Technical University of Berlin

Institute of Software Engineering and Theoretical Computer Science
Research Group Artificial Intelligence

Masters Thesis

Scalable Inference for Correlated Noise Classification Models

Lorenz Vaitl

Matriculation Number: 384937
November 26, 2018

Supervised by
Prof. Dr. Manfred Opper

Assistant Supervisor
Prof. Dr. Klaus Obermayer

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, 26.11.2018

.....

(Signature Lorenz Vaitl)

Abstract

This thesis tackles classification of data that has been disturbed by correlated noise. This noise can be introduced by different sources, such as artifacts during measurement or unwanted structure in the data. The presented variational approach models linear dependencies and correlated noise by using a Linear Mixed Model, consisting of a linear classifier and a Sparse Gaussian Process. Modeling both parts, the method can recover the latent linear dependencies in a Bayesian fashion. The likelihood of the logit classifier is made analytically tractable by using data augmentation, which allows efficient fitting of the variational distributions to the data with Variational Inference (VI). Different priors can be used to enforce sparsity of the linear part. We present VI algorithms for two sparse priors, the Laplace and the Horseshoe prior.
The presented algorithm scales linearly to the number of dimensions by using dynamic programming and cubically to the number of inducing points, which do not directly depend on the number of samples.
Experiments on synthetic and real data show the behavior and performance of the model, which are then compared to baselines. While at worst being competitive on all used data, the presented classifier trumps the competition especially in a high-dimensional setting with few samples, shows a higher feature selection stability and better generalization to unseen subcategories in data with a hierarchical class-structure.

Zusammenfassung

Diese Abschlussarbeit behandelt die Klassifikation von Daten, die durch korrelierte Störfaktoren beeinträchtigt sind. Ursachen für diese Störungen können beispielsweise Artefakte bei der Messung oder unerwünschte Strukturen in den Daten sein. Der hier eingeführte Inferenz-Alogrithmus modelliert sowohl lineare Zusammenhänge, als auch korreliertes Rauschen, indem er ein lineares gemischtes Modell mit einem linearen Teil und einem Gauss'schen Prozess benutzt. Das Modell ist so gewählt, dass es das Rauschen und den zugrunde liegenden linearen Teil durch probabilistische Modellierung mit einem Bayes'schen Ansatz separieren kann.
Durch Augmentierung mit einer Pòlya-gamma Verteilung wird die Likelihood des Logit-Klassifizierers in eine analytisch handhabbare Form gebracht. Hiermit können die benutzten variationellen Verteilungen effizient mit Variational Inference (VI) auf die Daten angepasst werden. Zwei a-priori Verteilungen werden in verschiedenen Versionen in den Algorithmus eingefügt. Diese Verteilungen sorgen für ein simples lineares Modell mit wenigen Faktoren.
Die Zeitkomplexität des Algorithmus' erlaubt die Benutzung auf großen Datenmengen, sowohl mit einer großen Anzahl von Datenpunkten als auch mit einer hohen Dimensionalität. Die Komplexität ist linear in Relation zur Dimensionalität und ist nur indirekt abhängig von der Anzahl der Datenpunkte, da inducing points benutzt werden. In Experimenten auf simulierten und echten Daten wird das Verhalten und die Performanz des Modells getestet und mit anderen Verfahren verglichen. Gerade in hochdimensionalen Problemen mit wenigen Datenpunkten zeigt der Klassifizierer ein besseres Verhalten als die Konkurrenz.

Contents

List of Figures

1. Introduction

In Machine Learning problems can arise from the untreated occurrence of noise. Apart from the fact that noise always leads to uncertainty, correlated noise also can introduce unwanted structure into the data. This in turn can be a pitfall, as these structures might be present in the observed data, but not in unseen data, leading to a model that overfits. An example for this can be found in statistical genetics, where confounders can effect the subjects of studies locally. Health risks, for example, are not only subject to genes, but also to the healthcare sector. Only looking at dependencies between genes and diseases can then lead to false conclusions. Another case are attributes in people that are cause for discrimination, like race and sex. These attributes can be seen as confounders, which we want the model to ignore.
The model used here is a Linear Mixed Model (LMM) similar to Mandt et al. [29], which consists of two parts, namely a linear model coupled with an estimate of the correlated noise. The latter part is realized with a Gaussian Process (GP). In order to use this classification model in a Bayesian way, data augmentation with a Pòlya-gamma variable [37] is applied. By marginalization, the variational distributions for the linear and the noise model can be used to predict unseen data, including or excluding the confounding noise, as learned by the model.
The model can also be used for outlier detection, where –ideally– the correlated noise models the similarity between members of a single inlier class and the linear part models the underlying structure of the ensemble of inlier classes. A simple visualization of the assumed latent model can be seen in Figure 1.1. The color of a region shows its class, the underlying linear dependencies are in the left image, which together with the confounding noise, leads to the observable data on the right. We want to recover the original linear weights, because, while correlated noise is present in our dataset, this does not generalize to all data.
In order to recover the latent linear weights, the LMM needs to separate the two parts. We tackle this by having different prior distributions on the Gaussian Process – which is a Normal distribution – and on the linear weights $\boldsymbol{\beta}$. Our assumption is that the underlying linear model is sparse, encouraging the use of sparse priors. These shrink the linear weights, leading to a value of zero or near zero for the majority of parameters.
Different priors for $\boldsymbol{\beta}$ are investigated and integrated into the algorithm, leading to a Maximum A Posteriori estimator and a Fully Bayesian model. These are trained with Variational Expectation Maximization and Variational Inference respectively.
One focus is set on scalability, meaning that the presented model scales well to the number of samples n, as well as to the dimensionality d of the data.

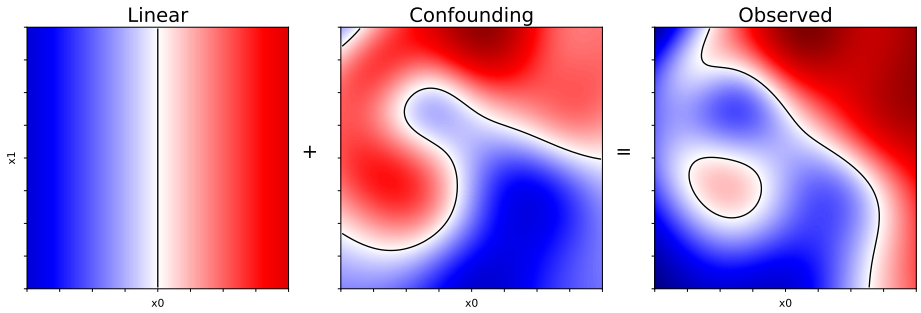

Figure 1.1.: Model for the underlying structure. The linear and the confounding model are added and give us the observable data.

The thesis is structured as follows. Covering the fundamentals, **Chapter 2** treats related work and the components the model is built from.
Chapter 3 bases on this. It builds the model and treats different variants and methods for training the model. Every type of prior and optimization algorithm used, is treated here.
The presented model then is evaluated on different datasets and with several experiments in **Chapter 4**.
Finally, **Chapter 5** concludes the thesis.

2. Fundamentals and Related Work

2.1. Notation

Matrices are denoted with upper case X, while vectors are written with bold letters $\boldsymbol{x}$ and scalars as x. Exceptions to this are functions, like the Loss $\mathcal{L}$.
The number of samples is noted as n, d is the dimensionality, m the number of inducing points and s the batch-size. A dataset consist of the design matrix $X \in \mathbb{R}^{d \times n}$ and the class labels $\boldsymbol{y} \in \mathbb{R}^d$. The variable $i \in [1, n]$ typically iterates over samples and $j \in [1, d]$ does so over the dimensions. The notation $\boldsymbol{\beta}_{-j} \in \mathbb{R}^{d-1}$ is the vector $\boldsymbol{\beta}$ without the j^{th} entry.

2.2. Variable Selection

For our application we need to select relevant features, which also show low correlation to the confounder. The problem of selecting the correct variables and setting the rest to zero has 2^d possible configurations. This means the approaches considering all combinatory possibilities (e.g., [26, 42]) are not scalable to a growing dimensionality.
Another class of approaches is feature ranking [48]. A chosen metric – e.g. the χ^2-norm or Information Gain – calculates feature importance variables that fall below a threshold are eliminated. 'Sure Screening' methods [12], like SAFE [17], can remove features that are guaranteed to be irrelevant, given certain circumstances.

2.3. Sparse Priors

When using a probabilistic model approach, feature selection can be achieved with a sparse prior on the weights $\boldsymbol{\beta}$. This setting is not equivalent to variable selection, since here we estimate continuous weight-values, but features that receive a low weight do not account for the model, essentially eliminating them. There are several approaches on how to design and use a sparse prior. The most prominent is the Laplace Prior, which shrinks all weights to zero. Its logarithm is proportional to the widely used $\ell 1$-norm, which is used for LASSO [44].
A different approach is the Spike and Slab prior [27, 31]. The Spike and Slab prior consists of two distributions, the spike, which is a distribution with low variance at zero and the slab, a heavy tailed distribution. The idea is to assign a high probability to zero weights via the spike, while still allowing non zero weights with the slab. Titsias and Lázaro-Gredilla [46] used a mean field approximation to the different dimensions of

$\boldsymbol{\beta}$ while assuming dependence between the spike and the slab in one dimension. They applied Variational Inference to this model and updated their variational distributions with Expectation Maximization.
Hierarchical priors are a combination of several priors. When using two components, the prior $p(\boldsymbol{\beta}) = \int p(\boldsymbol{\beta}|\boldsymbol{\lambda})p(\lambda)\mathrm{d}\lambda$ is dependent on another variable λ, which in turn has another distribution. Many approaches use a Gaussian normal on $\boldsymbol{\beta}$ with a scale mixture for $\boldsymbol{\lambda}$ leading to $p(\boldsymbol{\beta}|\boldsymbol{\lambda}) \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{\lambda}^2)$. Hierarchical priors are useful for sparsity, because their design allows flexibility; The variance of $\boldsymbol{\beta}$ it can be scaled globally with σ and locally for single β_j with λ_i. With the help of data-augmentation the Laplace distribution can be transformed into a hierarchical prior [11, 19, 32] and the Spike and Slab prior also has hierarchical versions – see [24] for more information.
Apart from the Gaussian distribution, all priors integrated in a Bayesian fashion in this thesis are normal scale mixture models. This has the advantage, that other mixture models can be integrated into the framework with ease. Further scale mixture models are treated in Gelman et al. [16] and Polson and Scott [36].
In this thesis the sparse priors Laplace and Horseshoe [8] are used, while the Spike and Slab prior was not. This is because we found conditionally conjugate models for former two, whereas we did not for the latter.

