_\eta \mathcal{L}(\boldsymbol{\eta}) = [\nabla_\eta \mathbf{m}^T] \cdot \nabla_m \mathcal{L}_\star(\mathbf{m}) = [\mathbf{F}(\boldsymbol{\eta})] \cdot \nabla_m \mathcal{L}_\star(\mathbf{m}). \quad (2.5)$$

Note. The Jacobian of transformation described in Eq. (2.5) can be obtained by applying the chain rule for derivatives and using the exponential family properties: $\mathbf{F}(\boldsymbol{\eta}) = \nabla_\eta^2 \phi(\boldsymbol{\eta}) = \nabla_\eta \mathbf{m}^T$.

By composing Eq. (2.3) and Eq. (2.5), we obtain a simple update for the natural-gradient descent, where the gradient is computed with respect to the expectation-parameter \mathbf{m} , but taking a step in the natural-parameter space

$$\boldsymbol{\eta}_{(t+1)} = \boldsymbol{\eta}_{(t)} + \beta_{(t)} \nabla_m \mathcal{L}_\star(\mathbf{m}_{(t)}), \quad (2.6)$$

which avoids the difficult computation of the inverse FIM.

2.2 Mean-Field Variational Inference

Another way how to restrict the family of distributions \mathcal{Q} for the posterior described in Eq. (1.27) and exploit the potential of the expectation-parameter is using the factorized distributions. This factorized form of variational inference is called the *mean-field* approximation [4]. By sticking to the general latent variables \mathbf{z} , model parameters (*weights* and *biases*) $\boldsymbol{\theta}$ and hyperparameters $\boldsymbol{\alpha}$ notation, we write this as

$$q(\mathbf{z}) \approx q(\boldsymbol{\theta})q(\boldsymbol{\alpha}) = \prod_{k=1}^K q_k(\theta_k) \prod_{j=1}^J q_j(\alpha_j). \quad (2.7)$$

We can reparameterize this with the natural parameters and express its natural-gradient updates simultaneously for both types of parameters

$$q_\eta(\mathbf{z}) \approx q_\eta(\boldsymbol{\theta})q_\eta(\boldsymbol{\alpha}). \quad (2.8)$$

Let us now use knowledge from above to derive these updates for the surrogate posterior from Section 1.7 within the mean-field framework. In Eq. (1.49) we selected the Gaussian prior on the model parameters and the Gamma prior on precision hyperparameters. So, Eq. (2.8) then passes on

$$q_\eta(\mathbf{z}) \approx q_\eta(\boldsymbol{\theta})q_\eta(\boldsymbol{\psi}). \quad (2.9)$$

2.2.1 Natural-Gradient Updates for Model Parameters

As the surrogate posterior for model parameters, we choose a Gaussian multivariate distribution with all its properties derived in Eq. (1.6), $q(\boldsymbol{\theta}|\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$ with the mean $\boldsymbol{\mu}_\theta$ and the covariance matrix $\boldsymbol{\Sigma}_\theta$:

$$\begin{aligned} \boldsymbol{\eta}_\theta^{(1)} &= \boldsymbol{\Sigma}_\theta^{-1} \boldsymbol{\mu}_\theta, & \mathbf{m}_\theta^{(1)} &= \mathbb{E}_q[\boldsymbol{\theta}] = \boldsymbol{\mu}_\theta, \\ \boldsymbol{\eta}_\theta^{(2)} &= -\frac{1}{2} \boldsymbol{\Sigma}_\theta^{-1}, & \mathbf{M}_\theta^{(2)} &= \mathbb{E}_q[\boldsymbol{\theta}\boldsymbol{\theta}^T] = \boldsymbol{\mu}_\theta \boldsymbol{\mu}_\theta^T + \boldsymbol{\Sigma}_\theta, \end{aligned} \quad (2.10)$$

where $\mathbf{m}_\theta^{(1)}, \mathbf{M}_\theta^{(2)}$ is the first and the second moment. We express the gradients with respect to the expectation-parameter defined in Eq. (2.10)

$$\begin{aligned} \nabla_{m_w^{(1)}} \mathcal{L}_\star &= \nabla_{\boldsymbol{\mu}_\theta} \mathcal{L} - 2 [\nabla_{\boldsymbol{\Sigma}_\theta} \mathcal{L}] \boldsymbol{\mu}_\theta, \\ \nabla_{M_\theta^{(2)}} \mathcal{L}_\star &= \nabla_{\boldsymbol{\Sigma}_\theta} \mathcal{L}. \end{aligned} \quad (2.11)$$

To make the notation more illustrative, the iterations will be marked as the second subscript of the iterated variable. We rewrite Eq. (2.6) in terms of Σ_θ

$$\begin{aligned} -\frac{1}{2}\Sigma_{\theta,(t+1)}^{-1} &= -\frac{1}{2}\Sigma_{\theta,(t)}^{-1} + \beta_{(t)} \cdot \nabla_{\Sigma_\theta} \mathcal{L}_{(t)} \\ \Sigma_{\theta,(t+1)}^{-1} &= \Sigma_{\theta,(t)}^{-1} - 2\beta_{(t)} \cdot \nabla_{\Sigma_\theta} \mathcal{L}_{(t)}, \end{aligned} \quad (2.12)$$

and in terms of μ_θ

$$\begin{aligned} \Sigma_{\theta,(t+1)}^{-1}\mu_{\theta,(t+1)} &= \Sigma_{\theta,(t)}^{-1}\mu_{\theta,(t)} + \beta_{(t)} \left[\nabla_{\mu_\theta} \mathcal{L}_{(t)} - 2(\nabla_{\Sigma_\theta} \mathcal{L}_{(t)})\mu_{\theta,(t)} \right] \\ \mu_{\theta,(t+1)} &= \Sigma_{\theta,(t+1)} \left[\Sigma_{\theta,(t)}^{-1}\mu_{\theta,(t)} + \beta_{(t)} \nabla_{\mu_\theta} \mathcal{L}_{(t)} - 2\beta_{(t)} (\nabla_{\Sigma_\theta} \mathcal{L}_{(t)})\mu_{\theta,(t)} \right] \\ \mu_{\theta,(t+1)} &= \Sigma_{\theta,(t+1)} \left[\left(\Sigma_{\theta,(t)}^{-1} - 2\beta_{(t)} \nabla_{\Sigma_\theta} \mathcal{L}_{(t)} \right) \mu_{\theta,(t)} + \beta_{(t)} \nabla_{\mu_\theta} \mathcal{L}_{(t)} \right] \\ \mu_{\theta,(t+1)} &= \Sigma_{\theta,(t+1)} \left[\Sigma_{\theta,(t+1)}^{-1}\mu_{\theta,(t)} + \beta_{(t)} \nabla_{\mu_\theta} \mathcal{L}_{(t)} \right] \\ \mu_{\theta,(t+1)} &= \mu_{\theta,(t)} + \beta_{(t)} \Sigma_{\theta,(t+1)} \left[\nabla_{\mu_\theta} \mathcal{L}_{(t)} \right], \end{aligned} \quad (2.13)$$

where $\nabla_x \mathcal{L}_{(t)}$ means the gradient of the ELBO \mathcal{L} with respect to a variable \mathbf{x} at iteration t . To sum up, via Eq. (2.12) and Eq. (2.13) we derived the natural-gradient updates of the Gaussian surrogate posterior for the model parameters.

In case of the mean-field approximation discussed above, we can define the covariance matrix in the Gaussian surrogate as $\Sigma_\theta = \text{diag}(\sigma_\theta^2)$ with variances on the diagonal, so that would correspond with Eq. (2.8):

$$\begin{aligned} \sigma_{\theta,(t+1)}^{-2} &= \sigma_{\theta,(t)}^{-2} - 2\beta_{(t)} \left[\nabla_{\sigma_\theta^2} \mathcal{L}_{(t)} \right], \\ \mu_{\theta,(t+1)} &= \mu_{\theta,(t)} + \beta_{(t)} \sigma_{\theta,(t+1)}^2 \circ \left[\nabla_{\mu_\theta} \mathcal{L}_{(t)} \right], \end{aligned} \quad (2.14)$$

where $\mathbf{a} \circ \mathbf{b}$ denotes the element-wise product between two vectors.

2.2.2 Natural-Gradient Updates for Hyperparameters

For the reparameterized precision hyperparameters in Eq. (1.20) we choose the Gamma surrogate posterior parameterized by its natural parameters. To simplify the problem, let $q_\eta(\psi_j) = \text{Gamma}(\psi_j | \gamma_j, \delta_j)$, $\forall j \in 1, \dots, J$. According to Eq. (2.7) and Eq. (2.8), we can write

$$q_\eta(\boldsymbol{\psi}) = \prod_{j=1}^J q_\eta(\psi_j) = \prod_{j=1}^J \frac{\delta_j^{\gamma_j}}{\Gamma(\gamma_j)} \psi_j^{\gamma_j-1} \exp(-\delta_j \psi_j) \quad (2.15)$$

and derive the natural-gradient updates of the Gamma surrogate posterior for the hyperparameters.

Again, we start by defining the natural parameters and the expectation-parameter from Eq. (2.4) of the Gamma surrogate following the same notation as in the Gaussian case:

$$\begin{aligned} \eta_{\psi_j}^{(1)} &= \gamma_j - 1, & m_{\psi_j}^{(1)} &= \mathbb{E}_q \left[\log(\psi_j) \right] = F(\gamma_j) - \log(\delta_j), \\ \eta_{\psi_j}^{(2)} &= \delta_j, & M_{\psi_j}^{(2)} &= \mathbb{E}_q \left[\psi_j \right] = \frac{\gamma_j}{\delta_j}. \end{aligned} \quad (2.16)$$

Note. The symbol $F(\cdot)$ denotes the *digamma function* defined as

$$F(x) = \frac{\partial}{\partial x} \log(\Gamma(x)),$$

where $\Gamma(\cdot)$ denotes the gamma function. Digamma function will be labeled this way, as in [36], to maintain consistency. We will also need the *trigamma function* defined as

$$F^{(1)}(x) = \frac{\partial^2}{\partial x^2} \log(\Gamma(x)).$$

To satisfy Eq. (2.5), it is necessary to calculate the FIM and then invert it:

$$\mathbf{F}(\boldsymbol{\eta}_\psi) = \begin{pmatrix} \mathbb{E} \left[-\frac{\partial^2}{\partial \gamma_j \partial \gamma_j} \log(q_\eta(\psi_j)) \right] & \mathbb{E} \left[-\frac{\partial^2}{\partial \gamma_j \partial \delta_j} \log(q_\eta(\psi_j)) \right] \\ \mathbb{E} \left[-\frac{\partial^2}{\partial \delta_j \partial \gamma_j} \log(q_\eta(\psi_j)) \right] & \mathbb{E} \left[-\frac{\partial^2}{\partial \delta_j \partial \delta_j} \log(q_\eta(\psi_j)) \right] \end{pmatrix}, \quad (2.17)$$

where $\log(q_\eta(\psi_j)) = \gamma_j \log(\delta_j) - \log(\Gamma(\gamma_j)) + (\gamma_j - 1) \log(\psi_j) - \delta_j \psi_j$. Substituting this into the FIM stated above, we obtain the following:

$$\mathbf{F}(\boldsymbol{\eta}_\psi) = \begin{pmatrix} \frac{\partial^2}{\partial \gamma_j^2} \log(\Gamma(\gamma_j)) & -\frac{1}{\delta_j} \\ -\frac{1}{\delta_j} & \frac{\gamma_j}{\delta_j} \end{pmatrix} \mathbf{F}^{-1} \mathbf{F}(\boldsymbol{\eta}_\psi)^{-1} = \frac{1}{F^{(1)}(\gamma_j) \frac{\gamma_j}{\delta_j} - \frac{1}{\delta_j^2}} \begin{pmatrix} \frac{\gamma_j}{\delta_j} & \frac{1}{\delta_j} \\ \frac{1}{\delta_j} & F^{(1)}(\gamma_j) \end{pmatrix}. \quad (2.18)$$

By recalling Eq. (2.5), we will be able to compute the gradients with respect to the expectation-parameter:

$$\begin{aligned} \nabla_{m_{\psi_j}^{(1)}} \mathcal{L}_\star &= \frac{1}{F^{(1)}(\gamma_j) - \frac{1}{\delta_j} \gamma_j} \nabla_{\gamma_j} \mathcal{L} + \frac{1}{F^{(1)}(\gamma_j) \frac{\gamma_j}{\delta_j} - \frac{1}{\delta_j^2}} \nabla_{\delta_j} \mathcal{L}, \\ \nabla_{M_{\psi_j}^{(2)}} \mathcal{L}_\star &= \frac{1}{F^{(1)}(\gamma_j) \frac{\gamma_j}{\delta_j} - \frac{1}{\delta_j^2}} \nabla_{\gamma_j} \mathcal{L} + \frac{1}{\frac{\gamma_j}{\delta_j} - \frac{1}{\delta_j^2 F^{(1)}(\gamma_j)}} \nabla_{\delta_j} \mathcal{L}. \end{aligned} \quad (2.19)$$

We rewrite Eq. (2.6) in terms of δ_j

$$\delta_{j,(t+1)} = \delta_{j,(t)} + \beta_{(t)} \cdot \left(\frac{1}{F^{(1)}(\gamma_{j,(t)}) \frac{\gamma_{j,(t)}}{\delta_{j,(t)}} - \frac{1}{\delta_{j,(t)}^2}} \nabla_{\gamma_j} \mathcal{L}_{(t)} + \frac{1}{\frac{\gamma_{j,(t)}}{\delta_{j,(t)}} - \frac{1}{\delta_{j,(t)}^2 F^{(1)}(\gamma_{j,(t)})}} \nabla_{\delta_j} \mathcal{L}_{(t)} \right) \quad (2.20)$$

and in terms of γ_j

$$\gamma_{j,(t+1)} = \gamma_{j,(t)} + \beta_{(t)} \cdot \left(\frac{1}{F^{(1)}(\gamma_{j,(t)}) - \frac{1}{\delta_{j,(t)}} \gamma_{j,(t)}} \nabla_{\gamma_j} \mathcal{L}_{(t)} + \frac{1}{F^{(1)}(\gamma_{j,(t)}) \frac{\gamma_{j,(t)}}{\delta_{j,(t)}} - \frac{1}{\delta_{j,(t)}^2}} \nabla_{\delta_j} \mathcal{L}_{(t)} \right). \quad (2.21)$$

Thus, Eq. (2.20) and (2.21) then describe the natural-gradient updates of the Gamma surrogate for each iteration (t).

2.3 Variational Online-Newton

In this method we firstly (per-sample) factorize the likelihood part in ELBO \mathcal{L} and take its negative logarithm form

$$f(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N f_n(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^N \log p(\mathcal{D}_n | \boldsymbol{\theta}). \quad (2.22)$$

We also prepare the minibatch stochastic-gradient estimates \mathcal{M} of the function $f(\boldsymbol{\theta})$, where the samples M are chosen uniformly at random:

$$\hat{\mathbf{g}}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{n \in \mathcal{M}} \nabla_{\boldsymbol{\theta}} f_n(\boldsymbol{\theta}) \quad (2.23)$$

and eventually choose the model parameters prior $p(\boldsymbol{\theta})$ as the Gaussian with zero mean and the unit covariance matrix controlled by the known precision hyperparameter ψ and the surrogate posterior also as the Gaussian parameterized by its natural parameters:

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta} | \mathbf{0}, \frac{1}{\psi} \mathbf{I}\right), \quad q(\boldsymbol{\theta} | \boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}). \quad (2.24)$$

After this preparation, we are able to derive the Variational Online-Newton (VON) method originally proposed in (and according to) [22], which should facilitate an efficient application of the backpropagation to computing the gradients and Hessians in the natural-gradient updates.

Note. Since some derivations are too long, they will be defended in Appendix A for better readability and orientation in the text.