2.4. Kernels

Kernel matrices K are constructed with kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$, which calculate the similarity or distance between two data-points. Kernels are also used as a model for the covariance in Gaussian Processes. In statistical genetics, similarity kernels model population structures and selection biases, as so-called Kinship matrices [2]. The motivation for the latter usage is, that genetic similarity hints at similar geography for populations. Data-points stemming from the same data source, are subject to the same local confounding influences that might hide the true underlying structure we want to recover.
Our approach uses a Gaussian Process to model confounding variables. In order to represent a covariance function, K needs to be symmetric and positive semi-definite. For two data matrices X and X', we write the matrix of the similarities between individual samples $\boldsymbol{x}_k$ and $\boldsymbol{x}'_l$ as the matrix $K(X, X')$, where $k(\boldsymbol{x}_k, \boldsymbol{x}'_l) = K(X, X')_{k,l}$.
The kernel-function used is a weighted linear combination of a white kernel, a linear kernel and a Radial Basis Function (RBF) kernel. The white kernel is 1 only if $\boldsymbol{x}_k = \boldsymbol{x}'_l$, resulting in a identity matrix, used to model the individual noise for each data-point.
The linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \cdot \boldsymbol{x}'$ is greatly used in statistical genetics. It performs well for modeling the kinship of two subjects, based on their genes. For $\mathbb{E}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{x}'] = 0$ we have $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[\boldsymbol{x}^T \cdot \boldsymbol{x}']$, which shows the suitability of the linear kernel in this task.
The RBF kernel, as our last summand, is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\frac{(\boldsymbol{x} - \boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}')}{2l^2}\right\}$$

with l being the length scale. All kernels have variances, which are the weights they have in the kernel-function.
There are many more kernels like Automatic Relevance Derivation, Matern Kernels, etc., but the kernels presented above form a set of basic kernels used in a standard setting.

2.5. Linear Classification Models

The Generalized Linear Model (GLM) uses a linear model with added i.i.d. noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

$$\hat{y}_{(x)} = g(\boldsymbol{x}^T \boldsymbol{\beta} + \epsilon) \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$ is the data-point, $\boldsymbol{\beta} \in \mathbb{R}^{d \times 1}$ are the linear weights and the noise variable $\boldsymbol{\epsilon} \in \mathbb{R}$. The model is a linear model if the function $g(\cdot)$ is linear.
For binary classification we can use $\hat{y}_{(x)}$ as a latent function and apply a response function $r(\cdot)$ to obtain class probabilities $p(y|\boldsymbol{x})$. A response function converts its input from $[-\infty, \infty]$ to a valid probability in $[0, 1]$. The probability for the class-label of $\boldsymbol{x}$ being one then is $p(y = 1|\boldsymbol{x}) = r(\hat{y}_{(x)})$.
Possible response functions are probit $\phi(\cdot)$ or logit $\sigma(\cdot)$ – also called Log Odds – function. Where the former is defined as $\phi(x) = \int_{-\infty}^{x} \mathcal{N}(x'|0, 1)dx'$ and the latter as $\sigma(x) = \frac{1}{1-\exp(-x)}$. Both functions are cumulative distribution functions of distributions – the Normal and the logistic distribution – which are symmetric around zero. This gives us the property $r(-\hat{y}) = 1 - r(\hat{y})$ and $p(y|\boldsymbol{x}) = r(y \cdot \hat{y}_{(x)})$, assuming classes +1 and -1.

2.5.1. Linear Mixed Models

Linear Mixed Models (LMM) [20] model population structure in the data and are a method for analyzing data that involve random effects. The LMM assume that,

$$\boldsymbol{y} = X^T \boldsymbol{\beta} + \boldsymbol{f} + \boldsymbol{\epsilon}$$

where $\boldsymbol{f} \in \mathbb{R}^n$ models random fixed effects. Confounding influences can be modeled by $\boldsymbol{f}$ [39], which we do in this thesis.
This model can be seen as a problem of signal recovery, where the latent weights and noise components have to be separated.
Mandt et al. [29] use a Probit LMM, modeling $\boldsymbol{f}$ with Similarity Kernels K. This means that the confounding variables $\boldsymbol{f}$ are modeled with the distribution $\mathcal{N}(0, K)$. For notational simplicity K also models the independent noise $\boldsymbol{\epsilon}$. In order to enforce sparsity in the linear weights a $\ell 1$-norm is added to the negative marginal likelihood, which is optimized with Expectation Propagation. This approach does not scale well to the number of samples.
This thesis expands upon the work of Mandt et al. [29]. This is done by using a

Sparse Gaussian Process for modeling $\boldsymbol{f} + \boldsymbol{\epsilon}$ (cf. Section 2.6.1), a Pòlya-Gamma data-augmentation (cf. Section 2.7) for analytical tractability and finally, further priors are integrated in Section 3.2 for achieving sparsity in $\boldsymbol{\beta}$.

2.6. Gaussian Process Classification

Gaussian Processes(GP) are ideal for modeling confounders, as their kernel models the correlations between data-points. Rasmussen [40] describes the Gaussian Process as "*a collection of random variables, any finite number of which have a joint Gaussian distribution*". This means that it gives us a predictive Gaussian distribution at every test point. This means that we always have an estimate of the variance of the prediction. For simplicity we assume a mean of zero, then the GP is written as:

$$f(\boldsymbol{x}) \sim \mathrm{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}^*)) \tag{2.2}$$

Where k is the covariance function. Without any assumed noise, the predicted label f^* for test point $\boldsymbol{x}^*$ and observed data-points $(X, \boldsymbol{f})$ is

$$\begin{bmatrix} f^* \\ \boldsymbol{f} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\boldsymbol{x}^*, \boldsymbol{x}^*) & K(\boldsymbol{x}^*, X) \\ K(X, \boldsymbol{x}^*) & K(X, X) \end{bmatrix}\right). \tag{2.3}$$

In order to calculate this distribution one has to invert the covariance matrix, which leads to a runtime complexity of $O(n^3)$. This means, that in the standard configuration GPs do not scale to big datasets. In the next sections, approaches on how to make Gaussian Process scalable are be explained.

2.6.1. Sparse Gaussian Process Classification

Sparse Gaussian Processes try to lessen the time complexity by approximating the dataset with fewer points. This is a very common approach for making GPs scalable [10, 21, 38, 43, 45]. When using inducing points, the covariance matrix is approximated with a lower rank matrix. Apart from using inducing points, there are further approaches, like the Sparse Spectrum GP, where the covariance function is approximated by using a sparse representation in the frequency space [15, 38].
This thesis bases its model upon the work by Hensman et al. [22], which allows the use of Stochastic Variational Inference (SVI). Inducing points Z are in the same domain as the data-points and each inducing point $\boldsymbol{z}_i$ has a corresponding inducing variable u_i. Instead of n data-pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, m inducing pairs $\{(\boldsymbol{z}_i, u_i)\}_{i=1}^m$ are used for building the covariance matrix. For regression Hensman et al. [21] defined the predictive distribution as

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{f}) &\sim \mathcal{N}\left(\boldsymbol{f}|\sigma^{-1}I\right) \\ p(\boldsymbol{f}|\boldsymbol{u}) &\sim \mathcal{N}\left(K_{n,m}K_{m,m}^{-1}\boldsymbol{u}, \tilde{K}\right) \\ p(\boldsymbol{u}) &\sim \mathcal{N}\left(\boldsymbol{0}, K_{m,m}\right). \end{aligned} \tag{2.4}$$

Where σ is the assumed noise over the latent GP model. $K_{m,m}$ is the covariance matrix of the inducing points Z. $K_{n,m}$ is the covariance matrix between the points X of the data-set and Z and $\tilde{K} = K_{n,n} - K_{n,m}K_{m,m}^{-1}K_{m,n}$ is the difference between the original covariance function and its approximation. In this configuration all probabilities are analytically tractable and it is possible to apply Coordinate Ascent Variational Inference (CAVI) [4]. The inference algorithm optimizes the distribution of $\boldsymbol{u}$, such that the inducing variables best represent the data.
When using a Heaviside step function for classification and assuming noisy prediction in Equation 2.4, the model is equivalent to the probit function $\phi(\cdot)$. This is what Hensman et al. [22] did to expand their model to classification. The probabilistic model thus is similar to the regression case with only Equation 2.4 being replaced by

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \phi(y_i f_i). \tag{2.5}$$

In this model CAVI cannot be applied, since the Probit function can only be approximated numerically. Because of this, the model is trained with standard gradient ascent.

2.7. Pòlya-Gamma Logit Data Augmentation

The data augmentation with a Pòlya-Gamma variable, as done by Wenzel et al. [47], speeds up Sparse Gaussian Process Classification. Even though the time complexity of the algorithm stays the same, having analytically tractable natural gradients updates, instead of numerically approximated gradients, leads to decreases in runtime to up to three orders of magnitude [47]. This is because the individual calculations of the gradient are faster and mainly because natural gradients need less steps to converge to an optimum.
Data-augmentation for logit functions with Pòlya-Gamma was introduced by Polson et al. [37]. This is why instead of the Probit function, as done by Hensman et al. [22] or Mandt et al. [29], we use the Log-Odds ratio for our response function.
The moment generating function for the Pòlya-Gamma distribution (PG) , with $\omega \sim \mathrm{PG}(b, 0)$ for $b > 0$ is defined as

$$\mathbb{E}_{\mathrm{PG}(\omega|b,0)}\left\{\exp(-\omega t)\right\} = \frac{1}{\cosh^b(\sqrt{\frac{t}{2}})}. \tag{2.6}$$

This is the Laplace transform of an infinite convolution of gamma distributions [37]. The more general $\mathrm{PG}(b, c)$ is reached by exponential tilting of $\mathrm{PG}(b, 0)$. The distribution then is

$$PG(\omega|b, c) = \frac{\exp\left(-\frac{c^2}{2}\omega\right) PG(\omega|b, 0)}{\mathbb{E}_{\mathrm{PG}(\omega|b,0)}\left\{\exp(-\frac{c^2}{2}\omega)\right\}}. \tag{2.7}$$

where the denominator normalizes the function. By using both equations we can compute the moment generating function for $\mathrm{PG}(b, c)$:

$$\mathbb{E}_{\mathrm{PG}(\omega|b,c)}\left\{\exp(-\omega t)\right\} = \frac{\cosh^b(\frac{c}{2})}{\cosh^b\left(\sqrt{\frac{c^2/2+t}{2}}\right)},$$

The first moment is calculated by evaluating the first derivative of this function at $t = 0$. This yields [37]

$$\mathbb{E}_{PG(\omega|b,c)}[\omega] = \frac{b}{2c}\tanh(\frac{c}{2}). \tag{2.8}$$

After establishing the necessary information for the PG distribution, we can use it to augment the logit function $\sigma(\cdot)$

$$\begin{aligned}\sigma(z_i) &= \frac{1}{1+\exp(-z_i)} \\ &= \frac{\exp(\frac{z_i}{2})}{2\cosh(\frac{z_i}{2})}.\end{aligned}$$

We can introduce the Pòlya-Gamma variable by using Equation 2.6 with $p(\omega_i) = \mathrm{PG}(\omega_i|1, 0)$:

$$\sigma(z_i) = \frac{1}{2}\int \exp\left(\frac{z_i}{2} - \frac{z_i^2}{2}\omega_i\right) p(\omega_i)\mathrm{d}\omega_i.$$

Our model uses $p(y_i|f_i) = \sigma(y_i f_i)$ and with the labels $y_i \in \{-1, 1\}$. Because $y_i^2 = 1$, we can ignore the squared labels and write the terms as vectors

$$\sigma(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\omega}) \propto \exp(\frac{1}{2}\boldsymbol{y}^T\boldsymbol{f} - \frac{1}{2}\boldsymbol{f}^T\Omega\boldsymbol{f}) \tag{2.9}$$

where $\Omega = \mathrm{diag}(\boldsymbol{\omega})$.
This augmentation helps us transform the likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\omega})$ of the classification problem into a regression problem with a Gaussian likelihood, more information can be found in Polson et al. [37]. Thus we can make use of Coordinate Ascent Variational Inference, which is explained in the following.

2.8. Variational Inference

Variational Inference (VI) [5] is a method that approximates the exact posterior $p(\boldsymbol{\theta}|X)$ of the latent variables $\boldsymbol{\theta}$, given the observations X. The optimal approximate density $q^*(\boldsymbol{\theta})$, is part of a previously specified family of densities $\mathcal{D}$. This includes the decision on the dependencies between the latent variables. The function that VI optimizes is

$$q^*(\boldsymbol{\theta}) = \underset{q(\boldsymbol{\theta})\in\mathcal{D}}{\arg\min}\ KL(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X).$$

As can be seen, the quality of the approximation to the likelihood $p(\boldsymbol{\theta}|X)$ is measured with the Kullback Leibler Divergence (KL). The KL is a non-negative measure, that only becomes zero if the two densities are identical. It is defined as

$$KL(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|X)}\right].$$

Expanding the terms, one can see that the KL depends on the evidence $p(X)$, which is constant, but not computationally feasible.

$$KL(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X) = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta}, X)\right]}_{=-\text{ELBO}} + \log p(X)$$

By maximizing the Evidence Lower Bound (ELBO), which is the negative $KL(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X))$ plus the constant $p(X)$, we can minimize the divergence between the variational distribution $q(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|X)$ without calculating $p(X)$.
More intuitively the ELBO can be written as

$$\text{ELBO} = \mathbb{E}[\log p(X|\boldsymbol{\theta})] - KL(q(\boldsymbol{\theta})||p(\boldsymbol{\theta})).$$

Here we can see that it consists of a term that maximizes the expected log likelihood of the data while the second term accounts for the prior.
Variational parameters that are independent of the number of samples in the data are called global parameters, while the ones that cover a single data points are called local parameters.

2.8.1. Coordinate Ascent Variational Inference

Coordinate Ascent Variational Inference solves the optimization problem set with the ELBO, by iteratively updating the variational distributions, assuming all other distributions to be fixed. The optimal distribution $q^*(\theta_i)$ for maximizing the ELBO is found with the help of the full conditional on θ_i

$$q^*(\theta_i) \propto \exp \mathbb{E}_{q(\boldsymbol{\theta}_{-i})}\left[\log p(\theta_i|y, \boldsymbol{\theta}_{-i})\right].$$

This means that the optimization can be done without calculating the ELBO. Because of this, the development of a Coordinate Ascent Variational Inference (CAVI) algorithm can be comparatively easy. In this thesis, these updates are not always used, because we use a lower bound for the ELBO.
In our approach all conditional densities are in a exponential family. This family can be formalized as:

$$p(\theta|\gamma) = h(\theta) \exp\left\{\eta(\gamma)^T t(\theta) - a(\eta(\gamma))\right\},$$

where h is called the base measure, a is the log normalizer, η the natural parameter and t a sufficient statistic. Every probability that fits this form is part of the exponential

family. The multivariate Gaussian $(\mathcal{N})(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$, for example, can be brought into the adequate form with

$$h = (2\pi)^{D/2} \qquad \boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{x}\boldsymbol{x}^T)^T$$
$$\boldsymbol{\eta} = (\Sigma^{-1}\boldsymbol{\mu} - 1/2\Sigma^{-1}) \qquad a = \log|\Sigma| - 1/2\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu}.$$

Models that only contain exponential family conditionals are also known as conditionally conjugate models.

CAVI algorithms iterate through for the different variational distributions, by first calculating the optimal local parameters and then sequentially the global ones. A problem with these updates is, that for large datasets it takes a long time to calculate one step, because this means taking the whole dataset into account. Stochastic Variational Inference is a way out of this problem.

2.8.2. Stochastic Variational Inference

Hoffman et al. [23] proved that for conditionally conjugate models, the CAVI updates are the same as natural gradient updates. Natural gradient updates always give the optimal gradient by taking the distribution into account, this means that they generally find an optimum in less steps. Robbins and Monro [41] showed that, as long as the series of step size diverges, while the series of its square converges, noisy unbiased gradients optimize the objective function. This means that, if we use noisy natural gradients, the ELBO will still converge to an optimum. In order to adapt to mini-batches, we use

$$\mathbb{E}[\log p(X|\theta)] = \sum_{i=1}^{n} \mathbb{E}[\log p(\boldsymbol{x}_i|\theta)] = n\mathbb{E}[\log p(\boldsymbol{x}|\theta)].$$

When using mini-batches, we can calculate the expectation over the small subset, as an approximation to the expectation of the full dataset. Simply put we multiply every term that contains a sum over the samples with $\frac{n}{s}$, where s is the size of the mini-batch. This leads to noisy un-biased updates. These updates θ'_t, at timestep t, are applied to the current estimates of the variational parameters θ_t with the step size r_t as follows:

$$\theta_{t+1} = (1 - r_t)\theta_t + r_t\theta'_t$$

In this thesis the step-size decays with $r_t = t^{-\mathrm{lr}}$, where $\mathrm{lr} \in (0.5, 1]$ is the initially set learning rate.

3. Model

With the fundamentals covered, we can move on to the derivation of the model and algorithm. In this chapter the treatment of $\boldsymbol{\beta}$ is left for last. The other terms are addressed first, because the prior $p(\boldsymbol{\beta})$ can be integrated in different ways. To this end we first assemble the proposed model from the individual parts treated in Section 2. The integration of $p(\boldsymbol{\beta})$ follows in Section 3.2.

3.1. Correlated Noise Classification Models

As before we assume a logit classification model with correlated noise.

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\beta}) &\sim \prod_i \sigma(y_i(\boldsymbol{x}_i^T \boldsymbol{\beta} + f_i)) \\ \boldsymbol{f} &\sim GP(0, K(X, X)). \end{aligned} \tag{3.1}$$

Where $\sigma(\cdot)$ is the logit function and $\boldsymbol{f}$ represents the correlated noise. This logit model is augmented with a Pólya-gamma distribution [37] and the full conditional of $\boldsymbol{y}$ becomes a Gaussian distribution:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\beta}, \boldsymbol{\omega}) &\propto \exp\left\{\frac{1}{2}\boldsymbol{y}^T(X^T\boldsymbol{\beta} + \boldsymbol{f}) - \frac{1}{2}(X^T\boldsymbol{\beta} + \boldsymbol{f})^T\Omega(X^T\boldsymbol{\beta} + \boldsymbol{f})\right\} \\ \omega_i &\sim \mathrm{PG}(1, 0). \end{aligned} \tag{3.2}$$

When modeling $\boldsymbol{f}$ with a standard GP, this model does not scale to a growing number of data points. We follow Hensman et al. [21] by approximating the distribution for $\boldsymbol{f}$ using inducing points and a variational distribution (cf. Sec 2.6.1).