The VON method slightly modify Eq. (2.12) and Eq. (2.13). Therefore, we need to rewrite Eq. (2.2) to our terms

$$\mathcal{L}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) = \mathbb{E}_q[-Nf(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta} | \boldsymbol{\eta})], \quad (2.25)$$

and express the gradients of the expectation of $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ in terms of the gradient and Hessian of $f(\boldsymbol{\theta})$, to which we use the knowledge from [35]:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{\boldsymbol{\theta}}} \mathbb{E}_q[f(\boldsymbol{\theta})] &= \mathbb{E}_q[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})] = \mathbb{E}_q[\mathbf{g}(\boldsymbol{\theta})], \\ \nabla_{\boldsymbol{\Sigma}_{\boldsymbol{\theta}}} \mathbb{E}_q[f(\boldsymbol{\theta})] &= \frac{1}{2} \mathbb{E}_q[\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})] = \frac{1}{2} \mathbb{E}_q[\mathbf{H}(\boldsymbol{\theta})]. \end{aligned} \quad (2.26)$$

The aforementioned Hessian matrix (or *Hessian*) $\mathbf{H}(\cdot)$ is defined as a square matrix of all second-order partial derivatives of the input function with respect to its variables. Using these, we rewrite the gradients of \mathcal{L} , which are required in the natural-gradient updates in Eq. (2.12) and (2.13):

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{\boldsymbol{\theta}}} \mathcal{L} &= -\mathbb{E}_q[N \mathbf{g}(\boldsymbol{\theta})] - \psi \boldsymbol{\mu}_{\boldsymbol{\theta}}, \\ \nabla_{\boldsymbol{\Sigma}_{\boldsymbol{\theta}}} \mathcal{L} &= \frac{1}{2} \mathbb{E}_q[-N \mathbf{H}(\boldsymbol{\theta})] - \frac{1}{2} \psi \mathbf{I} + \frac{1}{2} \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}. \end{aligned} \quad (2.27)$$

To approximate the expectation over q , the Monte Carlo sampling can be used, where one sample will be defined, as follows

$$\boldsymbol{\theta}_{(t)} \sim \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_{\boldsymbol{\theta},(t)}, \boldsymbol{\Sigma}_{\boldsymbol{\theta},(t)}), \quad (2.28)$$

with which we can rewrite the natural-gradient updates for $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta},(t+1)} &= \boldsymbol{\mu}_{\boldsymbol{\theta},(t)} - \beta_{(t)} \boldsymbol{\Sigma}_{\boldsymbol{\theta},(t+1)} (N \mathbf{g}(\boldsymbol{\theta}_{(t)}) + \psi \boldsymbol{\mu}_{\boldsymbol{\theta},(t)}), \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta},(t+1)}^{-1} &= (1 - \beta_{(t)}) \boldsymbol{\Sigma}_{\boldsymbol{\theta},(t)}^{-1} + \beta_{(t)} [N \mathbf{H}(\boldsymbol{\theta}_{(t)}) + \psi \mathbf{I}]. \end{aligned} \quad (2.29)$$

By defining the transform matrix $\mathbf{S}_{\theta,(t)} = \frac{1}{N} (\boldsymbol{\Sigma}_{\theta,(t)}^{-1} - \psi \mathbf{I}) \rightarrow \boldsymbol{\Sigma}_{\theta,(t)} = [N (\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I})]^{-1}$, we get the following changed updates:

$$\begin{aligned} \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\left(\mathbf{S}_{\theta,(t+1)} + \frac{1}{N} \psi \mathbf{I} \right) \right]^{-1} \left[\mathbf{g}(\boldsymbol{\theta}_{(t)}) + \frac{1}{N} \psi \boldsymbol{\mu}_{\theta,(t)} \right], \\ \mathbf{S}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \mathbf{S}_{\theta,(t)} + \beta_{(t)} \mathbf{H}(\boldsymbol{\theta}_{(t)}). \end{aligned} \quad (2.30)$$

The updates shown in Eq. (2.30) are referred to as Variational Online-Newton method, which was proposed in [22]. It resembles a regularized version of online Newton's method where the scaling matrix is estimated online using the Hessians [47].

2.3.1 Mean-Field Variant of Variational Online Newton

We can also use the mean-field approximation of the updates from Eq. (2.30) by taking only the diagonal in Hessian $\mathbf{H}(\cdot)$ and the diagonal \mathbf{s} in the transform matrix \mathbf{S} :

$$\begin{aligned} \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\frac{\mathbf{g}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\theta,(t)}}{\mathbf{s}_{\theta,(t+1)} + \frac{\psi}{N}} \right], \\ \mathbf{s}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \mathbf{s}_{\theta,(t)} + \beta_{(t)} \text{diag}(\mathbf{H}(\boldsymbol{\theta}_{(t)})), \end{aligned} \quad (2.31)$$

where the two vector division \mathbf{a}/\mathbf{b} is defined element-wisely. Another approximation can be the use of stochastic gradient from Eq. (2.23) and the estimation of the Hessian of $f(\boldsymbol{\theta})$:

$$\begin{aligned} \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\frac{\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\theta,(t)}}{\mathbf{s}_{\theta,(t+1)} + \frac{\psi}{N}} \right], \\ \mathbf{s}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \mathbf{s}_{\theta,(t)} + \beta_{(t)} \text{diag}(\hat{\nabla}_{\theta\theta}^2 f(\boldsymbol{\theta}_{(t)})). \end{aligned} \quad (2.32)$$

Although there is a plenty of modern methods using automatic-differentiation to calculate the Hessian, this computation is very expensive [56]. Furthermore, we can assume that, in general, f can be a non-convex function, which can cause the Hessian and also the variances being negative. That would be incompatible with the definition of the variance and the method might break down.

2.3.2 Reparameterization Trick in Hessian

One of the ways how to avoid an expensive computation of the Hessian is to apply the *reparameterization trick* to the expectation over the Gaussian distribution. The reparameterization trick can be described as a way how to rewrite the expectation, so that the distribution with respect to which we take the gradient is independent of the model parameters $\boldsymbol{\theta}$ [25].

Using the derived identity in Eq. (2.26) and the mean-field variant of VON leads to the approximation of the expectation over the Hessian of the function f :

$$\mathbb{E}_q \left[\nabla_{\theta\theta}^2 f(\boldsymbol{\theta}) \right] = 2 \nabla_{\sigma_\theta^2} \mathbb{E}_q [f(\boldsymbol{\theta})] \approx \hat{\mathbf{g}}(\boldsymbol{\theta}) \frac{\mathbf{e}}{\sigma_\theta}, \quad (2.33)$$

where $\mathbf{e} \sim \mathcal{N}(\mathbf{e}|0, \mathbf{I})$ and $\boldsymbol{\theta} = \boldsymbol{\mu}_\theta + \sigma_\theta \circ \mathbf{e}$.

2.4 Variational Online Gauss-Newton

Another improvement of the VON method (specifically its updates) and especially a way to avoid negative variances is the use of the Generalized Gauss-Newton (GGN) approximation [34]:

$$\nabla_{\theta_j \theta_j}^2 f(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{n \in \mathcal{M}} [\nabla_{\theta_j} f_n(\boldsymbol{\theta})]^2 = \hat{h}_j(\boldsymbol{\theta}), \quad (2.34)$$

where θ_j is the j -th element of the model parameter vector $\boldsymbol{\theta}$. If we set the initial $\sigma_{\theta_j, (t=1)}^2$ as a positive number, it will remain positive throughout the whole iteration cycle. Using this approximation to the update $\mathbf{s}_{\theta_j, (t)}$ in Eq. (2.32), we get

$$\mathbf{s}_{\theta_j, (t+1)} = (1 - \beta_{(t)})\mathbf{s}_{\theta_j, (t)} + \beta_{(t)} \hat{\mathbf{h}}_j(\boldsymbol{\theta}_{(t)}), \quad (2.35)$$

whereas the update for $\boldsymbol{\mu}_\theta$ remains unchanged. We refer to these two updates combined as the Variational Online Gauss-Newton (VOGN) method, which eliminates the constraint on σ_θ^2 . Due to this approximation and the assumption of a positive variance at the beginning of the iteration cycle, it can be assumed, that the VOGN should have an advantage over the VON method. As in the previous case, this method was originally demonstrated in [22].

2.5 Variational RMSprop

A similar approximation of the Hessian of the function $f(\boldsymbol{\theta})$ was proposed in [8], there referred to as the Gradient Magnitude (GM) approximation:

$$\nabla_{\theta_j \theta_j}^2 f(\boldsymbol{\theta}) \approx \left[\frac{1}{M} \sum_{n \in \mathcal{M}} \nabla_{\theta_j} f_n(\boldsymbol{\theta}) \right]^2 = [\hat{g}_j(\boldsymbol{\theta})]^2, \quad (2.36)$$

but the order of operations is reversed here – GGN first squares the gradients and then sums them, while GM sums the gradients first and then squares them. Approximation in Eq. (2.36) is also used in the RMSprop (see also [43]), which is the well-known optimizer in the neural networks optimization. The RMSprop optimizer uses following updates on the model parameters $\boldsymbol{\theta}_{(t)}$:

$$\begin{aligned} \boldsymbol{\theta}_{(t+1)} &= \boldsymbol{\theta}_{(t)} - \alpha_{(t)} \frac{\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)})}{\sqrt{\bar{\mathbf{s}}_{\boldsymbol{\theta}, (t+1)} + \zeta}}, \\ \bar{\mathbf{s}}_{\boldsymbol{\theta}, (t+1)} &= (1 - \beta_{(t)})\bar{\mathbf{s}}_{\boldsymbol{\theta}, (t)} + \beta_{(t)} [\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) \circ \hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)})], \end{aligned} \quad (2.37)$$

where $\bar{\mathbf{s}}_{\boldsymbol{\theta}, (t)}$ is the vector that adapts the learning rate, ζ denotes a small positive scalar added to the denominator in order to avoid unintended dividing by zero and $\alpha_{(t)}$ and $\beta_{(t)}$ are different learning rates.

By revisiting Eq. (2.31), taking square-root over $\mathbf{s}_{\boldsymbol{\theta}, (t+1)}$ and using the GM approximation for the Hessian, we get a very similar update to the RMSprop update called the Variational RMSprop:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}, (t+1)} &= \boldsymbol{\mu}_{\boldsymbol{\theta}, (t)} - \alpha_{(t)} \frac{\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\boldsymbol{\theta}, (t)}}{\sqrt{\bar{\mathbf{s}}_{\boldsymbol{\theta}, (t+1)} + \frac{\psi}{N}}}, \\ \mathbf{s}_{\boldsymbol{\theta}, (t+1)} &= (1 - \beta_{(t)})\mathbf{s}_{\boldsymbol{\theta}, (t)} + \beta_{(t)} [\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) \circ \hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)})], \end{aligned} \quad (2.38)$$

where $\boldsymbol{\theta}_{(t)} \sim \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_{\boldsymbol{\theta}, (t)}, \sigma_{\boldsymbol{\theta}, (t)}^2)$ and $\sigma_{\boldsymbol{\theta}, (t)}^2 = \frac{1}{N(\bar{\mathbf{s}}_{\boldsymbol{\theta}, (t)} + \frac{\psi}{N})}$. This means, that the gradient in Variational RMSprop is computed at the model parameters $\boldsymbol{\theta}_{(t)}$ sampled from the Gaussian distribution. We call it a *weight-perturbation*, because the variance $\sigma_{\boldsymbol{\theta}, (t)}^2$ of perturbation is obtained from the vector $\mathbf{s}_{\boldsymbol{\theta}, (t)}$ that adapts the learning rate [22].

2.6 Variational ADAM

A state-of-the-art optimizer in neural network optimization is widely-used ADAM, a method for efficient stochastic optimization that only requires first-order gradients and computing individual adaptive learning rates for different parameters from the estimates of gradient's first and second moments [24].

Similar to the variational alternative of the RMSprop optimizer, the Variational ADAM (VADAM) can be proposed. VADAM, like Variational RMSprop, was originally introduced in [22] and performs the variational inference. It updates the parameters and the hyperparameters in the model at each step considering them as random variables whose surrogate posterior $q(\boldsymbol{\theta}|\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2)$ given the training data is inferred by variational inference at every training step [54].

The full derivation of VADAM update can be found in the original paper [22]. However, we give only the resulting equations (in combination with [24]) for the update in each iteration t for the parameters known from the Variational RMSprop description, $\boldsymbol{\theta}_{(t)} \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\theta,(t)}, \boldsymbol{\sigma}_{\theta,(t)}^2)$ and $\boldsymbol{\sigma}_{\theta,(t)}^2 = \frac{1}{N(s_{\theta,(t)} + \frac{\psi}{N})}$:

$$\begin{aligned} \mathbf{u}_{\theta,(t+1)} &= \xi_1 \mathbf{u}_{\theta,(t)} + (1 - \xi_1) \left(\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\theta,(t)} \right), \\ \mathbf{s}_{\theta,(t+1)} &= \xi_2 \mathbf{s}_{\theta,(t)} + (1 - \xi_2) [\hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)}) \circ \hat{\mathbf{g}}(\boldsymbol{\theta}_{(t)})], \\ \hat{\mathbf{u}}_{\theta,(t+1)} &= \frac{\mathbf{u}_{\theta,(t+1)}}{1 - \xi_1^t}, \\ \hat{\mathbf{s}}_{\theta,(t+1)} &= \frac{\mathbf{s}_{\theta,(t+1)}}{1 - \xi_2^t}, \\ \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \nu \cdot \frac{\hat{\mathbf{u}}_{\theta,(t+1)}}{\sqrt{\hat{\mathbf{s}}_{\theta,(t+1)} + \frac{\psi}{N}}}, \end{aligned} \tag{2.39}$$

where ν is the learning rate and $\xi_1, \xi_2 \in [0, 1)$ are the hyperparameters controlling the exponential decay.

Note. During the computing bias-corrected first and second moment estimates in Eq. (2.39) the symbols ξ_1^t and ξ_2^t denote ξ_1 and ξ_2 to the power t .

2.7 Variational ADAM with ARD Prior

Within this thesis, we propose a new possible algorithm for finding a sparse parameterization of the model. Let us assume probability model with likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ (Gaussian in the regression task, Bernoulli in the classification task), Gaussian prior on weights and Gamma prior on hyperparameter $\boldsymbol{\psi}$ in following expression

$$p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}, \boldsymbol{\psi}, \omega) \propto p(\mathcal{D}|\boldsymbol{\theta}) \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\psi}^{-1}) \cdot \mathbf{I}) \prod_{j=1}^K \text{Gamma}(\gamma_0, \delta_0). \tag{2.40}$$

Our goal is to approximate the posterior for all latent variables in factor form

$$p(\boldsymbol{\theta}, \boldsymbol{\psi}, \omega|\mathcal{D}) \approx q_\eta(\mathbf{z}) = q_\eta(\boldsymbol{\theta})q_\eta(\boldsymbol{\psi})q_\eta(\omega). \tag{2.41}$$

We use part of the derived equations to update the posterior factors of the hyperparameters in the linear regression from [50]:

$$\begin{aligned} q(\psi_j|\gamma_j, \delta_j) &= \text{Gamma}\left(\gamma_0 + \frac{1}{2}, \delta_0 + \frac{1}{2} \mathbb{E}[\theta_j^2]\right), \quad \forall j = 1, \dots, K, \\ q(\omega|\bar{\gamma}_j, \bar{\delta}_j) &= \text{Gamma}\left(\delta_0 + \frac{K}{2}, \mathbb{E}[(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})]\right), \end{aligned} \tag{2.42}$$

where $\mathbb{E}[\theta_j^2] = \mu_{\theta_j}^2 + \sigma_{\theta_j}^2$ and $\mathbb{E}[\psi_j] = \frac{\gamma_j}{\delta_j}$.

Note. The posterior factor for the hyperparameter ω is mentioned here only in the context of Section 1.7, where it was used. It will not appear in the following experiments as it will be taken as a known constant, but it can be estimated together with all latent variables in the case, where it is unknown according to Eq. (2.42).

Our proposed algorithm combines model negative log-likelihood described in Eq. (2.22) with the ARD prior in negative logarithm form:

$$f_{\text{new}}(\boldsymbol{\theta}) = \underbrace{-\frac{1}{N} \sum_{n=1}^N \log p(\mathcal{D}_n | \boldsymbol{\theta})}_{f(\boldsymbol{\theta})} - \sum_{j=1}^K \frac{1}{2} \theta_j^2 \psi_j. \quad (2.43)$$

This changes the meaning of the function $f(\boldsymbol{\theta})$ from Eq. (2.22) entering the optimization. The resulting proposed algorithm using the variational ADAM with the ARD prior can then be written, as follows in Algorithm 1.

Algorithm 1 VADAM with the ARD prior, updates for the posterior parameters in one iteration (t).

- 1: **Initialize prior parameters** γ_0, δ_0 , learning rates in ADAM, (ω).
 - 2: **Calculate in each dimension** $\gamma_{j(t)}, \delta_{j(t)}, (\overline{\gamma_{j(t)}}, \overline{\delta_{j(t)}})$.
 - 3: $\psi_{j(t)} \leftarrow \frac{\gamma_{j(t)}}{\delta_{j(t)}}, (\omega_{(t)} \leftarrow \frac{\overline{\gamma_{j(t)}}}{\overline{\delta_{j(t)}}})$.
 - 4: **Do step** described in Eq. (2.39) with $f_{\text{new}}(\boldsymbol{\theta})$.
 - 5: **Update** $\gamma_{j(t)}, \delta_{j(t)} (\overline{\gamma_{j(t)}}, \overline{\delta_{j(t)}})$ according to Eq. (2.42).
 - 6: $(t + 1) \leftarrow (t)$.
-

This algorithm is designed to search for possible sparse solutions using prior knowledge. By doing so, we insert new information into the model, in the form of the shrinkage prior, that we want in particular the sparse solutions from the set of all possible solutions.

Chapter 3

Experiments

3.1 Sparse Linear Regression

In the first simulation scenario, we set up a simple linear regression problem on which we want to observe the behavior of different methods for achieving a sparse parameterization. We use artificially generated (synthetic) dataset given by a data matrix \mathbf{X} , where the entries are sampled from the Gaussian distribution, and the response variable, $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}_{\text{true}} + \mathbf{e}$, with a fixed known parameter vector (groundtruth) $\boldsymbol{\theta}_{\text{true}}$ and the Gaussian noise \mathbf{e} of the mean $\boldsymbol{\mu}_e$ and variance σ_e^2 . Keep in mind that one realization of the data matrix (using a fixed seed) is equivalent to one realization of the random matrix with the Gaussian distribution.

In this way, we can then monitor whether the estimates delivered by various methods actually converge to the exact values obtained from an analytically solvable estimator. That is, in this simple linear regression problem, there surely exists the analytically tractable estimate of $\boldsymbol{\theta}_{\text{true}}$ given by

$$\widehat{\boldsymbol{\theta}}_{\text{true}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \quad (3.1)$$

With $\boldsymbol{\theta}_{\text{true}} = (2.9, 1.1, 0.02, 0.05, 10.0, 7.2, 0.06, 9.1, 0.001, 0.2, 0.76)^T \in \mathbb{R}^{11}$, $(\mathbf{1}, \mathbf{X}) \in \mathbb{R}^{100 \times 10+1}$ and Gaussian noise $e_n \sim \mathcal{N}(0, \sigma_e^2)$ with $\sigma_e^2 = 0.5$ we generate response variable $\mathbf{y} \in \mathbb{R}^{100}$. Our model will then look like this (in matrix form):

$$\begin{pmatrix} y_1 \\ y_n \\ \vdots \\ y_{100} \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,10} \\ 1 & x_{2,1} & \dots & x_{2,10} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{100,1} & \dots & x_{100,10} \end{pmatrix} \cdot \begin{pmatrix} \theta_{\text{true},1} \\ \theta_{\text{true},2} \\ \vdots \\ \theta_{\text{true},11} \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{100} \end{pmatrix} \quad (3.2)$$

We shuffle the data, split dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$ into the train and test sets with the ratio 0.8 and choose the batchsize 16. Eventually, we run several methods with various settings.