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{u}) &= \mathcal{N}(\boldsymbol{f}|K'_{n,m}\boldsymbol{u}, \tilde{K}) \\ p(\boldsymbol{u}) &= \mathcal{N}(\boldsymbol{u}|0, K_{m,m}^{-1}), \end{aligned}$$

where $K'_{n,m} = K_{n,m}K_{m,m}^{-1}$ and we use $K'_{m,n} = {K'_{n,m}}^T$ in the following. The model from Equation 3.2 is similar to the XGPC model of Wenzel et al. [47], with the difference that the input to $\sigma(\cdot)$ has the added term of the linear weights $X^T\boldsymbol{\beta}$. Coupled with sparse priors this added term is designed to model the few strong signals in $\boldsymbol{\beta}$.
We can adapt this model to variables, we do not want or cannot use for prediction on unseen data. If, for example, we have sensitive data, that we do not use for classification, we do not use these for the linear model. This 'side information' is only used for computing the kernel of the GP.

This way, we still use all data for training. Because the confounder is modeled by the GP part, the linear classifier can predict with less correlation to the sensitive variables. Experimental results on this are shown in Section 4.8.2.

3.1.1. Evidence Lower Bound

The model is optimized with Variational Inference (cf. Section 2.8), which means that we want to maximize the evidence $p(\boldsymbol{y})$. Because we approximate $\boldsymbol{f}$ with inducing points, we cannot compute the ELBO directly. Like Hensman et al. [21] we use Jensen's Inequality to get a analytically tractable lower bound of the log likelihood:

$$\begin{aligned}\log(p(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})) &= \log \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})}\left[p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] \\ &\geq \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right].\end{aligned}$$

With this we can lower bound the evidence $p(\boldsymbol{y})$ by

$$\begin{aligned}\log p(\boldsymbol{y}) &\geq \text{ELBO} \\ &= \log \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})}\left[\log \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})}\left[p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right]\right] - KL(q(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})||p(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})) \\ &\geq \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] - KL(q(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})||p(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})).\end{aligned}$$

Using a structured mean-field approach, we assume independence between $\boldsymbol{u},\boldsymbol{\omega}$ and $\boldsymbol{\beta}$, as well as between the local parameters ω_i. Factorizing $q(\boldsymbol{u},\boldsymbol{\omega},\boldsymbol{\beta})$ leads to $q(\boldsymbol{u})q(\boldsymbol{\beta})\prod q(\omega_i)$:

$$\begin{aligned}\mathcal{L} := &\mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] \\ &- KL(q(\boldsymbol{u})||p(\boldsymbol{u})) - KL(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - KL(q(\boldsymbol{\beta})||p(\boldsymbol{\beta})).\end{aligned} \tag{3.3}$$

In the following we note the first and second moments of the global variational distributions with $\boldsymbol{m}$ and S, e.g. $\mathbb{E}_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}] = \boldsymbol{m}_\beta$ and $\text{Var}_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}] = S_\beta$, and find the form of the lower bound for the log of the expected likelihood.

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] &= \frac{1}{2}\mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\Big[y^T(X^T\boldsymbol{\beta}+\boldsymbol{f}) - (X^T\boldsymbol{\beta}+\boldsymbol{f})^T\Omega(X^T\boldsymbol{\beta}+\boldsymbol{f})\Big] \\ &= \frac{1}{2}\Big[y^T(X^T\mathbb{E}_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}] + \mathbb{E}_{q(\boldsymbol{f})}[\boldsymbol{f}]) - \mathbb{E}_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}^T X^T\Theta X^T\boldsymbol{\beta}^T] \\ &\quad - 2\mathbb{E}_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}^T]X\Theta\mathbb{E}_{q(\boldsymbol{f})}[\boldsymbol{f}] - \mathbb{E}_{q(\boldsymbol{f})}[\boldsymbol{f}^T\Theta\boldsymbol{f}]\Big]\end{aligned}$$

where we use the notations $\Theta = \mathbb{E}_{q(\boldsymbol{\omega})}[\Omega]$ and $q(\boldsymbol{f}) = \int q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})\mathrm{d}\boldsymbol{u}$. Like Hensman et al. [22] and Wenzel et al. [47] we assume a Normal distribution $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}_u, S_u)$ and in order to find the last summand we apply $p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(K'_{n,m}\boldsymbol{u}, \tilde{K})$

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{f})}[\boldsymbol{f}^T\Theta\boldsymbol{f}] &= \mathbb{E}_{q(\boldsymbol{u})}\left[\mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})}[\boldsymbol{f}^T\Theta\boldsymbol{f}]\right] \\ &= \mathbb{E}_{q(\boldsymbol{u})}[\boldsymbol{u}^T K'^T_{m,n}\Theta K'_{n,m}\boldsymbol{u} + Tr(\Theta\tilde{K})] \\ &= \boldsymbol{m}_u^T K'^T_{m,n}\Theta K'_{n,m}\boldsymbol{m}_u + Tr(K'^T_{m,n}\Theta K'_{m,n}S_u) + Tr(\Theta\tilde{K}).\end{aligned}$$

When setting this result back in, we can calculate the expected likelihood

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] = \frac{1}{2}\Big[&\boldsymbol{y}^T(X^T\boldsymbol{m}_\beta + \hat{\boldsymbol{m}}_f) - \boldsymbol{m}_\beta^T X\Theta X^T\boldsymbol{m}_\beta - Tr(X^T\Theta X S_\beta)\\ &- \hat{\boldsymbol{m}}_f{}^T\Theta\hat{\boldsymbol{m}}_f - Tr(\Theta\tilde{K}) - Tr(K'_{m,n}\Theta K'_{m,n}S_u)\\ &- 2\boldsymbol{m}_\beta^T X\Theta\hat{\boldsymbol{m}}_f\Big],\end{aligned} \tag{3.4}$$

where we note $K'_{n,m}\boldsymbol{m}_u$ as $\hat{\boldsymbol{m}}_f$.
In order to calculate the full ELBO, we still need the Kullback Leibler divergences for the different variational parameters, which follow in the next sections.

3.1.2. Variational Distributions

The following sections first introduce the variational distributions and updates for $\boldsymbol{u}$ and $\boldsymbol{\omega}$. The treatment of the different priors for $\boldsymbol{\beta}$ are then found in Section 3.2.
The variational distribution for $q(\boldsymbol{u})$ and $q(\boldsymbol{\omega})$ depend on $\boldsymbol{m}_\beta$ and S_β, but otherwise the distributions stay unchanged.

Global Parameters

Because we use Jensen's Inequality to lower bound the ELBO, it is not straightforward to use the CAVI update rules for $\boldsymbol{u}$ and we recover the optimal parameters by using derivatives. We assume a Normal Gaussian for $q(\boldsymbol{u})$.
The Kullback-Leibler Divergence $KL(q(\boldsymbol{u})||p(\boldsymbol{u}))$ between the two multivariate Gaussians $q(\boldsymbol{u})$ and $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, K_{m,m}^{-1})$ is

$$KL(q(\boldsymbol{u})||p(\boldsymbol{u})) \overset{c}{=} \frac{1}{2}\left(Tr(K_{m,m}^{-1}S_u) + \boldsymbol{m}_u^T K_{m,m}^{-1}\boldsymbol{m}_u + \log\frac{|K_{m,m}|}{|S_u|}\right). \tag{3.5}$$

With this information and Equation 3.4 we can compute the optimal $q^*(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}_u^*, S_u^*)$, by finding the gradients with respect to the parameters and setting them to zero.

$$\begin{aligned}S_u^* &= \left(K_{m,m}^{-1} + K'_{m,n}\Theta K'_{n,m}\right)^{-1}\\ \boldsymbol{m}_u^* &= S_u^*(\frac{1}{2}K'_{m,n}\boldsymbol{y} - K'_{m,n}\Theta X^T\boldsymbol{m}_\beta)\end{aligned}$$

The exact derivation of these results can be found in Appendix A.1. As in Wenzel et al. [47], these updates are equivalent to natural gradient updates.

Local Parameters

The full conditional $p(\omega_i|\boldsymbol{\omega}_{-i}, \boldsymbol{f}, \boldsymbol{u}, \boldsymbol{\beta}, \boldsymbol{y})$ on ω_i has the form.

$$\begin{aligned}\mathbb{E}_{q(-\omega_i)}\left[\log p(\omega_i|\boldsymbol{y},\boldsymbol{\omega}_{-i},\boldsymbol{f},\boldsymbol{u},\boldsymbol{\beta})\right] &\propto \mathbb{E}_{q(-\omega_i)}\left[\log p(\boldsymbol{y}|\omega_i,\boldsymbol{\omega}_{-i},\boldsymbol{f},\boldsymbol{u},\boldsymbol{\beta})p(\omega_i)\right]\\ &\propto \left(-\frac{1}{2}\mathbb{E}_{q(\boldsymbol{\beta})q(\boldsymbol{f})}\left[(\boldsymbol{x}_i^T\boldsymbol{\beta} + f_i)^2\right]\omega_i\right) + \log PG(\omega_i|1,0)\end{aligned} \tag{3.6}$$

where $q(-\omega_i)$ is $q(\boldsymbol{\omega}_{-i}, \boldsymbol{\beta})q(\boldsymbol{f})$. When looking at Equation 3.6, we can see that its exponential is a Pòlya-gamma distribution $PG(\omega_i|1, c_i)$, as defined in Equation 2.7. As Wenzel et al. [47] we use a variational distribution which has the same form:

$$q(\omega_i) = PG(\omega_i|1, c_i).$$

The Kullback-Leibler Divergence between $q(\omega_i)$ and $p(\omega_i)$ is found with

$$\begin{aligned} KL(q(\omega_i)||p(\omega_i)) &= \mathbb{E}_{q(\omega_i)}\left[\log q(\omega_i) - \log p(\omega_i)\right] \\ &= \mathbb{E}_{q(\omega_i)}\left[\log\cosh(\frac{c_i}{2}) - \frac{c_i^2}{2}\omega_i + \log PG(\omega_i|1,0) - \log PG(\omega_i|1,0)\right] \\ &= \log\cosh(\frac{c_i}{2}) - \frac{c_i^2}{2}\mathbb{E}_{q(\omega_i)}\left[\omega_i\right] \\ &= \log\cosh(\frac{c_i}{2}) - \frac{c_i}{4}\tanh\left(\frac{c_i}{2}\right). \end{aligned}$$

The Pòlya-gamma distributions cancel out and we can calculate the ELBO without the need of the actual calculation of $PG(\omega_i|1, 0)$.
By using Equation 2.8, we can calculate the expectancy of ω_i:

$$\theta_i := \mathbb{E}[\omega_i] = \frac{1}{2c_i}\tanh(\frac{c_i}{2})$$

We obtain the optimal $q^*(\omega_i)$ by using the gradient and setting it to zero leading to the updates

$$c_i^* = \sqrt{(\hat{\boldsymbol{m}}_{f_i} + \boldsymbol{x}_i^T\boldsymbol{m}_\beta)^2 + \boldsymbol{x}_i^T S_\beta \boldsymbol{x}_i + \tilde{K}_{i,i} + \boldsymbol{k}'_{i,m} S_u \boldsymbol{k}'_{m,i}}, \tag{3.7}$$

where $\boldsymbol{x}_i$ is the i^{th} column of X, i.e. the i^{th} sample in the dataset and $\boldsymbol{k}_{i,m}$ is the i^{th} row of $K_{n,m}$, which is the vector of covariances between sample i with the m inducing points. Further we note $\boldsymbol{k}_{i,m}K_{m,m}^{-1}$ as $\boldsymbol{k}'_{i,m}$, $\boldsymbol{k}'^T_{i,m}$ as $\boldsymbol{k}'_{m,i}$.
The exact derviation can be found in Appendix A.2

3.2. Priors on $\boldsymbol{\beta}$

First we present the Maximum A Posteriori approximations to $p(\boldsymbol{\beta})$ for the Gaussian and Laplace prior and then treat the fully Bayesian approaches for the Gaussian, Laplace and Horseshoe prior.

3.2.1. Maximum A Posteriori estimators

For this approximation we optimize the MAP estimate by using the Maximum Likelihood Estimate as the cost function. This is then maximized with Variational Expectation Maximization (VEM). In the following, we first present the VEM algorithm, followed by the models with the corresponding priors.

Variational Expectation Maximization

In this setting we optimize the MAP estimate for the posterior $p(\boldsymbol{\beta}|\boldsymbol{y})$, by iteratively updating the variational distributions, estimating the probability (E-step) and then maximizing it (M-step). We use VEM, where in the E-step we approximate

$$p(\boldsymbol{\beta}|\boldsymbol{y}) \approx \mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})}[p(\boldsymbol{\beta}|\boldsymbol{f}, \boldsymbol{\omega}, \boldsymbol{y})]$$

using the variational distributions for $\boldsymbol{\omega}$ and $\boldsymbol{u}$ and in the M-step we find $\boldsymbol{\beta}^*$ such that

$$\begin{aligned}\boldsymbol{\beta}^* &= \arg\max_{\boldsymbol{\beta}} \mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})}[p(\boldsymbol{\beta}|\boldsymbol{f}, \boldsymbol{\omega}, \boldsymbol{y})] \\ &= \arg\max_{\boldsymbol{\beta}} \log \mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})}[p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\omega}, \boldsymbol{\beta})] + \log p(\boldsymbol{\beta}).\end{aligned}$$

This model only differs to the fully Bayesian approach, in that we have a MAP estimator for $\boldsymbol{\beta}$. The other latent variables are still optimized with variational distributions.
When seen as a function of $\boldsymbol{\beta}$, this model corresponds to an optimization problem with a $\ell 2$ or $\ell 1$ norm, for the Normal Gaussian and Laplace priors respectively:

$$\hat{\mathcal{L}}_{\boldsymbol{\beta}} = \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{\omega})}\left[\log p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\omega}, \boldsymbol{\beta})\right] + \log p(\boldsymbol{\beta}) \tag{3.8}$$

$$\propto \frac{1}{2}\Big(\boldsymbol{y}^T X^T \boldsymbol{\beta} - 2\hat{\boldsymbol{m}}_f^T \Theta X^T \boldsymbol{\beta} - \boldsymbol{\beta}^T X \Theta X^T \boldsymbol{\beta}\Big) - \frac{1}{2\sigma_\beta}||\boldsymbol{\beta}||_p^p + const. \tag{3.9}$$

where we note the approximate log likelihood with $\mathcal{L}(\boldsymbol{\beta})$. The variance of $p(\boldsymbol{\beta})$ is σ_β. For a Gaussian prior the variable p equals 2 and 1 for Laplace.
When applying the **CAVI updates** to the variational distributions $q(\boldsymbol{u})$ and $q(\boldsymbol{\omega})$, the are the same as in Section 3.1.2 with $S_\beta = 0$ and the mean $\boldsymbol{m}_\beta$ being equal to the MAP estimate of $\boldsymbol{\beta}$. This is because the point estimate for $\boldsymbol{\beta}$ can also be represented by a Dirac delta function around $\boldsymbol{\beta}$, which is equivalent to a Normal distribution with a variance tending to zero and a mean at $\boldsymbol{\beta}$.
After this update, the **estimation** of the loss is done with Equation 3.8.
For both priors, we can either **maximize** $\boldsymbol{\beta}$ directly or β_j individually, using coordinate ascent. For the latter approach, one iteration through all dimensions of $\boldsymbol{\beta}$ has a linear complexity in d, because it circumvents the inversion of a $d \times d$ matrix ($\mathcal{O}(d^3)$). The downside is that coordinate ascent needs several iterations to converge. This approach has useful properties when using the Laplace distribution. Further details on that can be found in Section 3.2.1.
The fact that we do not have to estimate S_β does not reduce the complexity of the algorithm.

Gaussian prior

When using Equation 3.9 with a Gaussian prior, the optimal $\boldsymbol{\beta}^*$ is found by setting the derivate to zero:

$$\frac{\partial \hat{\mathcal{L}}_{\boldsymbol{\beta}}}{\partial \boldsymbol{\beta}} = \frac{1}{2} X\boldsymbol{y} - X\Theta\hat{\boldsymbol{m}}_f - X\Theta X^T\boldsymbol{\beta} - \frac{1}{\sigma_\beta}\sum \beta_j$$
$$\boldsymbol{\beta}^* = \left(\frac{1}{\sigma_\beta} I + X\Theta X^T\right)^{-1} (\frac{1}{2} X\boldsymbol{y} - X\Theta\hat{\boldsymbol{m}}_f)$$

Note that in this case we can use the Woodbury Identity to reduce the complexity of inversing the matrix. Nevertheless we still have a minimum complexity of $O(d^2)$, because the matrix has $d \times d$ dimensions.
In practice this approximation is only used as a sanity check and has no practical relevance.

Laplace prior

While it is possible to represent it as an infinite mixture of Gaussians (cf. Section 3.2.2), it is not possible to develop a CAVI algorithm for the standard Laplace distribution. This is because of the absolute value in the exponent:

$$p(\beta_j) = \frac{1}{2\sigma_\beta} \exp\left\{-\frac{|\beta_j|}{\sigma_\beta}\right\}$$

When using the MAP, the optimization problem is equivalent to the LASSO setting [44]; We have a quadratic function coupled with a ℓ1-norm regularization term (cf. Equation 3.9).
As mentioned above it can be useful to use coordinate ascent for updating the individual weights instead of taking the direct approach.

Coordinate Ascent

As before we look at the gradient with respect to β_j

$$\frac{\partial \hat{\mathcal{L}}_{\boldsymbol{\beta}}}{\partial \beta_j} = \frac{1}{2}\boldsymbol{x}_j\boldsymbol{y} - \boldsymbol{x}_j\Theta\hat{\boldsymbol{m}}_f - \boldsymbol{x}_j\Theta\boldsymbol{x}_j^T\beta_j - \boldsymbol{x}_j\Theta X_{-j}\boldsymbol{\beta}_{-j} + \frac{\partial}{\partial \beta_j}\log p(\beta_j)$$
$${\beta_j}^* = \frac{\frac{1}{2}\boldsymbol{x}_j\boldsymbol{y} - \boldsymbol{x}_j\Theta\hat{\boldsymbol{m}}_f - \boldsymbol{x}_j\Theta X_{-j}\boldsymbol{\beta}_{-j} + \frac{\partial}{\partial \beta_j}\log p(\beta_j)}{\boldsymbol{x}_j\Theta\boldsymbol{x}_j^T}$$

Here we can make use of the Soft-Threshold operator $\mathrm{ST}(\cdot)$, similar to Friedman et al. [13]. Optimal β_j^* then is:

$$\beta_j^* = \frac{\mathrm{ST}_{\sigma^{-1}}\left(\frac{1}{2}\boldsymbol{x}_j\boldsymbol{y} - \boldsymbol{x}_j\Theta\hat{\boldsymbol{m}}_f - \boldsymbol{x}_j\Theta X_{-j}\boldsymbol{\beta}_{-j}\right)}{\boldsymbol{x}_j\Theta\boldsymbol{x}_j^T}.$$