3.1.1 Model Architecture

In the neural networks paradigm, the model in Eq. (3.2) can be interpreted as a neural network with no hidden layer. The architecture of this simple neural network consists of an input layer (\mathbb{R}^{11}) and an output layer with an identity activation function. We choose total of 11 trainable parameters in this network, showing its graphical representation in Fig. 3.1.

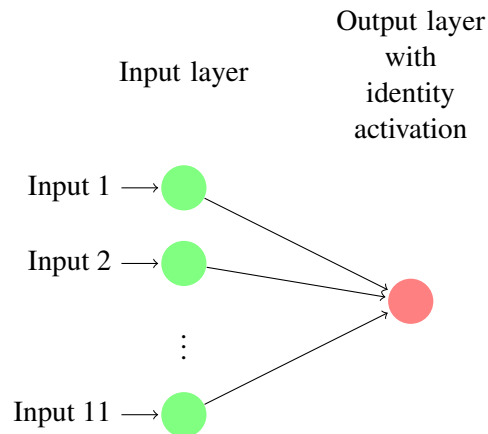


Figure 3.1: Neural network architecture for the linear regression task.

3.1.2 Maximum Likelihood Estimation in Linear Regression

In the first experiment, we will try to tune the full model, while closely monitoring the model error (*loss*) in the form of the *mean-squared error* (MSE) and the parameter convergence across epochs. We choose the ADAM optimizer with the hyperparameters $\nu = 0.001$, $\xi_1 = 0.9$ and $\xi_2 = 0.999$ to fit the model's parameters.

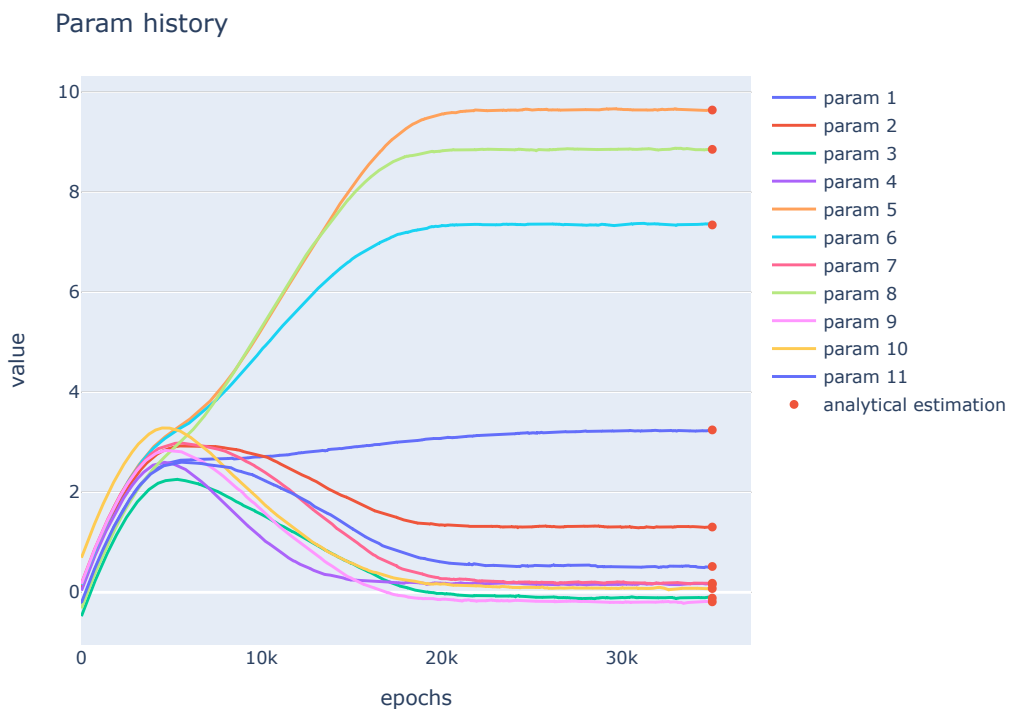


Figure 3.2: Linear regression parameters during training.

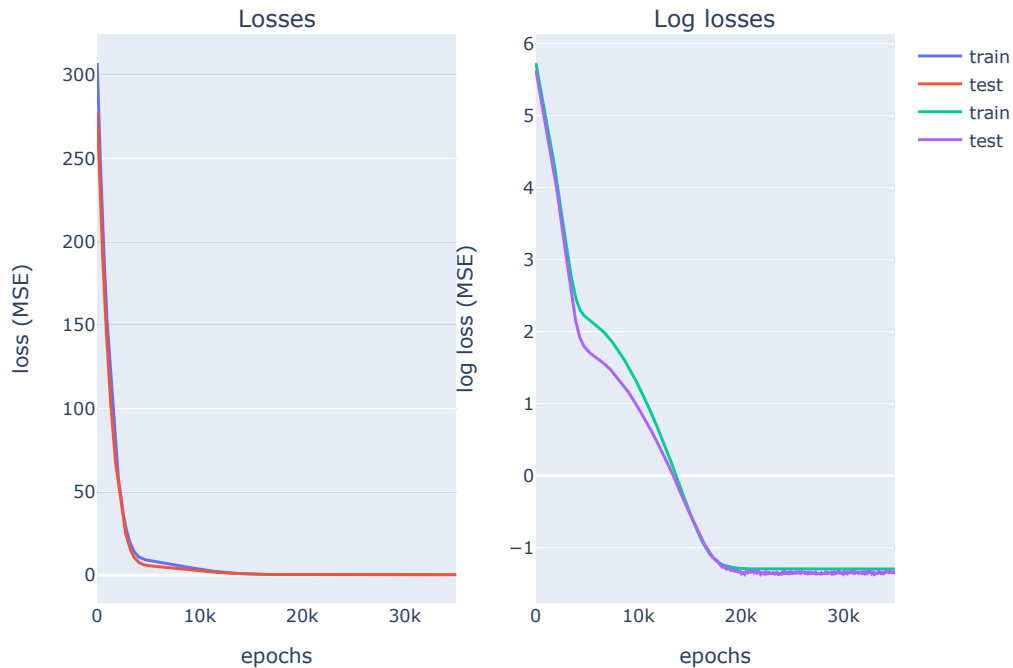


Figure 3.3: Values of loss function on train and test sets in linear regression.

In Fig. 3.2, we can see that the classical MLE method has converged to an analytical solution from Eq. (3.1). The loss function values in Fig. 3.3 have reached their minimum. This is no surprise – the response variable was generated according to the known parameter θ_{true} . The training of the full model is presented here mainly for interest, since we only have a point estimate of the parameters and did not force any sparsity. Alternatively, it would have to be at everyone’s opinion how they would like to assess the area around the point estimates. The intention was to have components close to zero in the θ_{true} so we could either control the sparsity at our discretion or use a method that could automatically zero them. Therefore, we add the Laplace prior on the model parameters to help us achieve a sparser parameterization.

3.1.3 L_1 Penalization in Linear Regression

Adding the Laplace prior from Eq. (1.15) with the zero mean on the model parameters to the Gaussian likelihood creates a new optimization task. The new loss function to minimize is then in the same form as in Eq. (1.56) with optional hyperparameter λ , which is a tuning knob we would like to set right to achieve a sparse parameterization.

Thus, we first analyze the relationship of how the values of the trained parameters change as a function of λ to estimate its reasonable range. This is mainly for an initial look, when most parameters start to converge to zero by estimation. Initial setting of the optimizer, the batchsize, and the data split, will be the same as those in the MLE estimation. We choose a rough λ -grid in the range $\lambda \in (0.0, 1.0, 2.0, \dots, 40.0)$. We always take one particular λ , train the parameters and record their values to plot the convergence history.

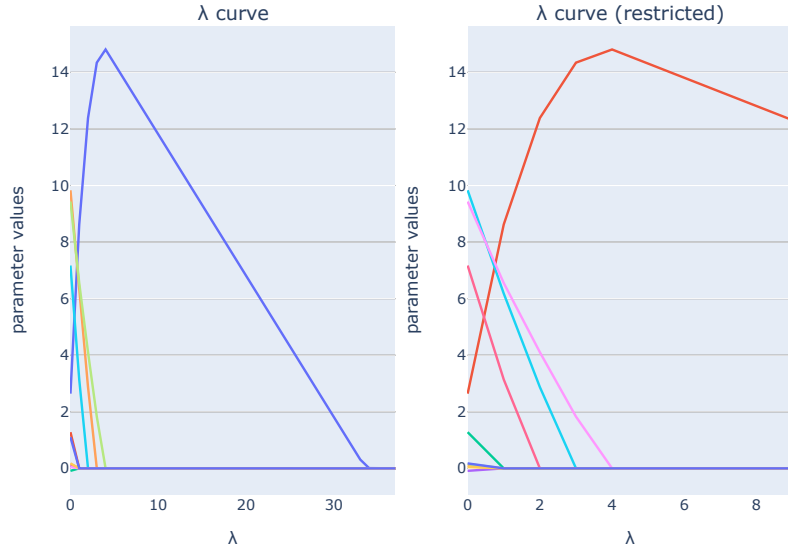


Figure 3.4: Relationship between λ and parameters values in $\lambda \in (0.0, 1.0, 2.0, \dots, 40.0)$ grid. On the right with zoomed range of $\lambda \in (0.0, 1.0, 2.0, \dots, 9.0)$.

Fig. 3.4 shows us which values of the hyperparameter λ is required to zero the parameters of the model. A closer look reveals that except for one parameter in the model, it is possible to zero out the vast majority of the other parameters using $\lambda \in (0.0, 4.0)$. It is clear that the most change happens in this restricted interval, so we make the λ -grid smoother with a smaller step, $\lambda \in (0.0, 0.1, \dots, 4.0)$, and to find a certain trade-off between the model *error* and sparse parameterization, we plot the so-called Pareto frontier, as described in Algorithm 2.

Algorithm 2 Pareto frontier.

- 1: Set the model, loss, optimizer, λ .
 - 2: Train the model and save the results of the last epoch.
 - 3: Check if any component of the trained parameter vector belongs to an ε -neighborhood of zero.
 - 4: Compute the loss value on test data with selected parameters.
 - 5: Plot loss value on test data and number of components, which do not belong to the ε -neighborhood.
 - 6: Repeat step 1 to 5 with another value of λ .
 - 7: Highlight the set of models with the lowest error given λ .
-

Pareto frontier is a good graphical tool for comparing models in the case of observed sparsity. Ideally, we would like one of the axes of the Pareto frontier to be the model *error* and the other the L_0 norm of trained parameter (the number of non-zero parameters). However, due to numerical calculations, the strict L_0 norm does not give good results, so it needs to be generalized. We define its generalized form, $L_{0,\varepsilon}$, as follows:

$$\|\theta\|_{0,\varepsilon} = \#\{\theta_k \notin (0 - \varepsilon, 0 + \varepsilon) | k = 1, \dots, K\}, \quad (3.3)$$

where the symbol $\#$ denotes total number of elements falling in the ε -neighborhood and ε is the optional tolerance parameter. The parameter ε can be set manually according to personal preference, as to how much around the zero the user considers the parameter to be accepted as zero. In the case of variational methods (as shown in the following sections), this tolerance parameter, ε , can be replaced by the variance estimates already included in those methods.

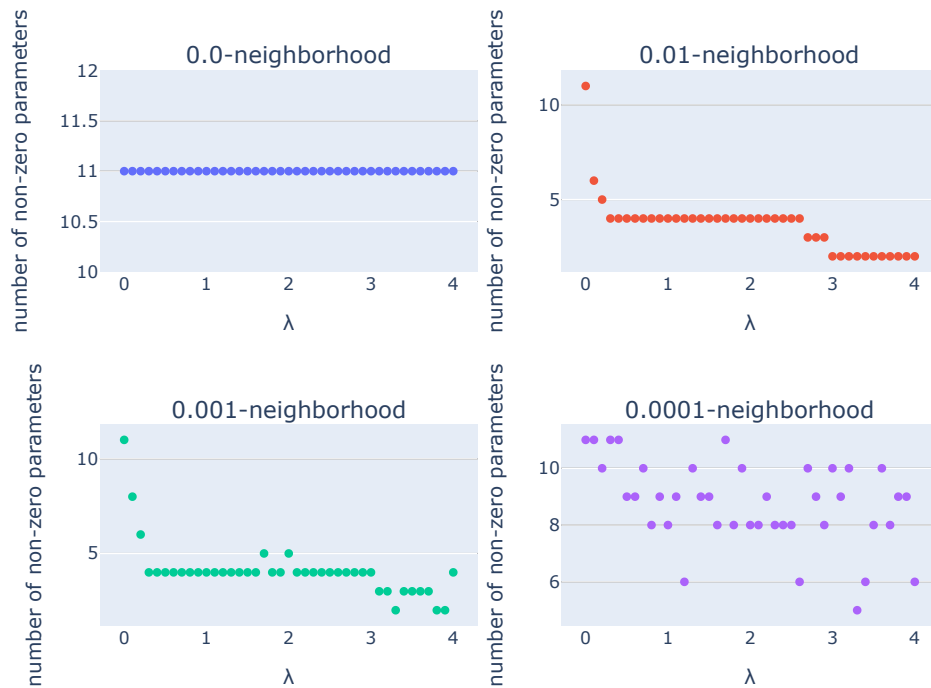


Figure 3.5: The relationship between λ and non-zero parameters depending on the ϵ -neighborhood given by $\epsilon \in \{0.0, 0.01, 0.001, 0.0001\}$.

With each minor adjustment of the tolerance parameter ϵ in Fig. 3.5, we get a more precise insight into at what λ -value and what ϵ -preference, the trained parameters start to zero out (leading to a sparse parameterization of the model). Although sparsity can be achieved in this way, it is still necessary to evaluate whether any particular choice of λ is adequate compared to the *error* of the model on the test data. Therefore, we select $\epsilon = 0.001$, and, according to Algorithm 2, the Pareto frontier of the linear regression using L_1 penalization will be plotted.

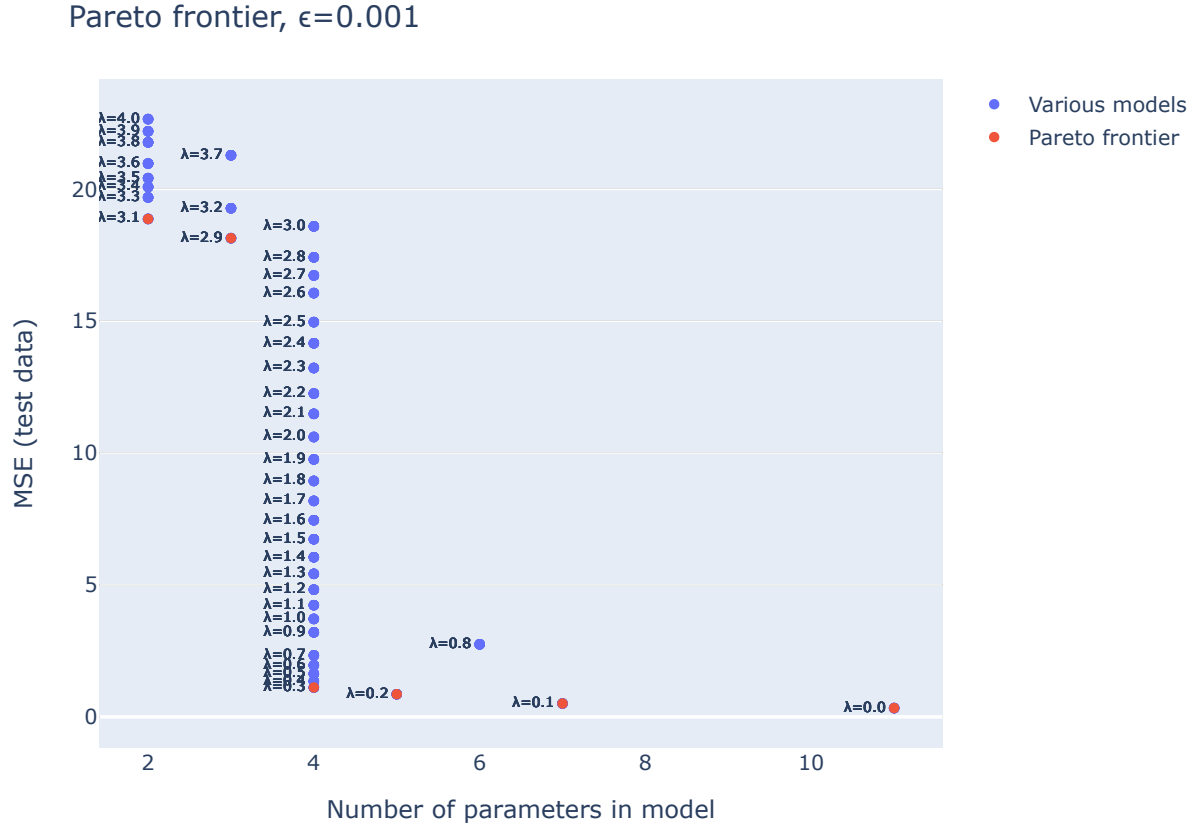


Figure 3.6: The Pareto frontier of linear regression using the L_1 -penalization for $\lambda \in (0.0, 0.1, \dots, 4.0)$ grid and tolerance parameter $\epsilon = 0.001$.