The Soft-Threshold operator makes use of the subgradients for the absolute value.
For a convex function h, subgradients always exist. If h is differentiable, the subgradient is the gradient $\frac{\partial h(x)}{\partial x}$. If no gradient exists then the subgradient g is the set of elements g_k such that

$$h(y) \geq h(x) + g_k \cdot (y - x)$$

This obviously holds for g, where h is differentiable. Applied to the absolute value $h(x) = \lambda|x|$, the subgradient at $x = 0$ is $\mathrm{g} = \{g_k | g_k \in [-\lambda, \lambda]\}$. The Soft-Threshold then is defined as

$$ST_\lambda(x) = \begin{cases} x - \lambda & \text{if } x > \lambda \\ 0 & \text{if } -\lambda \leq x \leq \lambda \\ x + \lambda & \text{if } x < -\lambda \end{cases} \tag{3.10}$$

For the $\ell 1$ norm, this always leads to an optimal x^*. We can proof this optimality by ensuring that the gradient $\frac{\partial h(x^*)}{\partial x} = 0$.
For the first and last case of 3.10, the result of $\mathrm{ST}(\cdot)$ is the same as using the gradient of $|x|$. Thus trivially the optimality is fulfilled.
For the middle case – which leads to $ST_\lambda(x) = 0$ – there exists a subgradient g_i with $-\lambda \leq g_i \leq \lambda$ such that $0 = x - g_i$. This concludes the proof.
The runtime until converge can be reduced by using an active set strategy for choosing the elements of $\boldsymbol{\beta}$ instead of iteratively updating all. An example for such a strategy can be found in Friedman et al. [14]. Using screening (e.g. [17]) we can possibly eliminate β_j's that are guaranteed to be zero. This is an effective measure for leveraging the sparsity of $\boldsymbol{\beta}$. Using mini-batches however, this rarely yields effect, because the signals are far noisier and in one single batch less weights are set to 0. Thus interestingly M-steps can take longer to converge for a single mini-batch, than for the whole dataset. Further fine tuning of the coordinate ascent for $\boldsymbol{\beta}$ is certainly possible, but this thesis is not the focus of this thesis.
When calculating $\boldsymbol{\beta}^*$ directly, coordinate ascent is not possible and gradient ascent can be unstable because of the absolute value. One common approach here is to use Alternating Direction Method of Multipliers (ADMM) [6].

Alternating Direction Method of Multipliers

In the presence of more than one optimization term, the ADMM approach splits the function into two distinct objectives with own variables and then lets an augmented

Lagrangian optimize for the two variables to be equal. [7]. ADMM can only be used for a convex problem, which is the case here.
The function is split as follows:

$$\max f(\boldsymbol{\beta}) - g(\hat{\boldsymbol{\beta}}) \quad \text{s.t. } \boldsymbol{\beta} - \hat{\boldsymbol{\beta}} = 0.$$

The augmented Lagrangian can then be written as

$$\tilde{\mathcal{L}}_\beta = f(\boldsymbol{\beta}) - g(\hat{\boldsymbol{\beta}}) - \frac{\rho}{2}||\boldsymbol{r} + \boldsymbol{o}||_2^2 + \frac{\rho}{2}||\boldsymbol{o}||_2^2.$$

Where $\boldsymbol{r} = \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}$ and $\boldsymbol{o} = \rho^{-1}\boldsymbol{y}$ and ρ is the penalty parameter. This is the scaled version of the ADMM algorithm, which is less efficient but more intuitive. Setting in the terms from Equation 3.9, we can write

$$\hat{\mathcal{L}}_\beta \overset{c}{=} \underbrace{\frac{1}{2}\Big(\boldsymbol{y}^T X^T \beta - 2\boldsymbol{m}_u^T K_{m,m}^{-1} K_{m,n} \Theta X^T \boldsymbol{\beta} - \boldsymbol{\beta}^T X \Theta X^T \boldsymbol{\beta}\Big)}_{f(\boldsymbol{\beta})} - \underbrace{\frac{1}{\sigma^2}||\boldsymbol{\beta}||_1}_{g(\hat{\boldsymbol{\beta}})}$$

Again, we differentiate to find the optimal parameters.

$$\begin{aligned}
\frac{\partial f(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= 1/2 X\boldsymbol{y} - X\Theta K'_{n,m}\boldsymbol{m}_u - X\Theta X^T\boldsymbol{\beta} + \rho(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta} + \boldsymbol{o}) \\
\boldsymbol{\beta}^* &= (X\Theta X^T + I\rho)^{-1}(1/2 X\boldsymbol{y} - X\Theta K'_{n,m}\boldsymbol{m}_u + \rho(\hat{\boldsymbol{\beta}} + \boldsymbol{o})) \\
\frac{\partial g(\hat{\boldsymbol{\beta}})}{\partial \hat{\boldsymbol{\beta}}} &= -\frac{1}{\sigma^2}\partial\left(||\hat{\boldsymbol{\beta}}||_1\right) - \rho(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta} + \boldsymbol{o}) \\
\hat{\boldsymbol{\beta}}^* &= ST_{\frac{1}{\sigma^2\rho}}(\boldsymbol{\beta} - \boldsymbol{o}) \\
\frac{\partial \tilde{\mathcal{L}}_\beta}{\partial \boldsymbol{o}} &= \rho(\boldsymbol{r} + \boldsymbol{o}) - \rho\boldsymbol{o} \\
&= \boldsymbol{r} \\
\boldsymbol{o}_{k+1} &= \boldsymbol{o}_k + \boldsymbol{r}
\end{aligned}$$

With this we have the necessary update rules. In the M-step, ADMM gradient ascent updates are executed until convergence. ADMM can be further refined by using an adaptive penalty parameter ρ.

3.2.2. Fully Bayesian

When applying the Bayesian approach, we can use a standard CAVI algorithm, as treated in Section 2.8. The different priors are a normal Gaussian, a Laplace and a Horseshoe Prior.
The latter two belong to the class of hierarchical priors, more specifically the class of Gaussian scale mixture models. This means that we augment with stochastic variables λ_j that determine the variances for the priors $p(\beta_j)$ locally.

For $q(\boldsymbol{\beta})$ we assume a Gaussian distribution. As a special case of the multivariate Gaussian, we can use a Mean Field approximation, assuming independence $q(\boldsymbol{\beta}) = \prod_j q(\beta_j)$. The advantage of the Mean Field (MF) approach is, that we do not need to infer the $\frac{d^2-d}{2}$ different covariances off the diagonal of S_β. The MF approach scales linearly to the number of dimensions, but can be more unstable compared to the multivariate approach (MV). The updates of $q(\boldsymbol{\beta})$ for the normal scale mixture models, have the same form as the updates for the Gaussian prior.

Gaussian prior

The Gaussian prior $p(\boldsymbol{\beta}) = \mathcal{N}(\mathbf{0}, \Sigma_\beta)$ has a diagonal covariance matrix. Without hierarchical priors, the entries σ_{β_j} on the diagonal of Σ_β are all set to σ_β.
The KL between two Gaussians has already been established in Equation 3.5, coupled with the lower bound for the expected loglikelihood in Equation 3.4, we can calculate look at the lower bound $\mathcal{L}$ as a function of the mean and variance of $q(\boldsymbol{\beta})$.

$$\begin{aligned}\mathcal{L}(\boldsymbol{m}_\beta, S_\beta) \overset{c}{=} \frac{1}{2}\Big[&\boldsymbol{y}^T X^T \boldsymbol{m}_\beta - \boldsymbol{m}_\beta^T X \Theta X^T \boldsymbol{m}_\beta - Tr(X^T \Theta X S_\beta) \\ &- 2\boldsymbol{m}_\beta^T X \Theta \hat{\boldsymbol{m}}_f - \Big(Tr(\Sigma_\beta^{-1} S_\beta) + \boldsymbol{m}_\beta^T \Sigma_\beta^{-1} \boldsymbol{m}_\beta - \log|S_\beta|\Big)\Big]\end{aligned}$$

When assuming a **multivariate** $q(\boldsymbol{\beta})$, we find the optimal distributions by derivation.

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{m}_\beta, S_\beta)}{\partial \boldsymbol{m}_\beta} &= \frac{1}{2}\Big(X\boldsymbol{y} - 2X\Theta X^T \boldsymbol{m}_\beta - 2X\Theta\hat{\boldsymbol{m}}_f - 2\Sigma_\beta^{-1}\boldsymbol{m}_\beta\Big) \\ \frac{\partial \mathcal{L}(\boldsymbol{m}_\beta, S_\beta)}{\partial S_\beta} &= \frac{1}{2}\Big(-X\Theta X^T - {\Sigma_\beta}^{-1} - S_\beta^{-1}\Big)\end{aligned}$$

Setting to zero yields the optimal parameters:

$$\begin{aligned}{S_\beta^*}^{-1} &= \Sigma_\beta^{-1} + X\Theta X^T \\ \boldsymbol{m}_\beta^* &= S_\beta[\frac{1}{2}X\boldsymbol{y} - X\Theta\hat{\boldsymbol{m}}_f]\end{aligned}$$

These updates contain the inversion of the $d \times d$ matrix S_β, which suggests the use of the **Mean Field approximation** for high-dimensional datasets. This is a special case of the multivariate case and the optimal parameters are again found by setting the gradients to zero, with the difference that we optimize every dimension separately. Because now S_β is assumed to be diagonal matrix, we can use $\log(|S_\beta|) = Tr(\log S_\beta)$. The derivations then are

$$\begin{aligned}\frac{\partial \mathcal{L}(q(\boldsymbol{\beta}))}{\partial m_{\beta_j}} &= \frac{1}{2}\Big(\boldsymbol{x}_j\boldsymbol{y} - 2\boldsymbol{x}_j\Theta X_{-j}^T \boldsymbol{m}_{\beta_{-j}} - 2\hat{\boldsymbol{m}}_f^T\Theta\boldsymbol{x}_j^T - 2\boldsymbol{x}_j\Theta\boldsymbol{x}_j^T m_{\beta_j} - 2{\sigma_{\beta_j}}^{-1}\boldsymbol{m}_\beta\Big) \\ \frac{\partial \mathcal{L}(q(\boldsymbol{\beta}))}{\partial s_{\beta_j}} &= \frac{1}{2}\Big(-\boldsymbol{x}_j^T\Theta\boldsymbol{x}_j - {\sigma_{\beta_j}}^{-1} - s_{\beta_j}^{-1}\Big)\end{aligned}$$

Setting the gradients to zero yields.

$$s_{\beta_j}^{*\,-1} = \frac{1}{\sigma_{\beta_j}} + \boldsymbol{x}_j \Theta \boldsymbol{x}_j^T \tag{3.11}$$

$$m_{\beta_j}^* = s_{\beta_j}^* \left(\frac{1}{2} \boldsymbol{x}_j \boldsymbol{y} - \boldsymbol{x}_j \Theta X_{-j}^T \boldsymbol{m}_{\beta_{-j}} - \boldsymbol{x}_j \Theta \hat{\boldsymbol{m}}_f \right). \tag{3.12}$$

By using dynamic programming, the cost for a single $q(\beta_j)$ does not depend of the dimensionality of our problem. Further details on how this is implemented can be found in Appendix A.3. Information of the performance of the MF approximation is found in Section 4.
When using hierarchical priors, each epoch, $q(\boldsymbol{\beta})$ and then the variational distribution $q(\boldsymbol{\lambda})$ of the augmenting variable are updated.

Laplace prior

The Laplace distribution can be represented by an infinite mixture of Gaussians [11, 18]. This approach has been successfully applied to LASSO [19, 32], by augmenting the prior for $\boldsymbol{\beta}$.

$$\begin{aligned} p(\beta_j) &= \frac{1}{2\sigma_\beta} \exp\left\{ -\frac{|\beta_j|}{\sigma_\beta} \right\} \\ &= \int_0^\infty \mathcal{N}(\beta_j | 0, \frac{\sigma_\beta}{\lambda_i}) p(\lambda_i) d\lambda_i \\ \lambda_i &\sim \frac{4}{\lambda_i^2} \exp(-\frac{1}{2\lambda_i}) \end{aligned}$$

Here again we approximate the distribution $p(\boldsymbol{\beta}, \boldsymbol{\lambda})$ with the variational distributions $p(\boldsymbol{\beta}, \boldsymbol{\lambda}) \approx q(\boldsymbol{\beta}) q(\boldsymbol{\lambda}) = q(\boldsymbol{\beta}) \prod q(\lambda_i)$.
The optimal distribution $q^*(\boldsymbol{\beta})$ is the same as for the Gaussian prior, with the difference that $\sigma_{\beta_j}^2$ now is $\sigma_\beta^2 / \mathbb{E}_q[\lambda_j]$. For finding the optimal distribution $q^*(\lambda_j)$, we can use the standard CAVI approach of using the full conditional, because the full conditional on λ_i does not depend the lower bounded likelihood term.

$$\begin{aligned} p(\lambda_j | \boldsymbol{\lambda}_{-j}, \boldsymbol{y}, \boldsymbol{\omega}, \boldsymbol{f}, \boldsymbol{\beta}) &\propto p(\beta_j | \lambda_j) p(\lambda_j) \\ &\overset{c}{=} \frac{\sqrt{\lambda_j}}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} \frac{\beta_j^2 \lambda_j}{\sigma_\beta^2} \right) \frac{4}{\lambda_j^2} \exp\left(-\frac{1}{2\lambda_j} \right) \\ &\propto \lambda_j^{-\frac{3}{2}} \exp\left(-\frac{1}{2} \left(\frac{\beta_j^2 \lambda_j}{\sigma_\beta^2} + \frac{1}{\lambda_j} \right) \right). \end{aligned}$$

This is a Generalized Inverse Gaussian (GIG) distribution. The GIG has the form

$$GIG(a, b, p) = \frac{\left(\frac{a}{b}\right)^{p/2}}{2K_p(\sqrt{ab})} x^{p-1} \exp\left(-\frac{1}{2} \left(ax + \frac{b}{x} \right) \right),$$

where $K_p(\cdot)$ is the modified Bessel function of the second kind. When setting

$$q(\lambda_j)^* \propto \exp \mathbb{E}_{q(-\lambda_j)}[\log p(\lambda_j|\boldsymbol{\lambda}_{-i}, \boldsymbol{y}, \boldsymbol{\omega}, \boldsymbol{f}, \boldsymbol{\beta})] \tag{3.13}$$

we again have a GIG distribution for $q(\lambda_j)$. The optimal parameters are $a_j^* = \mathbb{E}_q[\frac{{\beta_j}^2}{\sigma_\beta^2}]$, $b = 1$ and $p = -1/2$. Because b and p, do not depend on the data, we assume $q(\lambda_j) = GIG(a_j, 1, -1/2)$. This helps us for finding the expected λ_j

$$\mathbb{E}_{GIG(\lambda_j|a_j,b,p)}[\lambda] = \frac{\sqrt{b}K_{p+1}(\sqrt{a_j b})}{\sqrt{a_j}K_p(\sqrt{a_j b})}.$$

The modified Bessel function is symmetric in p with $K_{-p}(\cdot) = K_p(\cdot)$. Therefore the Bessel function cancels out and we get $\mathbb{E}_{q(\lambda_j)}[\lambda_j] = \frac{1}{\sqrt{a_j}}$. When setting in the optimal a_j^*, the Kullback Leibler divergence is

$$\begin{aligned} KL(q(\boldsymbol{\beta})||p(\boldsymbol{\beta})) &= \mathbb{E}_{q(\boldsymbol{\beta})}\left[\log q(\boldsymbol{\beta}) + \sum_{j=0}^{d} \mathbb{E}_{q(\lambda_j)}\left[\log \frac{q(\lambda_j)}{p(\beta_j|\lambda_j)p(\lambda_j)}\right]\right] \\ &\overset{c}{=} -\frac{1}{2}\Big(\sum_{j=0}^{d}\Big(2\log K_{-1/2}(\sqrt{a_j}) + \frac{1}{2}\log(a_j)\Big) \\ &\quad + \log(\det(S_\beta))\Big) \end{aligned}$$

The full derivation can be seen in Appendix A.4.

Horseshoe prior

The Horseshoe prior, as introduced by Carvalho et al. [8], is defined as

$$\begin{aligned} p(\beta_j|\lambda_j, \sigma_\beta) &\sim \mathcal{N}(0, \lambda_j^2\sigma_\beta^2) \\ p(\lambda_j) &\sim \mathcal{C}^+(0, 1), \end{aligned}$$

where $\mathcal{C}^+(a, b)$ is the half Cauchy distribution, i.e. the Cauchy distribution on $\mathbb{R}_{>0}$ scaled with factor 2. While $p(\beta_j)$ does not have a closed form solution, it can be approximated. The Horseshoe prior was further expanded upon by adding more random variables, leading to a hyperparameter-free prior [9] and Polson and Scott [36] gave a generalization of the class of hierarchical models with a half Cauchy. Here we chose the simpler version, because it allows the choice of σ_β.
The Horseshoe has its name from the so called 'shrinkage profile'. This profile is is function from 0 to 1 and defines the shrinkage weight κ, where $\kappa = 0$ means no shrinkage, i.e. no effect from the prior to β_j, and $\kappa = 1$ total shrinkage, i.e. the prior leading to $\beta_j = 0$. The shrinkage profile for $p(\beta_j)$ resembles a Horseshoe, meaning that there is no shrinkage for large signals , while for small ones it sets all weights close to zero.

As before, we use the full conditional for finding the variational distribution for the local scale λ_j, which only depends on $p(\beta_j, \lambda_j)$.

$$q^*(\lambda_j) \propto \exp \mathbb{E}_{q(\beta_j)} \left[\log p(\beta_j|\lambda_j) + \log p(\lambda_j)\right].$$

We do not have an analytically tractable form for the updates, but the variational distribution for λ_j only depends on one parameter, which allows an efficient use of a look-up table coupled with linear interpolation. For the updates of $q(\beta_j)$, we only need the term $\sigma_\beta^{-2}\mathbb{E}_{q(\lambda_j)}[\lambda_j^{-2}]$, which can be approximated numerically. We use a transformation in variable and calculate

$$\mathbb{E}_{q(\lambda_j)}\left[\frac{1}{\lambda_j^2}\right] = \mathbb{E}_{q(\gamma_j)}\left[\gamma_j\right] = \frac{1}{a_{(b_j)}} \int_0^\infty \frac{\gamma}{1+\gamma} \exp\left(-\frac{1}{2} b_j \gamma\right) \mathrm{d}\gamma,$$

where $a_{(b)} = \int_0^\infty \frac{1}{1+\gamma} \exp\left(-\frac{1}{2}b\gamma\right) d\gamma$ is the normalizer, which only depends on b. The optimal b_j^* for $q^*(\gamma_j)$ is $\mathbb{E}_{q(\beta_j)}\left[\frac{\beta_j^2}{\sigma_\beta^2}\right]$. The Kullback-Leibler divergence reduces to

$$KL(q(\boldsymbol{\beta}, \boldsymbol{\lambda})||p(\boldsymbol{\beta}, \boldsymbol{\lambda})) \stackrel{c}{=} -\left(\sum_{j=0}^{d} \log a_{(b_j)} + \frac{1}{2}\log(\det(S_\beta))\right)$$

Further detail on this can be found in Appendix A.5.