Pareto frontier shown in Fig. 3.6 is a good graphical tool to select a preferable model. It is up to the subjective view of the reader whether they prefer a model with more parameters but a less *error*, or fewer parameters but still an acceptable *error*. The reader should keep in mind, however, that it is still necessary to determine in advance the threshold at which the parameter can be taken as zero (the tolerance parameter ϵ) – which is, again, a matter of subjective discretion. For example, choosing $\lambda = 0.3$ zeroes out 7 parameters, but the *error* on the test data is still comparable to the full model with 11 parameters. However, removing another parameter from the model would increase the model *error* by almost 20 times.

3.1.4 Variational ADAM in Linear Regression

A possible disadvantage of the previous way to force sparsity is the manual setting of the tolerance parameter, ϵ . The following method (divided into two cases) uses the findings from Section 2.6 and Section 2.7 to perform the variational inference and obtain the Gaussian posterior for θ_{true} . Thus, at the end of learning, we get not only the estimates of the mean values μ_θ of the parameters θ_{true} , but also their variances σ_θ^2 , which allows us to compute the standard deviations $\widehat{\sigma}_\theta$.

We first check the convergence of the VADAM optimizer without the prior to see if the mean values converge to the analytical solution in Eq. (3.1). Similarly to the classic ADAM optimizer, the hyperparameters $\nu = 0.001$, $\xi_1 = 0.9$ and $\xi_2 = 0.999$ are chosen for this method.

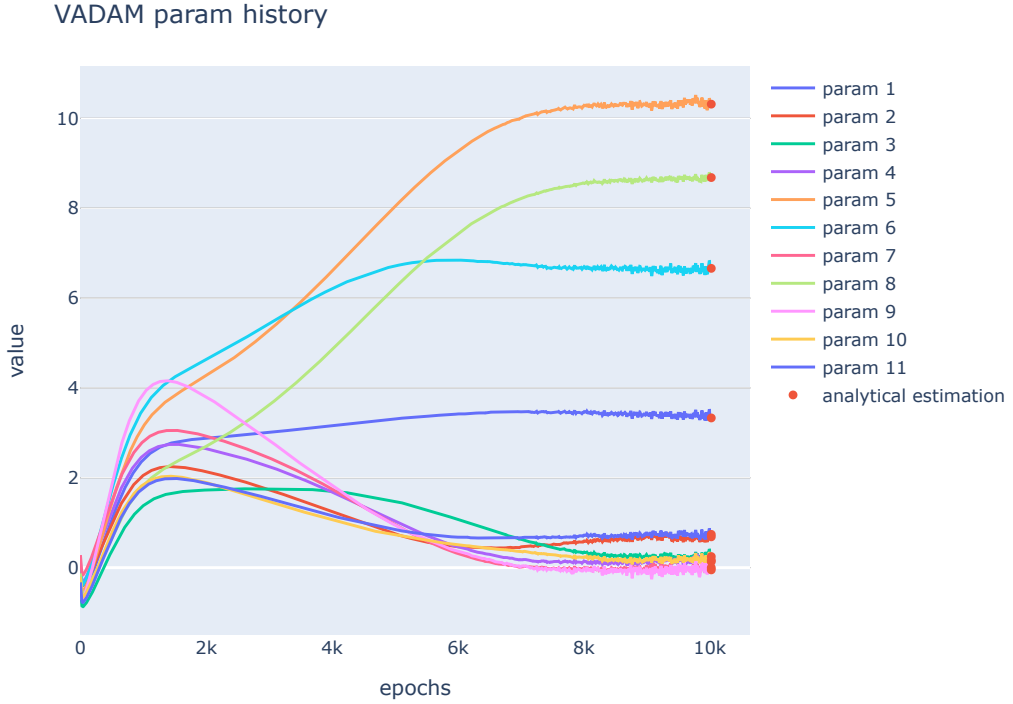


Figure 3.7: The parameters of the linear regression model during the training with the VADAM optimizer without any prior.

Similarly to the MLE (Fig. 3.2), this variational method converged to an analytical solution in Eq. (3.1), as shown in Fig. 3.7. There are minor oscillations around the mean values in the last few epochs of training. This is due to the corrections of the first and second moments in the VADAM optimizer. Unlike the MLE or L_1 regularization at the end of training, we have the inferred posterior estimate of the parameters. Fig. 3.7 can also be taken as a kind of check that the method does what it is supposed to do and that the variational inference was successful.

The estimated standard deviations are used to declare whether a parameter is irrelevant to the model and therefore can be zeroed out. To evaluate the number of relevant non-zero parameters, we introduce a similar generalized form of the L_0 norm for variational methods as in Eq. (3.3), i.e.,

$$\|\theta\|_{0, \widehat{\sigma}_\theta, d} = \#\{\mu_{\theta_k} \notin (0 - d \cdot \widehat{\sigma}_{\theta_k}, 0 + d \cdot \widehat{\sigma}_{\theta_k}) | k = 1, \dots, K \wedge d \in \mathbb{R}^+\}, \quad (3.4)$$

where one can choose as a high value of d as preferred. In this variational case, we make a grid for the parameter d to observe its impact on the zeroing-out behaviour and the test loss. Using the variance estimates, the corresponding standard deviations are calculated and then put into the generalized L_0 norm in Eq. (3.4) for mean values of the last epoch. First, we show an example with the choice $d = 2$ in the last-epoch's parameters from Fig. 3.7, and, eventually, by selecting a particular d -value in the grid of $d \in (0.0, 0.5, 1.0, \dots, 10.0)$, applying the generalized L_0 norm and computing the loss in the form of the MSE on the test data, the Pareto frontier will be plotted.

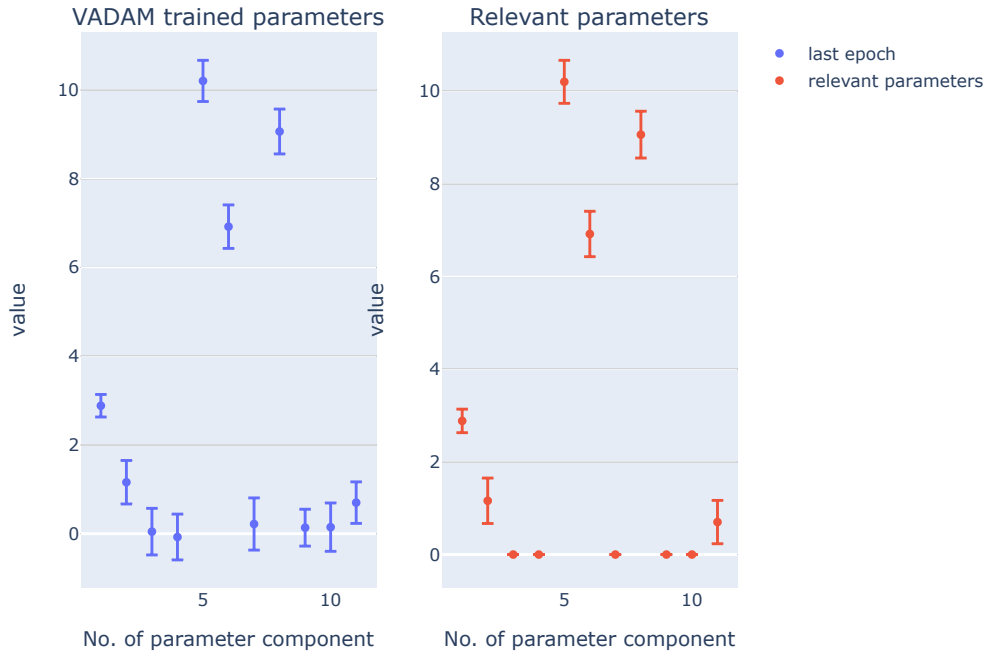


Figure 3.8: The linear regression parameters of the VADAM optimizer from the last epoch with the means and the corresponding standard deviations for $d = 2$, $\|\theta\|_{0,\hat{\sigma}_\theta,2} = 6$.

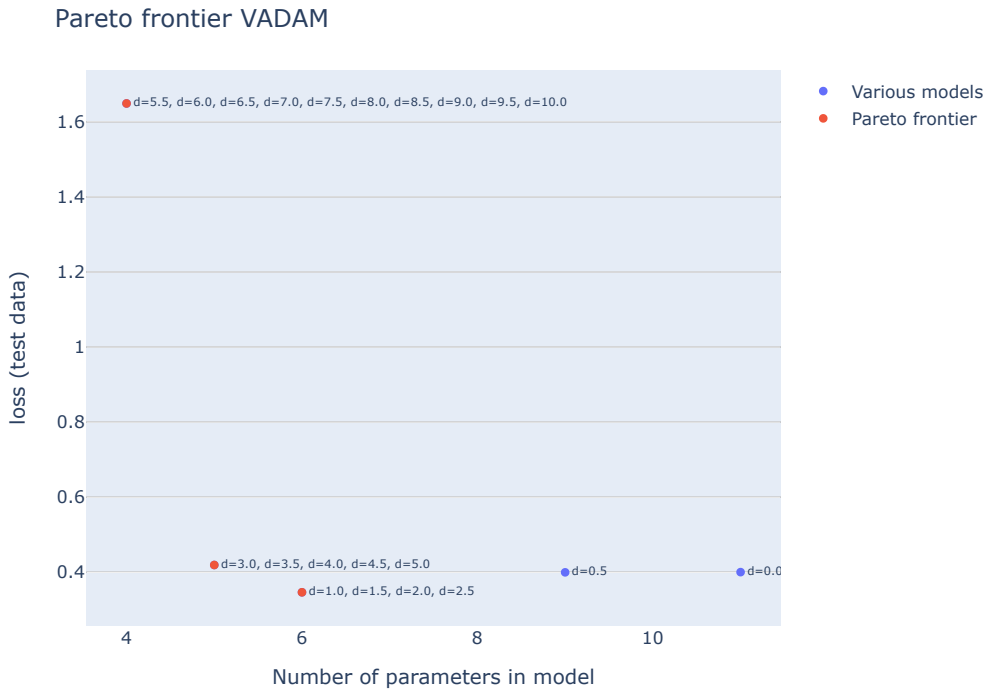


Figure 3.9: The Pareto frontier of the linear regression using the VADAM optimizer for $d \in (0.0, 0.5, 1.0, \dots, 10.0)$.

The Pareto frontier in Fig. 3.9 shows the model *error* on the test data including the relevant number of parameters estimated by the VADAM optimizer for a particular choice of the number of standard deviations d . For the aforementioned setting, $d = 2$, we get Fig. 3.8, which shows us the parameter value of each component in a sparse parameterization with a minimal loss according to Fig. 3.9.

3.1.5 VADAM with ARD Prior in Linear Regression

We now demonstrate our proposed algorithm, as described in Section 2.7, to perform an alternative variational inference to the VADAM optimizer without any prior from above. With the same experimental setup as with the MLE, L_1 regularization and the VADAM optimizer without prior, we check the convergence of the method and the ability to find the sparse solution we want.

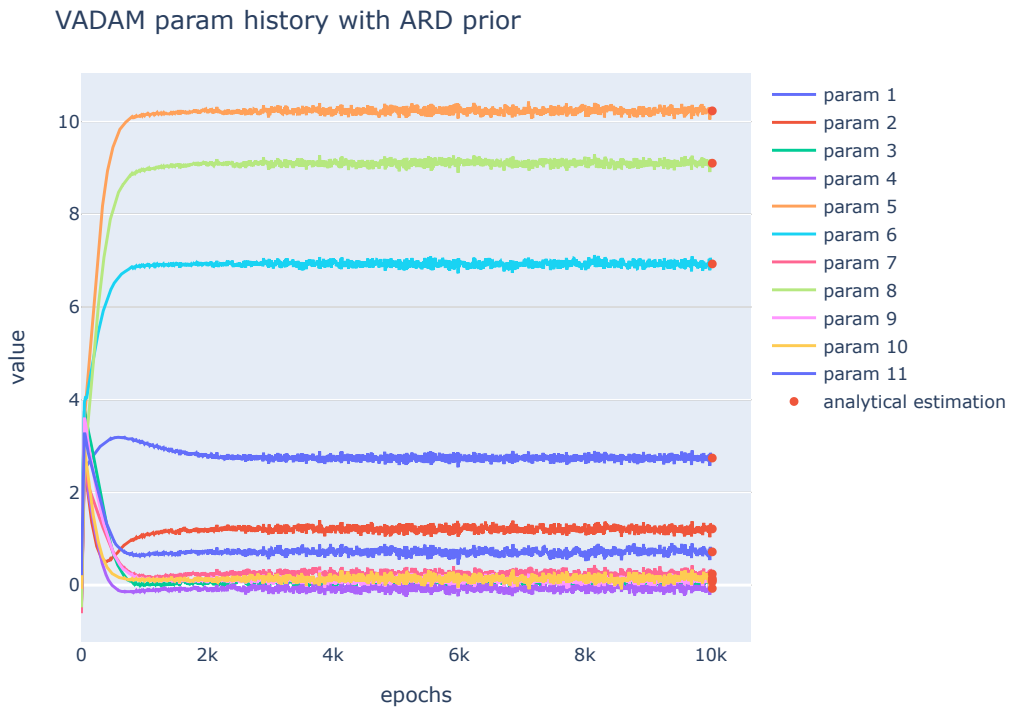


Figure 3.10: The parameters of the linear regression during the training using the VADAM optimizer with the ARD prior.

Fig. 3.10 shows that the method has converged to a sparse solution that matches the analytical estimate in Eq. (3.1). Compared to Fig. 3.7, it is clear that adding the ARD prior to the method speeded up the convergence considerably, while still finding out the correct estimate. The hyperparameter ψ is estimated automatically (see Algorithm 1).

As in previous case of the VADAM optimizer without the prior, we show an example with the choice $d = 2$ for the parameters from the last epoch in Fig. 3.10 and then plot the Pareto frontier.

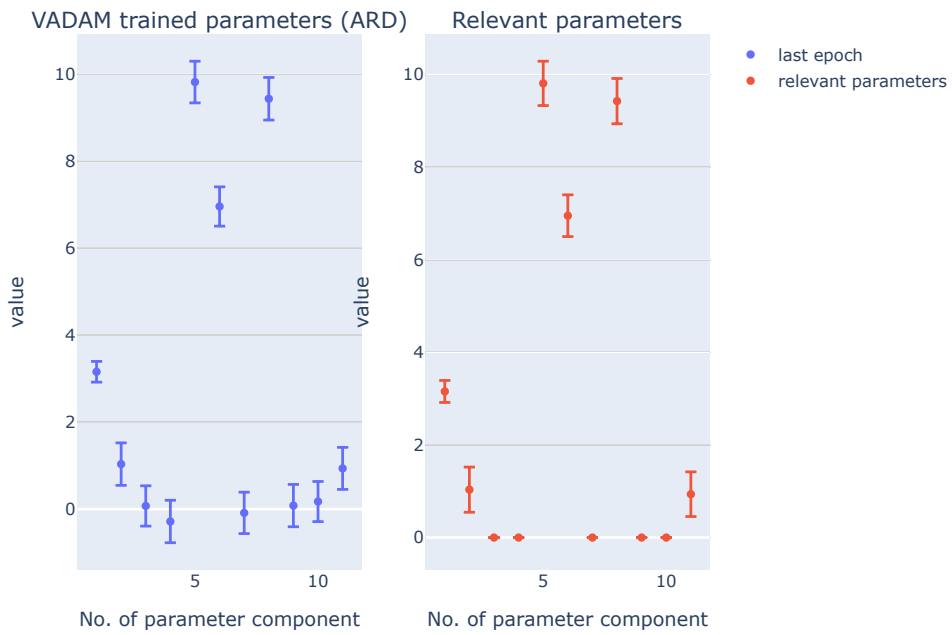


Figure 3.11: The linear regression using the VADAM optimizer with the ARD prior for the parameters from the last epoch, with the mean values and the corresponding standard deviations for the choice $d = 2$, $\|\theta\|_{0, \hat{\sigma}_\theta, 2} = 6$.

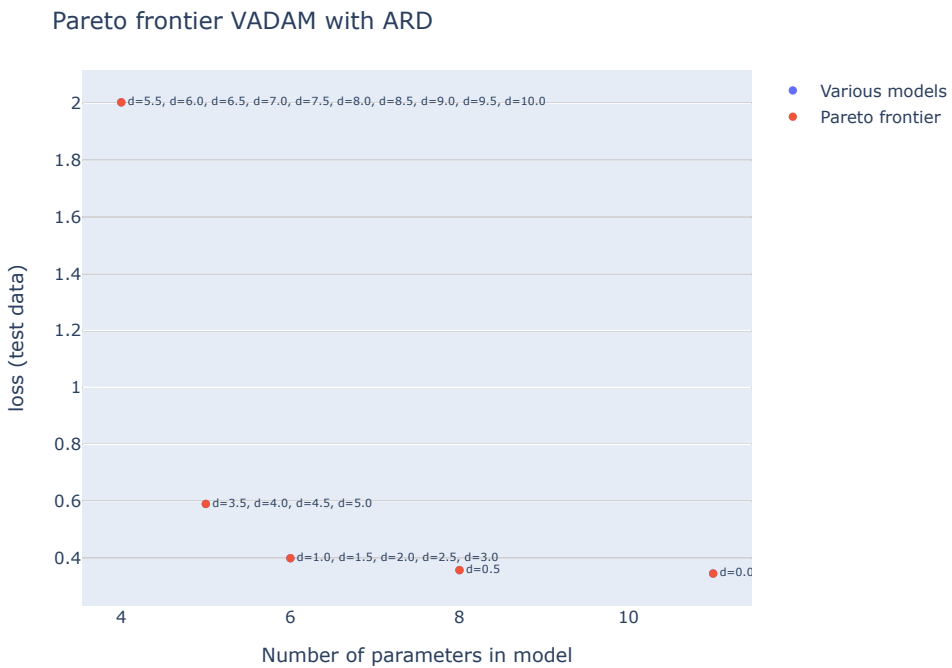


Figure 3.12: The Pareto frontier of the linear regression using the VADAM optimizer with the ARD prior for $d \in (0.0, 0.5, 1.0, \dots, 10.0)$.

The Pareto frontier plotted in Fig. 3.12 shows only the set of the lowest loss value models given an appropriate d . The example with the choice $d = 2$ and the corresponding selected relevant components of the estimated parameters is shown in Fig. 3.11.

3.1.6 Evaluation of Methods

We would like to compare and evaluate the methods used to obtain the sparse parameterization in this linear regression model illustrated in Fig. 3.1, in particular, the L_1 penalization and our proposed method of the VADAM optimizer with the ARD prior. We take the Pareto frontiers from both cases plotted in Fig. 3.6 and Fig. 3.12, plot them into one figure and compare their curves. If one curve is lower than the other in a certain region of this figure, it can be said that in terms of finding a sparse parameterization of a certain number of parameters, and evaluating the model *error* on the test data, one method performs better than the other.

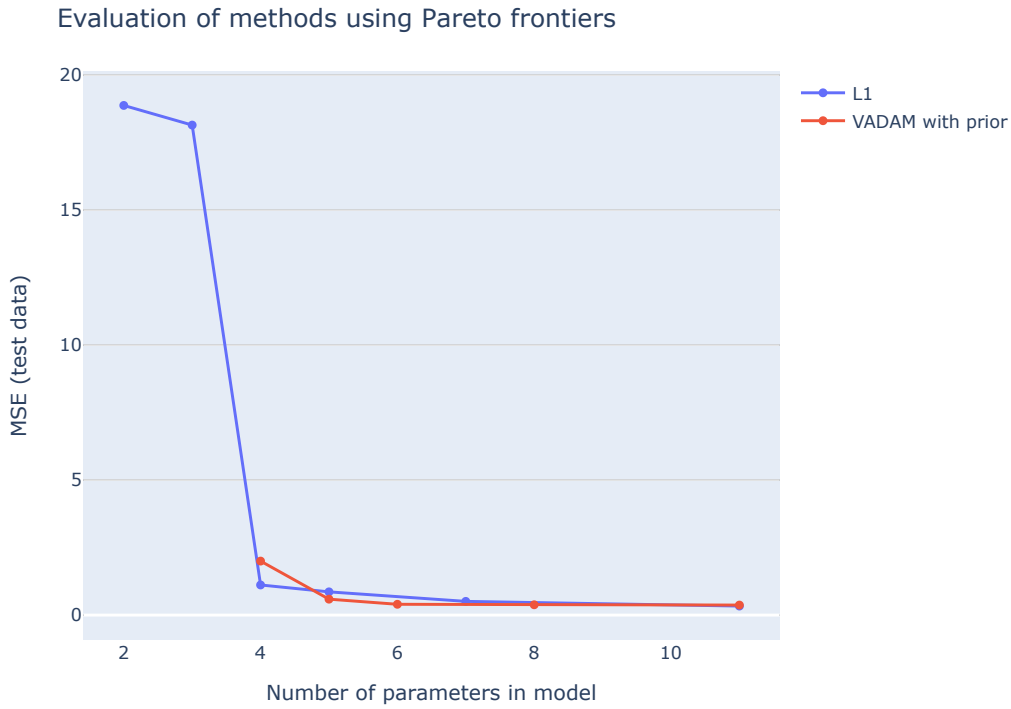


Figure 3.13: The evaluation of the methods to estimate a sparse parameterization in the linear regression experiment.

The ability to find a sparser parameterization is satisfied by both methods, as shown in Fig. 3.13. Our proposed method performs better in finding sparse parameterization with 5, 6 or 8 parameters. In the case of a 4-parameter parameterization, the L_1 penalization reached a smaller *error* on the test data. It can be stated that for such an experimental setup, both methods performed well and both revealed the number of relevant elements in θ_{true} . We can state that with a minimal increase in the *error* on the test data, up to 7 parameters can be removed from the model in Eq. (3.2) using either the L_1 regularization or the VADAM optimizer with the ARD prior.

3.2 Sparse Logistic Regression

In the next experiment, we are concerned with sparsifying a logistic regression model when applied to the classification of the well-known Iris dataset [51] from UCI Machine Learning Repository. The dataset contains 150 observations of 4 variables describing the length and width of the sepals and petals with a three-class response variable encoding the affiliation to a given plant species, i.e., $\mathbf{X} \in \mathbb{R}^{150 \times 4}$ and $\mathbf{y} \in \{\text{setosa}, \text{versicolor}, \text{virginica}\}^{150}$. The data are split with the ratio 0.8 to the train and the test sets. The batchsize is chosen to be 4. The dataset is labelled, hence this is clearly a supervised learning task. This experiment relies on description from Section 1.8.

3.2.1 Model Architecture

For this task, the architecture of the neural network is as follows: the input layer of dimension 4, single hidden layer containing eight neurons and the ReLU activation function, and the output layer of dimension 3 with a softmax output activation function. There is a total of 67 trainable parameters. This architecture is shown in Fig. 3.14.

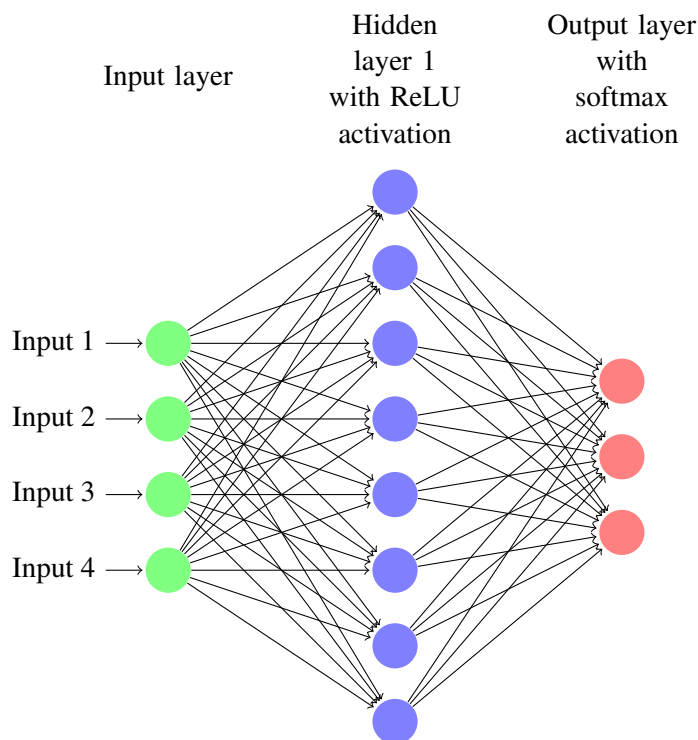


Figure 3.14: The neural network architecture for the logistic regression applied to the Iris dataset.

One can argue why to construct such a complex network just for the simple Iris dataset, which not only contains only 150 observations, but also the classes are well separable (the data are depicted in [40]). This second experiment is designed to use the methods from the first one with the linear regression and try to prune this neural network to obtain a sparse parameterization of only those parameters that are relevant to this problem, assuming the smallest possible increase in the model *error*.

3.2.2 Maximum Likelihood Estimation in Logistic Regression

The first run of this experiment was carried out without any prior or sparsity assumptions. We used, again, the ADAM optimizer with the hyperparameters $\nu = 0.001$, $\xi_1 = 0.9$ and $\xi_2 = 0.999$. We monitored the classification accuracy of the model and the value of the *binary cross-entropy* loss function from Eq. (1.55). These values calculated at each epoch during the training are plotted in Fig. 3.15.



Figure 3.15: Values of the binary cross-entropy loss function and the classification accuracy on the train and the test sets.

Already during the first hundreds of epochs, there is a sharp increase in model accuracy towards 95%. One can also observe oscillating values of the test loss and the test accuracy at the end of the training. It is clear that this model was overfitted and, therefore, it is necessary to prune it.

3.2.3 L_1 Penalization in Logistic Regression

We add the Laplace prior with the zero mean on the model parameters to the Bernoulli likelihood and obtain the loss function described in Eq. (1.56) with an optional hyperparameter λ . We will follow the same procedure as in the linear regression task. First, we plot how the parameter values depend on λ , and then, by selecting four different tolerance parameters ε , a dependency of the corresponding $L_{0,\varepsilon}$ from Eq. (3.3) on λ . Since this is a logistic regression task whose parameters are usually small, a smooth grid with a small step-size should be chosen. Therefore, we choose $\lambda \in (0.0, 0.0005, \dots, 0.04)$ grid.

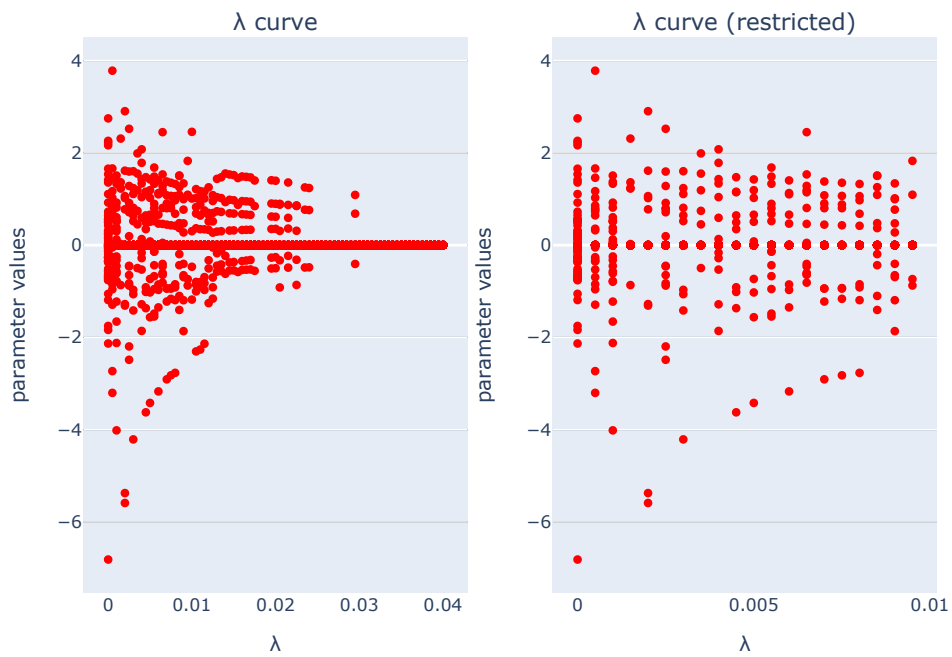


Figure 3.16: The relationship between λ and parameter values for $\lambda \in (0.0, 0.0005, \dots, 0.04)$ grid (left) and $\lambda \in (0.0, 0.0005, \dots, 0.01)$ grid (right).

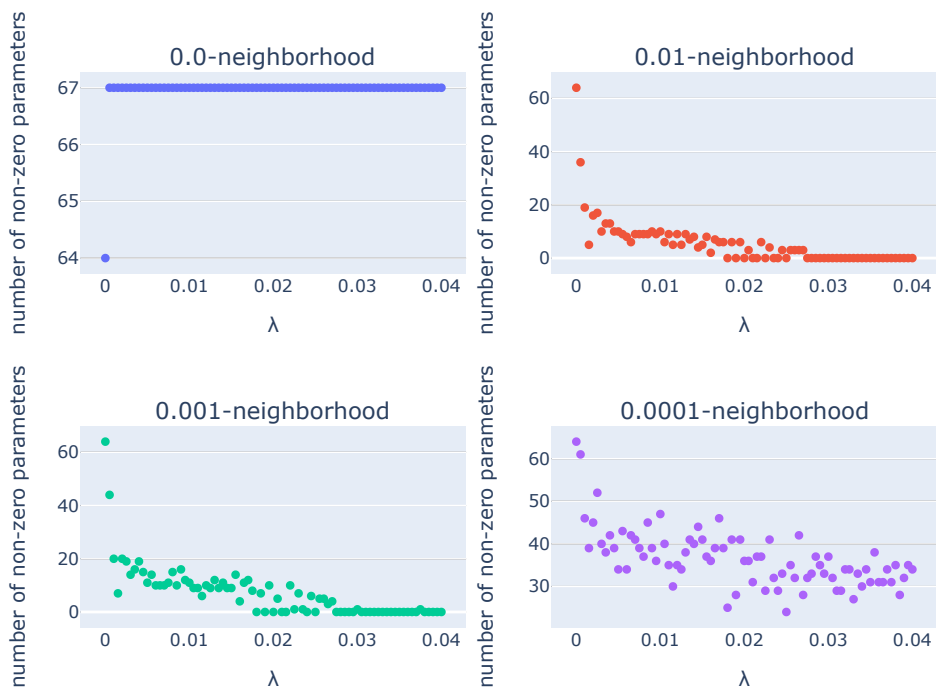


Figure 3.17: The relationship between λ and the number of the non-zero parameters depending on the ε -neighborhood for $\varepsilon \in \{0.0, 0.01, 0.001, 0.0001\}$.

Fig. 3.16 illustrates that many parameters in the model have a really small value (even if the grid was chosen very smoothly) and the trend decreases to zero. A slight increase in λ immediately penalizes the parameters and quickly tightens them to zero, which is confirmed in Fig. 3.17 by the fact that even small choices of the tolerance parameter ε do not hold the majority of the parameters of the model. Now, we select $\varepsilon = 0.001$ and plot the Pareto frontier according to Algorithm 2.

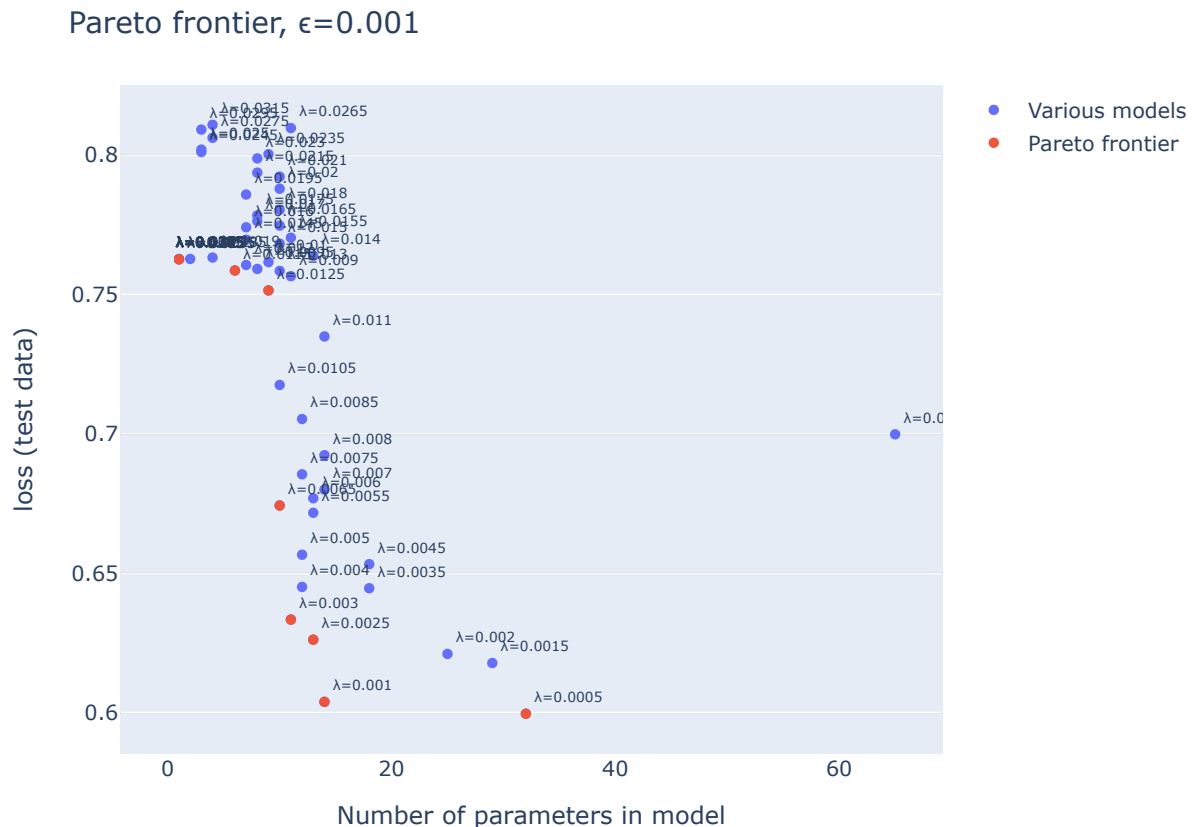


Figure 3.18: The Pareto frontier of the logistic regression with the L_1 penalization for $\lambda \in (0.0, 0.0005, \dots, 0.04)$ grid and the tolerance parameter $\varepsilon = 0.001$.

The Pareto frontier in Fig. 3.18 shows us several possible ways to choose a level of sparsity in the model depending on the test loss, always with a corresponding value of λ . Now it is again up to personal preference what parameterization to choose. For example, choosing $\lambda = 0.001$ can prune the neural network so that only 15 parameters remain, and the model *error* significantly decreases. At the cost of a small increase in test loss, the neural network can be pruned so that only the 3 most significant parameters remain.

3.2.4 Variational ADAM with ARD Prior in Logistic Regression

Now let us apply our proposed algorithm from Section 2.7 to this problem and thus search for a sparse parameterization using the variational inference. The experiment setting remains the same as in the MLE and L_1 cases. Plotting the convergence of the parameters is redundant in this case, since there is no analytical estimation as in the linear regression. Therefore it is appropriate to evaluate the sparsified models using the loss function evaluated on the test data.

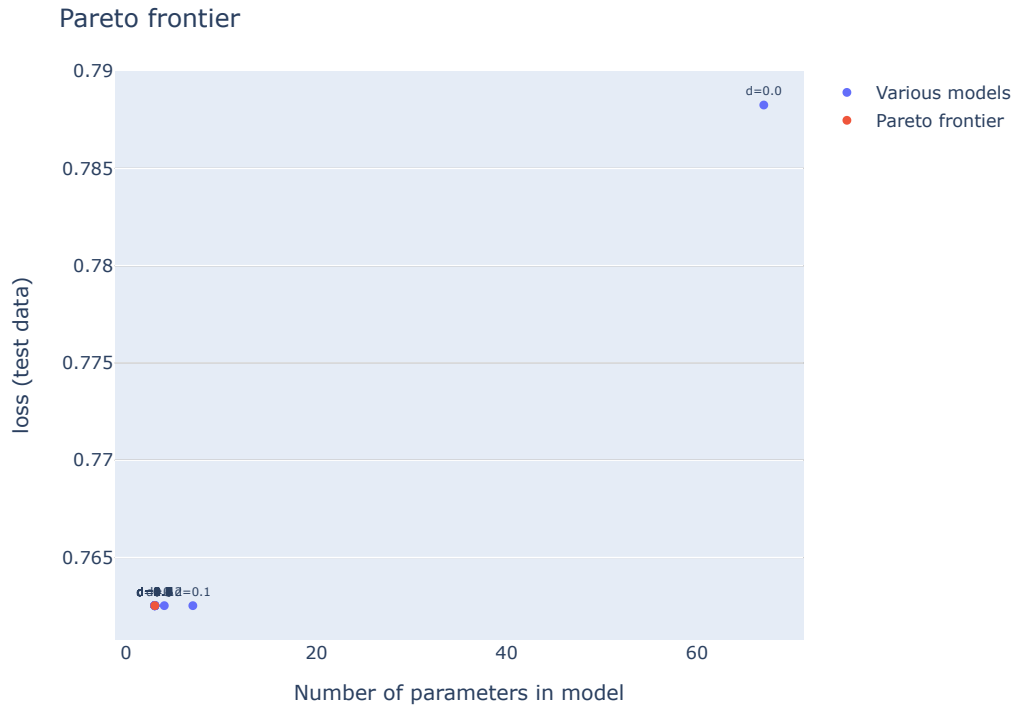


Figure 3.19: The Pareto frontier of the logistic regression using the VADAM optimizer with the ARD prior for $d \in (0.0, 0.1, \dots, 3.0)$ grid.

As can be seen in Fig. 3.19, our proposed method, relying on the VADAM optimizer with the ARD prior, found several sparse parameterizations, but even with an increasing number of standard deviations, the model *error* on the test data stabilized for only 3, 4 or 6 parameters of the model. As expected in Subsection 3.2.1, there are really just a few relevant parameters in such an overparameterized model sketched in Fig. 3.14. As revealed by our method, the model *error* with such a parameterization increases. We need to compare this method with L_1 penalization.

3.2.5 Evaluation of Methods

In a similar way to the linear regression, we would like to evaluate and compare the L_1 penalization and the VADAM optimizer with the ARD prior in terms of their ability to find sparser parameterizations. We take the Pareto frontiers from Fig. 3.18 and Fig. 3.19, plot them into one figure and compare them.

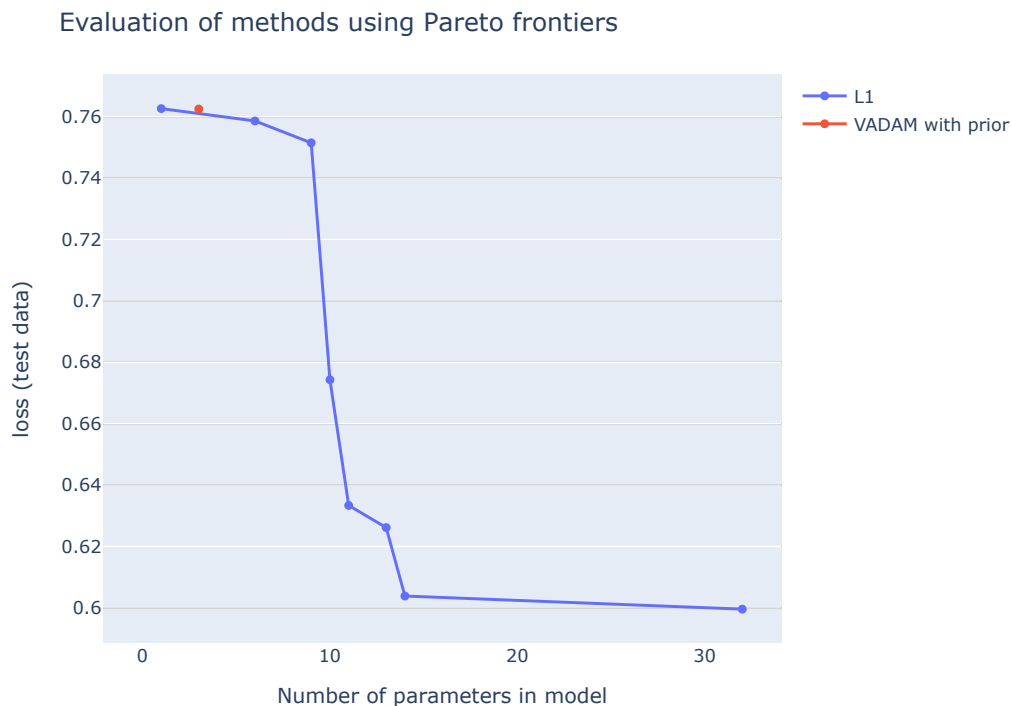


Figure 3.20: The evaluation of the methods to estimate a sparse parameterization in the experiment with the logistic regression.

Fig. 3.20 shows us that the ability to find a sparser parameterization is satisfied by the both methods. Here, the L_1 penalization has the advantage of providing multiple possible solutions for sparse parameterizations, whereas the VADAM optimizer with the ARD prior revealed only one possible sparse parameterization of the model. The ideal trade-off between a sparse parameterization and a minimal increase in the test *error* seems to be $\lambda = 0.001$, which leaves 15 significant parameters in the model. This fact also confirms Fig. 3.18. The VADAM optimizer with the ARD prior leaves only 3 parameters in the model, but the *error* on the test data is significantly higher than leaving a few more parameters in the model.

3.3 Sparse MIL

The third experiment is concerned with the multi-instance learning problem (as covered in Section 1.9). We adopt the dataset, \mathcal{D} , called Musk [52], which describes a set of 92 molecules (bags) of which 47 are judged by human experts to be musks and the remaining 45 molecules are judged to be non-musks. Our task is to classify whether the new molecule will be musk or non-musk. Along the lines of Section 1.9, we have 92 bags ($N = 92$) in toto, containing 476 instances ($I = 476$) of dimension 166 ($K = 166$) and 92 corresponding response variables, $\mathbf{y} \in \{0, 1\}^{92}$. Again, the dataset will be split into the train and test subsets with the ratio 0.8.

3.3.1 Model Architecture

The following architecture, see in Fig. 3.21, was proposed by the authors of the `Mill.jl` library [37], written in the `Julia` programming language. The library includes an example with this architecture, being suitable for the Musk dataset, on which we will test the L_1 regularization and the VADAM optimizer with the ARD prior to force a sparse parameterization.

The first part of the network is composed of an input layer for bags with instances of the fixed dimension 166 and a hidden layer with the tanh activation and 10 neurons. This is followed by a pooling layer with the *meanmax* aggregation operator whose output is a vector of the fixed dimension 20. The second part of the network contains another hidden layer with the tanh activation and 10 neurons, and is enclosed by an output layer with the binary output. In total, this architecture contains 1922 trainable parameters.

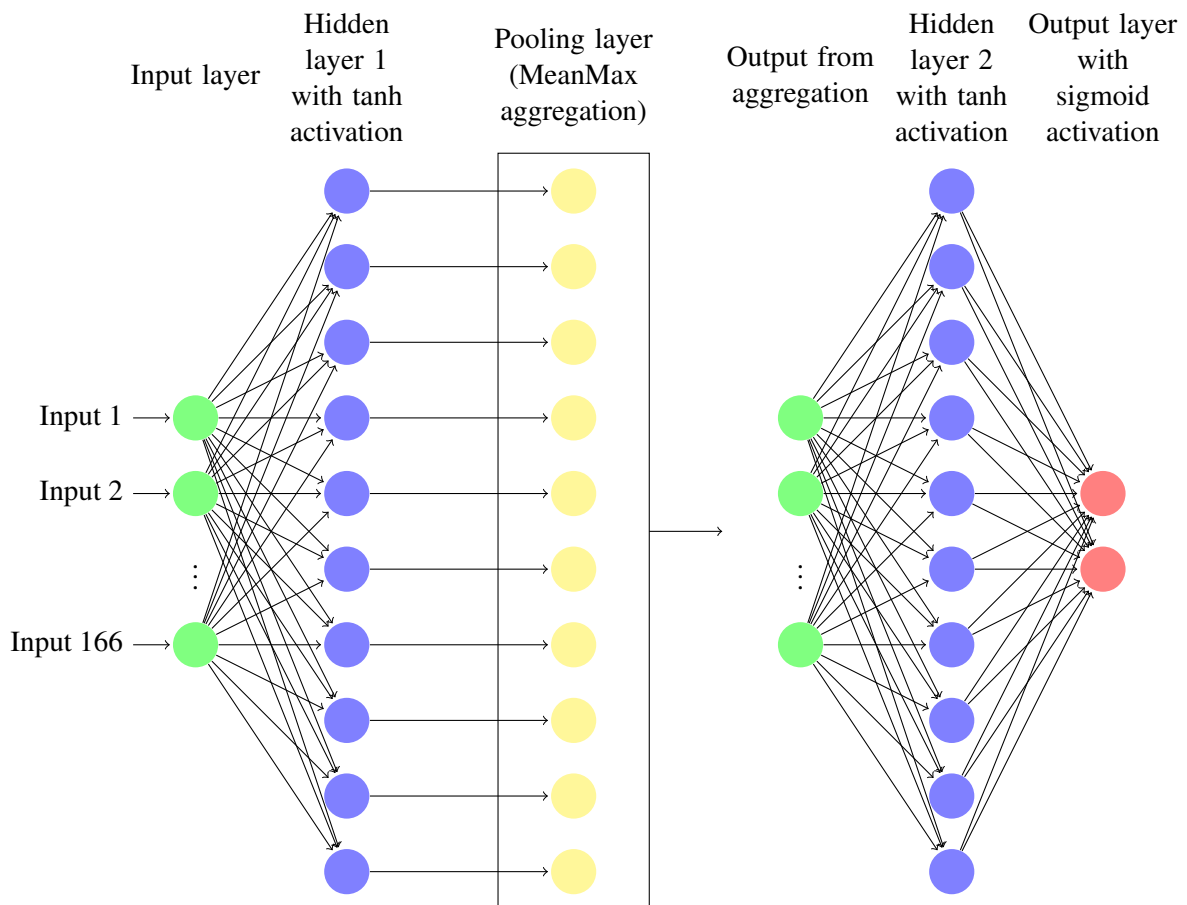


Figure 3.21: The neural network architecture for the MIL problem and the Musk dataset.

3.3.2 L_1 Penalization in MIL

Since we have a classification problem, *binary cross-entropy function* will be used as the loss to monitor the model *error*. Similar to the regression, we apply the Laplace prior with an additional hyper-parameter λ from Eq. (1.15). Despite the fact that there are now an order of magnitude more parameters, we plot them depending on the increasing λ during training. We set the $\lambda \in (0.0, 0.001, \dots, 0.1)$ grid (101 models in total) and, each time, we take a specific λ , train the parameters, record their values and plot. Then we apply the $L_{0,\varepsilon}$ norm to see how many parameters we can zero out based on a preferred tolerance parameter ε .

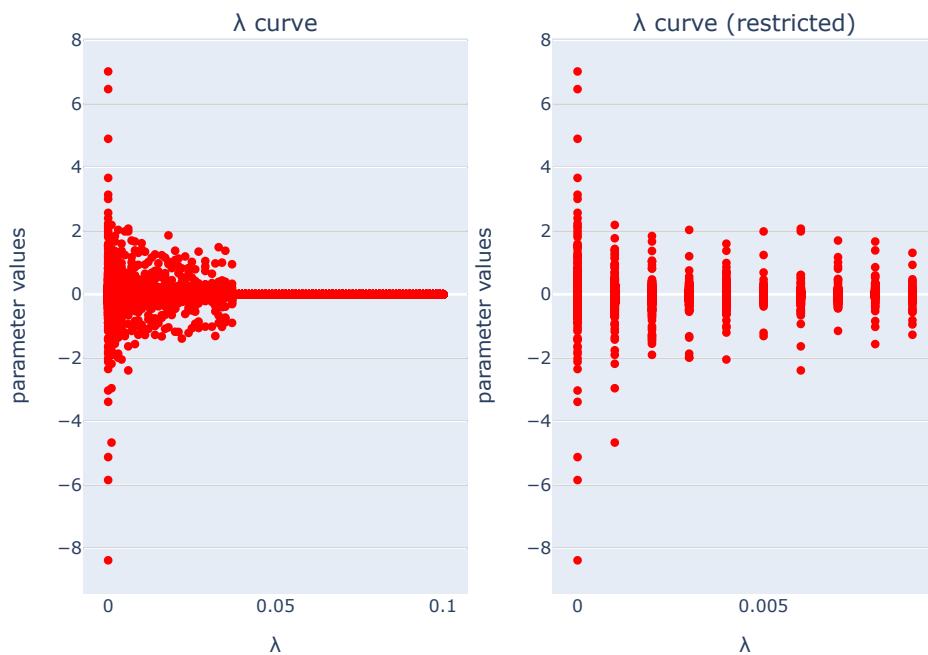


Figure 3.22: The relationship between λ and parameter values for $\lambda \in (0.0, 0.001, \dots, 0.1)$ grid (left) and $\lambda \in (0.0, 0.001, \dots, 0.01)$ grid (right).

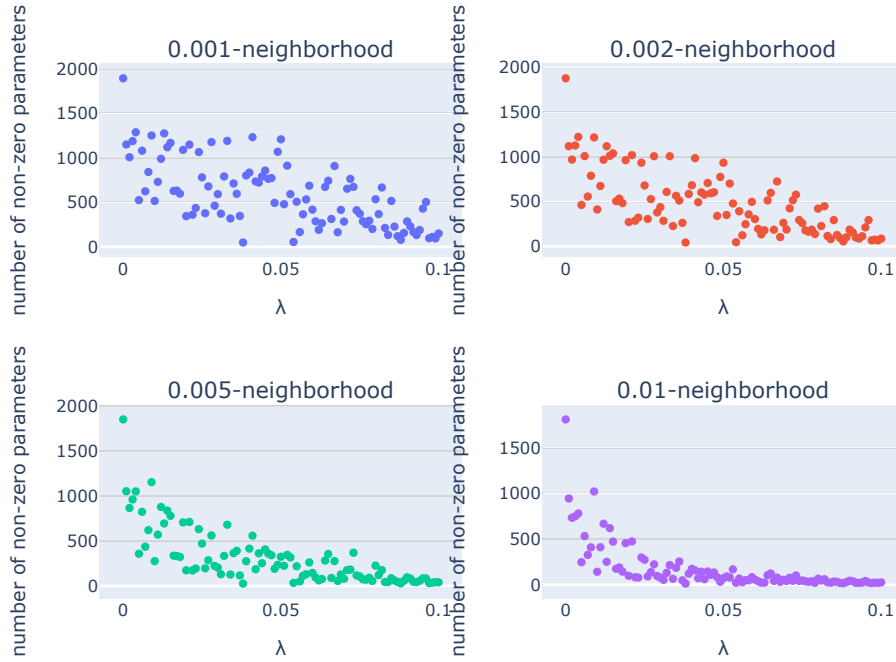


Figure 3.23: The relationship between λ and the number of the non-zero parameters depending on the ε -neighborhood given by $\varepsilon \in \{0.001, 0.002, 0.005, 0.01\}$.

From the Fig. 3.23, it is clear that for this problem we need to choose the tolerance parameter ε really small, so that we do not zero out all parameters with the small change of λ . We also use Algorithm 2 with the tolerance parameter $\varepsilon = 0.001$ and plot the Pareto frontier for the Musk problem.

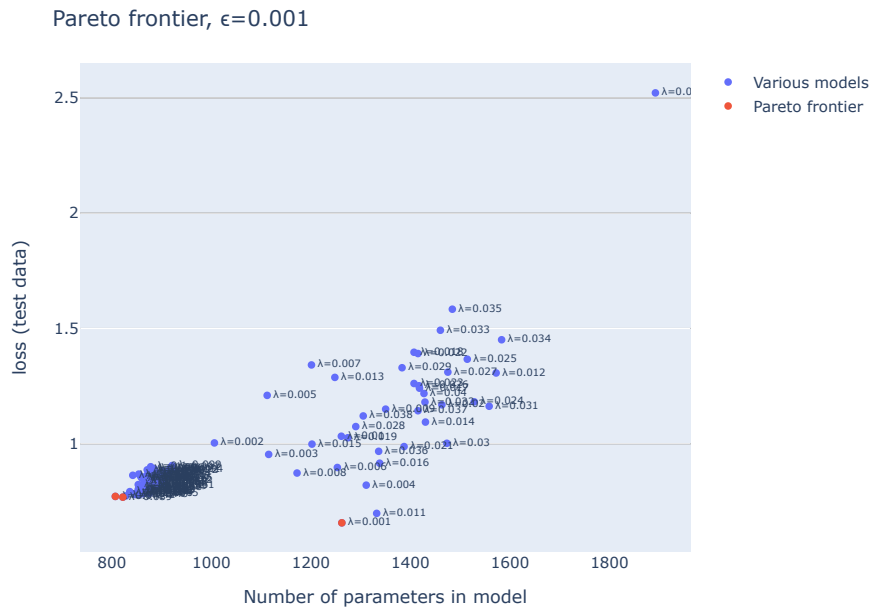


Figure 3.24: The Pareto frontier of the MIL problem using the L_1 penalization for $\lambda \in (0.0, 0.001, \dots, 0.1)$ grid and the tolerance parameter $\varepsilon = 0.001$.

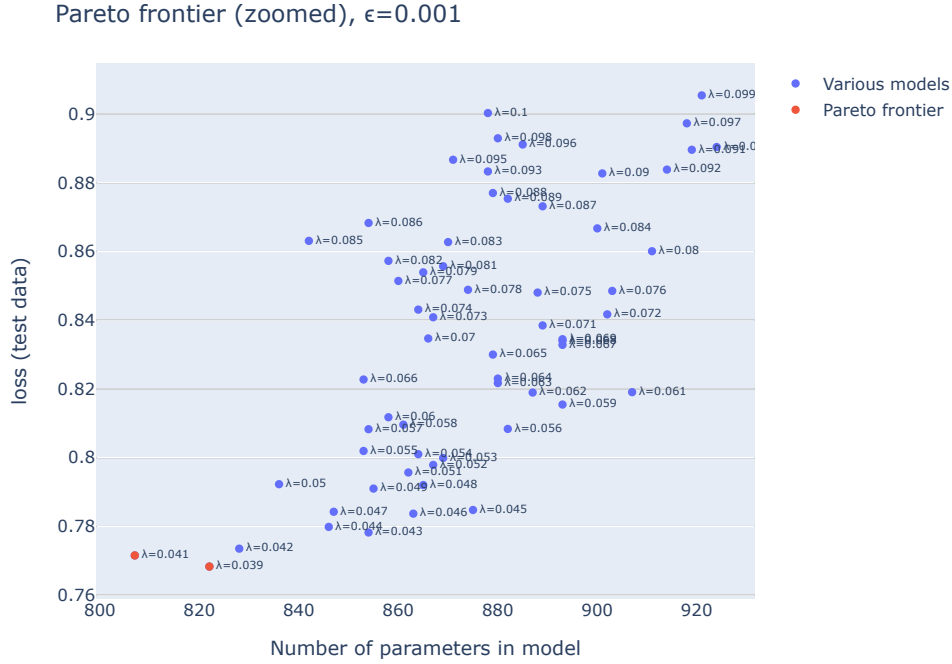


Figure 3.25: The Pareto frontier of the MIL problem using the L_1 penalization for $\lambda \in (0.0, 0.001, \dots, 0.1)$ grid and the tolerance parameter $\varepsilon = 0.001$ (zoomed Fig. 3.24 in the bottom-left clutter for $L_{0,\varepsilon} \in (800, 1000) \wedge \text{loss} \in (0.5, 1)$).

The Pareto frontier in Fig. 3.24 is a good help when tuning the λ hyperparameter, as it already zeroes out many parameters with small values and a specified tolerance parameter ε . It is, again, up to the subjective view of the reader whether they prefer a model with more parameters but less error, or fewer parameters and still an acceptable error. We have to determine our preferable ε . However, we must also remember that this is only valid with the previously mentioned smoothness of the λ grid and the tolerance parameter ε . Nonetheless, the sparse parameterization can be achieved. It is obvious that the neural network proposed in Fig. 3.21 is strongly overparameterized, since, by pruning the parameters in half, we can significantly reduce the model *error* on the test data.

This is confirmed by the choice $\lambda = 0.0$ in Fig. 3.24, where we obtain the test loss in the full model with 1922 parameters and without any penalization. This is in contrast to the choice $\lambda = 0.039$, which corresponds to the test loss with 822 parameters in the model, as depicted in Fig. 3.25.

3.3.3 Variational ADAM with ARD Prior in MIL

Since we have a dataset with real data and a more complex network architecture with more parameters than in the case of the linear regression, it is not valid to show the parameter convergence. This is because of the opacity of the convergence curves (1922 curves), and also because we do not have an analytical solution to which the parameters should converge (strongly non-linear model). From Fig. 3.22, we can assume that the vast majority of the parameters of the model will, indeed, have small values.

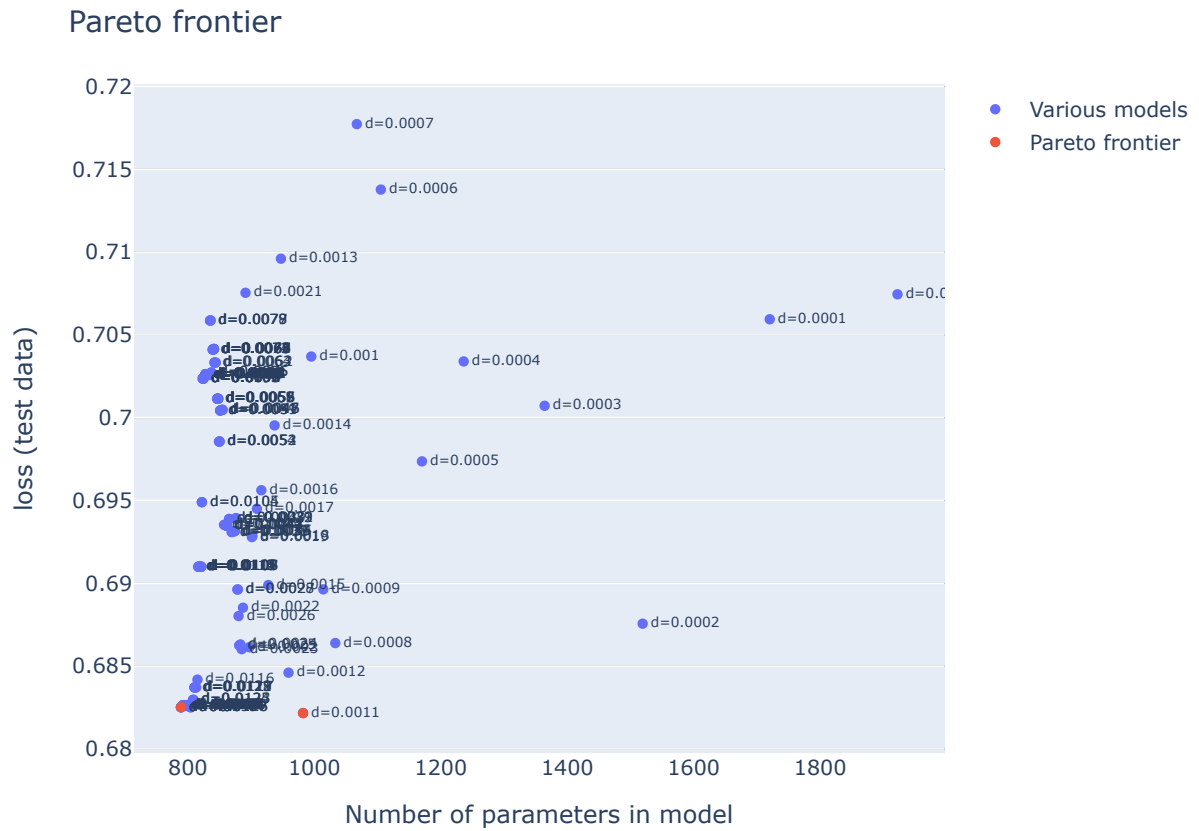


Figure 3.26: The Pareto frontier of the MIL problem using the VADAM optimizer with the ARD prior for $d \in (0.0, 0.0001, \dots, 0.015)$ grid.

The VADAM optimizer with the ARD prior found several possible sparse parameterizations, differing slightly in the value of the corresponding model *error*, which is illustrated in Fig. 3.26. The numbers of the standard deviations are set a bit tighter in this case, but this yields a more accurate estimate of the sparse parameterization and the number of relevant parameters (which differs rapidly).

3.3.4 Evaluation of Methods

In this third experiment, we would also like to evaluate and compare the L_1 penalization with the VADAM optimizer with the ARD prior. We compare the Pareto frontiers from Fig. 3.24, Fig. 3.25 (due to zoomed area) and Fig. 3.26 and plot them into a single figure (Fig. 3.27).

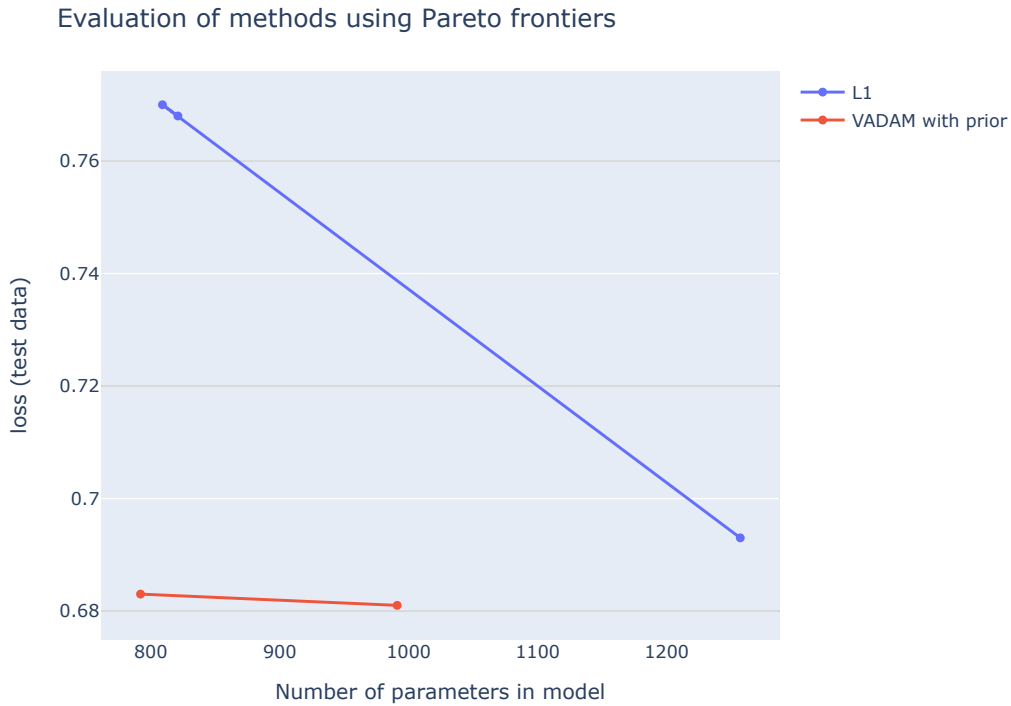


Figure 3.27: The evaluation of the methods to estimate a sparse parameterization in the experiment with the MIL.

We can see in Fig. 3.27, that both tested methods were able to prune the neural network sketched in Fig. 3.21 and found sparse parameterizations with significantly fewer parameters than the originally proposed architecture (1922 parameters). Our proposed method performs better in finding a sparse parameterization with two possible solutions containing of 792 or 991 relevant parameters.

We can state that the VADAM optimizer with the ARD prior successfully pruned the network from 1992 parameters to less than 1000 parameters, while still significantly reducing the test *error* with this sparser parameterization. On the other hand, it should also be noted that the second method, i.e., the L_1 penalization did not perform too badly. It pruned the network, found a sparse parameterization and reduced the test *error* of the original (non-sparisified) model, but not as much as the VADAM optimizer with the ARD prior.

Conclusion

This Master's thesis was focused on estimating sparse parameterizations of neural networks as a tool for pruning neural network architectures, i.e., achieving less complexity in such models. Deep neural networks can often suffer from overparametrization, which puts high demands on computing power to even train them properly. Therefore, this thesis presented and tested useful methods to achieve sparser parameterizations that can reduce the complexity of the network and also allow to quantify its uncertainty.

In the first chapter, a clear description of different shrinkage priors was introduced. These priors allowed to embed prior knowledge about sparse parameterization, thus reducing the number of irrelevant parameters in the model. It was outlined that it is not always possible to analytically compute the posterior distribution of the parameters. Making it necessary to introduce the variational Bayes framework. It was also illustrated how, in some special cases of neural networks, i.e., linear and logistic regression, one can approximate, or, even analytically derive, the posterior distribution of the parameters, using a particular shrinkage prior. The chapter concluded with a theoretical insight into the concept of multi-instance learning, which uses the specific learning structure, as opposed to conventional machine learning.

The second chapter dealt with variational optimization and inference. A comprehensive interpretation of methods using natural gradients as a tool to avoid the demanding, and expensive computation of Hessian matrices and inverse Fisher matrices during optimization, was given. Various variational methods were derived to approximate the intractable posterior distribution of the parameters. In particular, the variational RMSprop and variational ADAM optimizers, for which connections with the conventional optimizers were shown. At the end of the second chapter, the author's newly proposed variational method was described. This method consisted of combining the variational ADAM method with the shrinkage prior and aimed at finding the posterior distribution of the parameters with a sparse parameterization.

The third chapter tested the author's proposed method on different neural network architectures. In total, three major experiments were carried out on synthetic and real data. The first experiment with linear regression and synthetic data studied the ability of the conventional and variational methods to search for a sparse parameterization that would corresponded to a known solution so that convergence could be verified. Here, the proposed method was found to perform similarly to the L_1 penalization widely used against overfitting in practise, and its suitability for finding sparse parameterizations was confirmed. The second experiment focused on a deeper network, performing classification on the Iris dataset. The goal was to reveal how much the network can be pruned to still find a sparse parameterization with a minimal error growth on the test data. In this respect, the L_1 penalization and the author's proposed method were again compared. Although sparse parameterizations were found by both methods, the L_1 penalization performed better in parameter selection at the cost of less error on the test data. The author's proposed method found an overly strict sparse solution, achieving a higher error on the test data. The third experiment studied whether it is possible to prune the neural network designed on group data from the Musk dataset, while reducing its error on the test data. The proposed method was able to prune the network more than the L_1 penalization, including a significant reduction of error on the test data. Overall, it can be stated that the author's proposed method has a good potential for finding sparse

parameterizations and can be used as an alternative to conventional tools preventing overfitting in deep neural networks, to make them less complex.

In the field of neural network optimization, the topic of sparse parameterizations is particularly useful as a tool to prevent overfitting or to reduce model complexity. The ability of variational methods to detect irrelevant parameters in the neural network, and thus prune them, can be really helpful in designing the resulting model architectures. Therefore, this thesis could give the reader guidance on how to incorporate prior knowledge of a sparse parameterization into deep learning models, and how to use the proposed methods to achieve it. And not only achieve a sparse parameterization, but also the inferred posterior distribution of all the parameters in neural network, with which its uncertainty can be quantified.

Appendix A

Variational Online-Newton

To express Eq. (2.25) we firstly derive the ELBO in the natural parameterization with VON assumption of the factorized likelihood:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\eta}) &= \mathbb{E}_q [\log p(\mathcal{D}, \mathbf{z}) - \log q(\mathbf{z}|\boldsymbol{\eta})] = \mathbb{E}_q \left[\log \left(p(\mathcal{D}|\mathbf{z}) p(\mathbf{z}) \right) - \log q(\mathbf{z}|\boldsymbol{\eta}) \right] \\
&= \mathbb{E}_q [\log p(\mathcal{D}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\boldsymbol{\eta})] = \mathbb{E}_q \left[\log p(\mathcal{D}|\mathbf{z}) + \log \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\eta})} \right) \right] \\
&= \mathbb{E}_q \left[\frac{N}{N} \sum_{n=1}^N \log p(\mathcal{D}_n|\mathbf{z}) + \log \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\eta})} \right) \right].
\end{aligned} \tag{A.1}$$

The variational lower-bound in Eq. (A.1) can be rewritten for the model parameters as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta) &= \mathbb{E}_q [\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})] \\
&= \mathbb{E}_q [-Nf(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\eta})].
\end{aligned} \tag{A.2}$$

For obtaining the gradients with respect to $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ we use this derivation of Eq. (2.27)

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_\theta} \mathcal{L} &= \nabla_{\boldsymbol{\mu}_\theta} \mathbb{E}_q [-Nf(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\eta})] \\
&= -\mathbb{E}_q [N\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})] - \boldsymbol{\psi} \boldsymbol{\mu}_\theta \\
&= -\mathbb{E}_q [N \mathbf{g}(\boldsymbol{\theta})] - \boldsymbol{\psi} \boldsymbol{\mu}_\theta \\
\nabla_{\boldsymbol{\Sigma}_\theta} \mathcal{L} &= \frac{1}{2} \mathbb{E}_q \left[-N\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}) \right] - \frac{1}{2} \boldsymbol{\psi} \mathbf{I} + \frac{1}{2} \boldsymbol{\Sigma}_\theta^{-1} \\
&= \frac{1}{2} \mathbb{E}_q [-N\mathbf{H}(\boldsymbol{\theta})] - \frac{1}{2} \boldsymbol{\psi} \mathbf{I} + \frac{1}{2} \boldsymbol{\Sigma}_\theta^{-1}.
\end{aligned} \tag{A.5}$$

Final derivation of the natural-gradient updates for $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ in Eq. (2.29):

$$\begin{aligned}
\boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} + \beta_{(t)} \boldsymbol{\Sigma}_{\theta,(t+1)} \left(\nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}_{(t)} \right) \\
&= \boldsymbol{\mu}_{\theta,(t)} + \beta_{(t)} \boldsymbol{\Sigma}_{\theta,(t+1)} \left(-N \mathbf{g}(\boldsymbol{\theta}_{(t)}) - \boldsymbol{\psi} \boldsymbol{\mu}_{\theta,(t)} \right) \\
&= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \boldsymbol{\Sigma}_{\theta,(t+1)} \left(N \mathbf{g}(\boldsymbol{\theta}_{(t)}) + \boldsymbol{\psi} \boldsymbol{\mu}_{\theta,(t)} \right)
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_{\theta,(t+1)}^{-1} &= \boldsymbol{\Sigma}_{\theta,(t)}^{-1} - 2\beta_{(t)} \left(\nabla_{\boldsymbol{\Sigma}_\theta} \mathcal{L}_{(t)} \right) \\
&= \boldsymbol{\Sigma}_{\theta,(t)}^{-1} - 2\beta_{(t)} \left[-\frac{1}{2} N\mathbf{H}(\boldsymbol{\theta}_{(t)}) - \frac{1}{2} \boldsymbol{\psi} \mathbf{I} + \frac{1}{2} \boldsymbol{\Sigma}_{\theta,(t)}^{-1} \right] \\
&= \boldsymbol{\Sigma}_{\theta,(t)}^{-1} + \beta_{(t)} N\mathbf{H}(\boldsymbol{\theta}_{(t)}) + \beta_{(t)} \boldsymbol{\psi} \mathbf{I} - \beta_{(t)} \boldsymbol{\Sigma}_{\theta,(t)}^{-1} \\
&= (1 - \beta_{(t)}) \boldsymbol{\Sigma}_{\theta,(t)}^{-1} + \beta_{(t)} [N\mathbf{H}(\boldsymbol{\theta}_{(t)}) + \boldsymbol{\psi} \mathbf{I}].
\end{aligned} \tag{A.7}$$

By defining a transform matrix $\mathbf{S}_{\theta,(t)} = \frac{1}{N} (\boldsymbol{\Sigma}_{\theta,(t)}^{-1} - \psi \mathbf{I}) \rightarrow \boldsymbol{\Sigma}_{\theta,(t)} = [N (\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I})]^{-1}$ we derive more compact shape of updates equations showed in Eq. (2.30):

$$\begin{aligned} \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[N \left(\mathbf{S}_{\theta,(t+1)} + \frac{1}{N} \psi \mathbf{I} \right) \right]^{-1} \left[N \mathbf{g}(\boldsymbol{\theta}_{(t)}) + \psi \boldsymbol{\mu}_{\theta,(t)} \right] \\ &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\left(\mathbf{S}_{\theta,(t+1)} + \frac{1}{N} \psi \mathbf{I} \right) \right]^{-1} \left[\mathbf{g}(\boldsymbol{\theta}_{(t)}) + \frac{1}{N} \psi \boldsymbol{\mu}_{\theta,(t)} \right], \end{aligned} \quad (\text{A.8})$$

$$\begin{aligned} \boldsymbol{\Sigma}_{\theta,(t+1)}^{-1} &= (1 - \beta_{(t)}) \boldsymbol{\Sigma}_{\theta,(t)}^{-1} + \beta_{(t)} [N \mathbf{H}(\boldsymbol{\theta}_{(t)}) + \psi \mathbf{I}] \\ \frac{N}{N} (\boldsymbol{\Sigma}_{\theta,(t+1)}^{-1} - \psi \mathbf{I} + \psi \mathbf{I}) &= (1 - \beta_{(t)}) \left[N \left(\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I} \right) \right] + \beta_{(t)} (N \mathbf{H}(\boldsymbol{\theta}_{(t)}) + \psi \mathbf{I}) \\ N \left(\frac{1}{N} (\boldsymbol{\Sigma}_{\theta,(t+1)}^{-1} + \psi \mathbf{I}) + \frac{\psi \mathbf{I}}{N} \right) &= (1 - \beta_{(t)}) \left[N \left(\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I} \right) \right] + \beta_{(t)} (N \mathbf{H}(\boldsymbol{\theta}_{(t)}) + \psi \mathbf{I}) \\ \mathbf{S}_{\theta,(t+1)} + \frac{\psi \mathbf{I}}{N} &= (1 - \beta_{(t)}) \left(\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I} \right) + \beta_{(t)} \left(\mathbf{H}(\boldsymbol{\theta}_{(t)}) + \frac{\psi \mathbf{I}}{N} \right) \\ \mathbf{S}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \left(\mathbf{S}_{\theta,(t)} + \frac{1}{N} \psi \mathbf{I} \right) + \beta_{(t)} \left(\mathbf{H}(\boldsymbol{\theta}_{(t)}) + \frac{\psi \mathbf{I}}{N} \right) - \frac{\psi \mathbf{I}}{N} \\ \mathbf{S}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \mathbf{S}_{\theta,(t)} + \beta_{(t)} \mathbf{H}(\boldsymbol{\theta}_{(t)}). \end{aligned} \quad (\text{A.9})$$

And in case of mean-field variant of the VON method in Eq. (2.31):

$$\begin{aligned} \boldsymbol{\mu}_{\theta,(t+1)} &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\left(\mathbf{s}_{\theta,(t+1)} + \frac{\psi}{N} \right) \right]^{-1} \left[\mathbf{g}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\theta,(t)} \right] \\ &= \boldsymbol{\mu}_{\theta,(t)} - \beta_{(t)} \left[\frac{\mathbf{g}(\boldsymbol{\theta}_{(t)}) + \frac{\psi}{N} \boldsymbol{\mu}_{\theta,(t)}}{\mathbf{s}_{\theta,(t+1)} + \frac{\psi}{N}} \right] \\ \mathbf{s}_{\theta,(t+1)} &= (1 - \beta_{(t)}) \mathbf{s}_{\theta,(t)} + \beta_{(t)} \text{diag}(\mathbf{H}(\boldsymbol{\theta}_{(t)})). \end{aligned} \quad (\text{A.10})$$

Reparameterization Trick in Hessian

The use of the reparameterization trick in Eq. (2.33) is properly derived as follows

$$\begin{aligned} \mathbb{E}_q \left[\nabla_{\theta\theta}^2 f(\boldsymbol{\theta}) \right] &= 2 \nabla_{\sigma_\theta^2} \mathbb{E}_q [f(\boldsymbol{\theta})] \\ &= 2 \nabla_{\sigma_\theta^2} \mathbb{E}_{\mathcal{N}(\mathbf{e}|0, \mathbf{I})} [f(\boldsymbol{\mu}_\theta + \boldsymbol{\sigma}_\theta \circ \mathbf{e})] \\ &= 2 \mathbb{E}_{\mathcal{N}(\mathbf{e}|0, \mathbf{I})} \left[\nabla_{\sigma_\theta^2} f(\boldsymbol{\mu}_\theta + \boldsymbol{\sigma}_\theta \circ \mathbf{e}) \right] \\ &= 2 \mathbb{E}_{\mathcal{N}(\mathbf{e}|0, \mathbf{I})} \left[\nabla_{\sigma_\theta^2} f(\boldsymbol{\mu}_\theta + \sqrt{\boldsymbol{\sigma}_\theta \circ \boldsymbol{\sigma}_\theta} \circ \mathbf{e}) \right] \\ &= 2 \mathbb{E}_{\mathcal{N}(\mathbf{e}|0, \mathbf{I})} \left[\nabla_\theta f(\boldsymbol{\theta}) \cdot \frac{1}{2} \frac{\mathbf{e}}{\boldsymbol{\sigma}_\theta} \right] \\ &= \mathbb{E}_{\mathcal{N}(\mathbf{e}|0, \mathbf{I})} \left[\nabla_\theta f(\boldsymbol{\theta}) \frac{\mathbf{e}}{\boldsymbol{\sigma}_\theta} \right] \\ &\approx \hat{\mathbf{g}}(\boldsymbol{\theta}) \frac{\mathbf{e}}{\boldsymbol{\sigma}_\theta}, \end{aligned} \quad (\text{A.11})$$

where $\mathbf{e} \sim \mathcal{N}(\mathbf{e}|0, \mathbf{I})$ and $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma}_\theta \circ \mathbf{e}$.

Bibliography

- [1] AGARAP A. F. *Deep learning using rectified linear units (relu)*. arXiv preprint arXiv :1803.08375, 2018.
- [2] BANERJEE M. *L_p spaces, metrics on spaces of probabilities, and connections to estimation*. [online]. [cit. 2022-12-03]. University of Michigan. 2006. Available from: <http://dept.stat.lsa.umich.edu/~moulib/stat612-notes-0.pdf>.
- [3] BHATTACHARYA A., et al. *Bayesian shrinkage*. arXiv preprint arXiv:1212.6088, 2012.
- [4] BISHOP Ch. M., NASRABADI N. M. *Pattern recognition and machine learning*. New York: springer, 2006.
- [5] BLEI D. M., KUCUKELBIR A., MCAULIFFE J. D. *Variational inference: A review for statisticians*. Journal of the American statistical Association, 2017, 112.518: 859-877.
- [6] BLUNDELL Ch., et al. *Weight uncertainty in neural network*. In: ‘International conference on machine learning’. PMLR, 2015. p. 1613-1622.
- [7] BORSBOOM D., MELLENBERGH G. J., VAN HEERDEN J. *The theoretical status of latent variables*. Psychological review, 2003, 110.2:203.
- [8] BOTTOU L., CURTIS F., NOCEDAL J. *Optimization methods for large-scale machine learning*. Siam Review, 2018, 60.2: 223-311.
- [9] CARVALHO C. M., POLSON N. G., SCOTT J. G. *Handling sparsity via the horseshoe*. In: ‘Artificial Intelligence and Statistics’. PLMR, 2009. p. 73-80.
- [10] DRUILHET P., POMMERET D. *Invariant conjugate analysis for exponential families*. Bayesian Analysis, 2012, 7.4: 903-916.
- [11] ENGELHARDT B., HE X., PAN J., RAZEEN A., SRISTAVA A. *STA651: Probabilistic machine learning - Gaussian Models*. Princeton University. Department of Computer Science. [online]. [cit. 2022-05-03]. Available from: https://www.cs.princeton.edu/~bee/courses/scribe/lec_09_09_2013.pdf.
- [12] FINK D. *A compendium of conjugate priors*. Montana State University. Department of Biology. [online]. [cit. 2022-14-04]. Available from: <https://web.archive.org/web/20090529203101/http://www.people.cornell.edu/pages/df36/CONJINTRnew%20TEX.pdf>.
- [13] GODOY D. *Understanding binary cross-entropy/log loss: a visual explanation*. [online]. [cit. 2022-06-04]. Available from: <https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>.

- [14] HANS Ch. *Bayesian lasso regression*. *Biometrika*, 2009, 96.4: 835-845.
- [15] HANSEN P., O'LEARY D *The use of the L-curve in the regularization of discrete ill-posed problems*. *SIAM journal on scientific computing*, 1993, 14.6: 1487-1503.
- [16] HOBZA T. *Zobecněné lineární modely – vysokoškolské skriptum*. 2021. Praha. Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze.
- [17] HOFFMAN M. D., et al. *Stochastic variational inference*. *Journal of Machine Learning Research*, 2013.
- [18] HOFFMAN M. D., JOHNSON M. J. *Elbo surgery: yet another way to carve up the variational evidence lower bound*. In: 'Workshop in Advances in Approximate Bayesian Inference', NIPS. 2016.
- [19] GOLDBERGER J., et al. *An efficient image similarity measure based on approximations on KL-divergence between two gaussian mixtures*. In: 'ICCV'. 2003. p. 487-493.
- [20] JOSPIN L. V., et al. *Hands-on Bayesian Neural networks—a tutorial for deep learning users*. *IEEE Computational Intelligence Magazine*, 2022, 17.2: 29-48.
- [21] JORDAN M. I. *The exponential family: Conjugate priors*. [online]. [cit. 2022-06-03]. Available from: <https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf>.
- [22] KHAN M., et al. *Fast and scalable bayesian deep learning by weight-perturbation in Adam*. In: 'International Conference on Machine Learning', PMLR, 2018, p. 2611-2620.
- [23] KHAN M., et al. *Variational adaptive-Newton method for explorative learning*. arXiv preprint arXiv: 1711.05560, 2017.
- [24] KINGMA D. P., BA J. *Adam: A method for stochastic optimization*. arXiv preprint arXiv: 1412.6980, 2014.
- [25] KINGMA D. P., WELLING M. *Stochastic gradient VB and the variational auto-encoder*. In: 'Second International Conference on Learning Representations', ICLR. 2014. p. 121.
- [26] KRBÁLEK M. *Rovnice matematické fyziky*. Nakladatelství ČVUT v Praze. 2012.
- [27] LEEMIS L. *Exponential_Rayleigh*. College of William & Mary. Department of Mathematics. [online]. [cit. 2021-06-08]. Available from: <http://www.math.wm.edu/~leemis/chart/UDR/PDFs/ExponentialRayleigh.pdf>.
- [28] LIN W., KHAN M. E., SCHMIDT M. *Fast and simple natural-gradient variational inference with mixture of exponential-family approximations*. In 'International Conference on Machine Learning', PMLR. 2019, 3992-4002.
- [29] LOUZADA F., RAMOS P. L., NASCIMENTO D. *The inverse Nakagami-m distribution: A novel approach in reliability*. *IEEE Transactions on Reliability*, 2018, 67.3: 1030-1042.
- [30] NALISNICK E., HERNÁNDEZ-LOBATO J. M., SMYTH P. *Dropout as a structured shrinkage prior*. In: 'International Conference on Machine Learning', PMLR. 2019. p: 4712-4722.
- [31] NWANKPA Ch., et al. *Activation functions: Comparison of trends in practise and reasearch for deep learning*. arXiv preprint arXiv: 1811.03378, 2018.

- [32] MALSINER-WALLI G., WAGNER H. *Comparing spike and slab priors for Bayesian variable selection*. arXiv preprint arXiv: 1810.07259, 2018.
- [33] MANDLÍK Š. *Mapping the Internet - Modelling Entity Interactions in Complex Heterogenous Networks*. Prague. 2020. Master's Thesis. CTU in Prague, Faculty of Electrical Engineering.
- [34] MARTENS J. *New insights and perspectives on the natural gradient method*. arXiv preprint arXiv:1412.1193, 2014.
- [35] OPPER M., ARCHAMBEAU C. *The variational Gaussian approximation revisited*. *Neural Computation*, 2009, 21.3: 786-792.
- [36] PAIRMAN E. *Tables of the digamma and trigamma functions*. Cambridge University Press. Chicago, 1919.
- [37] PEVNÝ T., MANDLÍK Š. *Mill.jl framework: a flexible library for (hierarchical) multi-instance learning*. [online]. [cit. 2022-23-04]. Available from: <https://github.com/CTUAvastLab/Mill.jl>.
- [38] PEVNÝ T., SOMOP P. *Discriminative models for multi-instance problems with tree structure*. In: 'Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security'. 2016. p. 83-91.
- [39] PEVNÝ T., SOMOL P. *Using neural network formalism to solve multiple-instance problems*. In: 'International Symposium on Neural Networks'. Springer, Cham. 2017. p. 135-142.
- [40] *Plotting graph for IRIS Dataset Using Seaborn and Matplotlib*. [online]. [cit-2022-28-04]. Available from: <https://www.geeksforgeeks.org/plotting-graph-for-iris-dataset-using-seaborn-and-matplotlib/>.
- [41] POLYANSKIY Y. *f-divergences*. Massachusetts Institute of Technology, MIT. [online]. [cit. 2022-09-03]. Available from: http://people.lids.mit.edu/yp/homepage/data/LN_fdiv.pdf.
- [42] POPKENS A., et al. *Interpretable outcome prediction with sparse Bayesian neural networks in intensive care*. arXiv preprint arXiv: 1905.02599, 2019, 29.34: 107.
- [43] MUKKAMALA M. Ch., HEIN. M. *Variants of RMSprop and adagrad with logarithmic regret bounds*. In 'International Conference on Machine Learning', PMLR, 2017. p: 2545-2553.
- [44] ROBERT Ch. P., CHOPIN N., ROUSSEAU J. *Harold Jeffreys's theory of probability revisited*. *Statistical Science*, 2009, 24.2: 141-172.
- [45] SHAH A. *Understand Bayes Rule, Likelihood, Prior and Posterior*. [online]. [cit. 2022-14-04]. Available from: <https://towardsdatascience.com/understand-bayes-rule-likelihood-prior-and-posterior-34eae0f378c5>.
- [46] SHAIKH R. *Cross Validation Explained: Evaluating estimator performance*. [online]. [cit. 2022-07-04]. Available from: <https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>.
- [47] SCHRAUDOLPH N., YU J., GÜNTER S. *A stochastic quasi-Newton method for online convex optimization*. In: 'Artificial intelligence and statistics'. PMLR, 2007. p. 436-443.
- [48] SIKKA K., GIRI R., BARTLETT M. S. *Joint Clustering and Classification for Multiple Instance Learning*. In: 'BMVC'. 2015. p. 71.1-71.12.

- [49] SYVERSVEEN A. R. *Noninformative bayesian priors: interpretation and problems with construction and applications*. Preprint statistics, 1998, 3.3: 1-11.
- [50] ŠMÍDL V. *Hierarchical Bayesian Models – Linear Regression*. [cit. 2021-05-08]. Available from: <http://staff.utia.cz/smidl/files/hbm/prezentace04.pdf>.
- [51] UCI MACHINE LEARNING REPOSITORY *Iris Data Set*. [online]. [cit. 2022-06-04]. Available from: <https://archive.ics.uci.edu/ml/datasets/iris>.
- [52] UCI MACHINE LEARNING REPOSITORY *Musk Data Set*. [online]. [cit. 2022-22-04]. Available from: [https://archive.ics.uci.edu/ml/datasets/Musk+\(Version+1\)](https://archive.ics.uci.edu/ml/datasets/Musk+(Version+1)).
- [53] VAN ERP S., OBERSKI D. L., MULDER J. *Shrinkage priors for Bayesian penalized regression*. Journal of Mathematical Psychology, 2019, 89: 31-50.
- [54] WANG S., et al. *Variational HyperAdam: A meta-learning Approach to Network Training*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.
- [55] WIPF D., NAGARAJAN S. *A new view of automatic relevance determination*. Advances in neural information processing systems, 2007, 20.
- [56] WRIGHT S., et al. *Numerical optimization*. Springer Science, 1999, 35.67-68: 7.