3.3. Predictions

In order to predict the class y_t for test point $\boldsymbol{x}_t$, we use our variational distributions to approximate the posteriors:

$$\begin{aligned} p(y_t = 1|\boldsymbol{y}) &= \int \sigma(\boldsymbol{x}_t^T\boldsymbol{\beta} + f_t) p(f_t, \boldsymbol{\beta}|\boldsymbol{y}) \mathrm{d}f_t \mathrm{d}\boldsymbol{\beta} \\ &\approx \int \sigma(\boldsymbol{x}_t^T\boldsymbol{\beta} + f_t) p(f_t|\boldsymbol{u}) q(\boldsymbol{u}) q(\boldsymbol{\beta}) \mathrm{d}f_t \mathrm{d}\boldsymbol{u} \mathrm{d}\boldsymbol{\beta}. \end{aligned} \tag{3.14}$$

Because $\int p(f_t|\boldsymbol{u})q(\boldsymbol{u})\mathrm{d}u$ and $q(\boldsymbol{\beta})$ both are Normal Gaussian distributions, we can substitute $\boldsymbol{x}_t^T\boldsymbol{\beta} + f_t$ with the latent variable $\hat{y}_t$ which has the Gaussian distribution $q(\hat{y}_t) = \mathcal{N}(\hat{y}_t|m_t, {\sigma^2}_t)$, where the means and variances simply add up.

$$\begin{aligned} m_t &= {\boldsymbol{x}_t}^T \boldsymbol{m}_{\boldsymbol{\beta}} + \boldsymbol{k}_{x_t,m} K_{m,m}^{-1} \boldsymbol{m}_{\boldsymbol{u}} \\ \sigma_t^2 &= k_{x_t,x_t} + \boldsymbol{k}_{x_t,m} K_{m,m}^{-1} (S_u K_{m,m}^{-1} - I) \boldsymbol{k}_{m,x_t} + {\boldsymbol{x}_t}^T S_\beta \boldsymbol{x}_t. \end{aligned}$$

Now we can apply the logit function to our predictive distribution and marginalize

$$p(y_t = 1|\boldsymbol{y}) = \int \sigma(\hat{y}_t) q(\hat{y}_t) d\hat{y}_t \tag{3.15}$$

This is not analytically tractable, but we can either use numerical approximations or simply predict using m_t as a MAP estimator. Because both, the logistic function around zero, as well as the Gaussian around its mean, are symmetrical, the class predictions are the same in both approaches, with a higher confidence in the MAP case.
When using the numerical approximation to $p(y^* = 1|y)$, in practice it can happen, that for a very large ${\boldsymbol{\sigma}_j}^{2^*}$, the predicted confidence becomes 0.5. This can lead to a different class prediction. This only happens if the variational distributions have a large variance, which usually means that the LMM did not fit its model properly.

3.4. Hyperparameter optimization

Apart from optimizing the variational parameters, we can further use hyperparameter optimization. While this yields the danger of overfitting, it relieves us of the need of fine tuning the hyperparameters $\mathcal{H}$. We optimize $\mathcal{H}$ by using type II maximum likelihood (ML-II), as treated by Rasmussen [40]. This approach is also called Empirical Bayes and optimizes the lower bound for the marginal likelihood $p(y|X, \mathcal{H}) \geq ELBO(q|\mathcal{H})$, where $ELBO(q|\mathcal{H})$ is the ELBO given $\mathcal{H}$ as a function of the variational distributions q.
The Hyperparameters are the GP-Kernel parameters and the variables in the prior of $\boldsymbol{\beta}$. In our approach neither the priors of $\boldsymbol{\beta}$, nor the white noise in the kernel are not optimized. We do this, because we want to fix the level of sparsity of the linear model. The updates are done with gradient steps.
The hyperparameters are parametrized on an exponential scale, i.e. $\exp(\mathcal{H})$. This has several advantages. The most trivial is that this way no negative values can occur. The other is that this way the hyperparameters are on their natural scale. Usually, also grid based optimization is applied to a logarithmically spaced grid. The calculation of the gradients for the hyperparameters are lengthy but straight forward, they can be found in the Appendix A.6. After calculating the gradients, the hyperparameters are updated with help of Adaptive moment estimation (Adam) [25].
Adam computes an individual learning rate for every parameter, by estimating the first and second moment of the gradients. As can be seen in Section 4.7, Empirical Bayes can help improve the performance of the classifier substantially.

3.5. Algorithm

Empirical Bayes, has an outer loop that updates the hyperparameters and an inner loop that updates the variational parameters, as well as the MAP estimates for $\boldsymbol{\beta}$ for the VEM LASSO. When using mini-batches, the algorithm would not converge without continually decreasing the learning rate for variational parameters. Because of this, the inner loop does not terminate on convergence, but rather after a set number of iterations (v-step-num). Because the hyperparameter updates are done with gradient steps, several steps (h-step-num) are taken in the outer loop. This is done so that the outer loop updates are more accurate.
The basic algorithm with mini-batches for the fully Bayesian approaches then can be

seen in Algorithm 1. When using full batches, the old variational distributions are replaced by the new optimal distributions. The VEM algorithm is nearly equivalent to Algorithm 1, the only difference being one further update of the MAP estimate after the variational updates.

Algorithm 1 CAVI algorithm

```
set inducing points Z
initialize variational distributions
while not converged do
  for i in {1, ..., v-step-num} do
    sample mini-batch
    get q(ω)_(batch)
    for dist in variational distributions do
      get optimal dist*_(batch)
      update dist with dist*_(batch)
    end for
  end for
  for i in {1, ..., h-step-num} do
    sample mini-batch
    get q(ω)_(batch)
    get ∇_H L̃_(batch)
    hyperparameter ADAM step(∇_H L̃_(batch))
  end for
  get L̃_(batch)
end while
```

4. Experiments

In order to evaluate the presented algorithm, its performance with respect to several aspects is evaluated. Firstly the different variants are compared in run-time, convergence and performance. The variants include all treated priors and the VEM and fully probabilistic models, as well as the learned linear components of the LMM. Furthermore the model is compared to other related classifiers. To this end, the feature selection stability, the correlation to the confounding variables, performance as well as the generalization over different classes in a family is examined.

4.1. Baselines

As baselines a Logistic Linear regressor, a Sparse Gaussian Process classifier and a Black-Box approach have been used. The first two are the two extremes of the presented model. For $\boldsymbol{f} = \mathbf{0}$ our model corresponds to a linear logit classifier and for $\boldsymbol{\beta} = \mathbf{0}$ to the XGPC by Wenzel et al. [47].
For the **linear classifier** baseline an implementation from scikit-learn [34] was used with a $\ell 1$ regularization, which corresponds to the MAP estimation of the Laplace prior.
Sparse Gaussian Process Classification: Because no XGPC code is available for Python, the GPflow [30] implementation of the SPGC [22] was used. Because SPGC does not support sparse matrices, a proprietary implementation was used. This is the presented model with fixed weights $\boldsymbol{\beta} = \mathbf{0}$ and is denoted as XGPC.
The last baseline is a **Black-Box approach** that models the noise $\boldsymbol{f}$ with a neural network, with two hidden layers with 32 nodes each, has been used as a further model. The confidence is then calculated the same way as in the presented model:

$$p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \prod \sigma(y_i(\boldsymbol{x}_i^T \boldsymbol{\beta} + f_i))$$

The neural network is regularized with a $\ell 2$ norm, because of its similarity to the Gaussian prior, and the weights are normalized with a $\ell 1$ norm to enforce sparsity. The Black-Box approach was implemented in Pytorch [33]. The loss to be minimized is the predicted probability for the wrong class. Training consists of 100 epochs over the whole data. This novel approach is the solution to classification of correlated noise from the view of a Machine Learner. As a side-note for this classifier, the implementation here is very simplistic and the performance of the Black-Box can possibly be improved substantially, but this was out of scope for this thesis.

4.2. Performance Metrics

A informative measure for classification performance is the F_1 score, as it considers precision as well as recall. It is defined as

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

for a single class. The reported F_1 score here is the average for class 1 and -1.
Further performance measures are the Accuracy and the Area Under the Receiver Operator Curve (ROC AUC). The performances are given with the standard error.
Performance measures on the test set alone do not properly measure the correct isolation of the correlated noise. In fact it is difficult to measure this on a non-synthetic dataset, as we do not know the underlying model. Even if we know the underlying linear weights $\tilde{\boldsymbol{\beta}}$, only looking at the absolute differences between $\tilde{\boldsymbol{\beta}}$ and the learned weights $\boldsymbol{\beta}$ can be misleading. In a sparse setting, assuming $\boldsymbol{\beta} = \mathbf{0}$ yields a very low difference to $\tilde{\boldsymbol{\beta}}$, while not representing the underlying model at all. One solution here is using the labels y_{lin}, which are the labels created by the underlying linear model ($X^T\tilde{\boldsymbol{\beta}}$) alone, because this is the model we want to recover.
When using data with different subcategories, we can try to simulate different confounders by training on one set of subcategories and testing on another. This only works if our assumption holds, that there are correlations in members of a subcategory, while linear dependencies model the parent category.

4.3. General Behavior of the Presented Model

When running the CAVI algorithm on full batch, the first updates lead to the biggest increase in ELBO, followed by continuous improvement until convergence. The full batch CAVI updates never worsen the ELBO, because of they are analytically exact coordinate ascent updates over the whole dataset. The ML-II updates, on the other hand, can overshoot the maximum and lower the ELBO, because they are gradient steps. By choosing an adequate learning rate, this problem can be mitigated.
Figure 4.1 shows the ELBO and scores over time in the upper and lower row respectively. The left column shows an example for a full-batch run and the right one a mini-batch setting. We can see that maximizing the ELBO does not necessarily optimize the score on the test-set. The ELBO consists of the KL divergences for $\boldsymbol{u}$, $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ and the log likelihood. The upper figures show the development of the different parts up to a constant.
When using mini-batches, the course of the ELBO is neither smooth nor monotonous, because the updates and calculation of the ELBO are based on noisy mini-batch estimates. Another difference between the results of both approaches is, that, given the dataset, the full-batch runs are deterministic, while the sampled mini-batches influence the course of the algorithm.

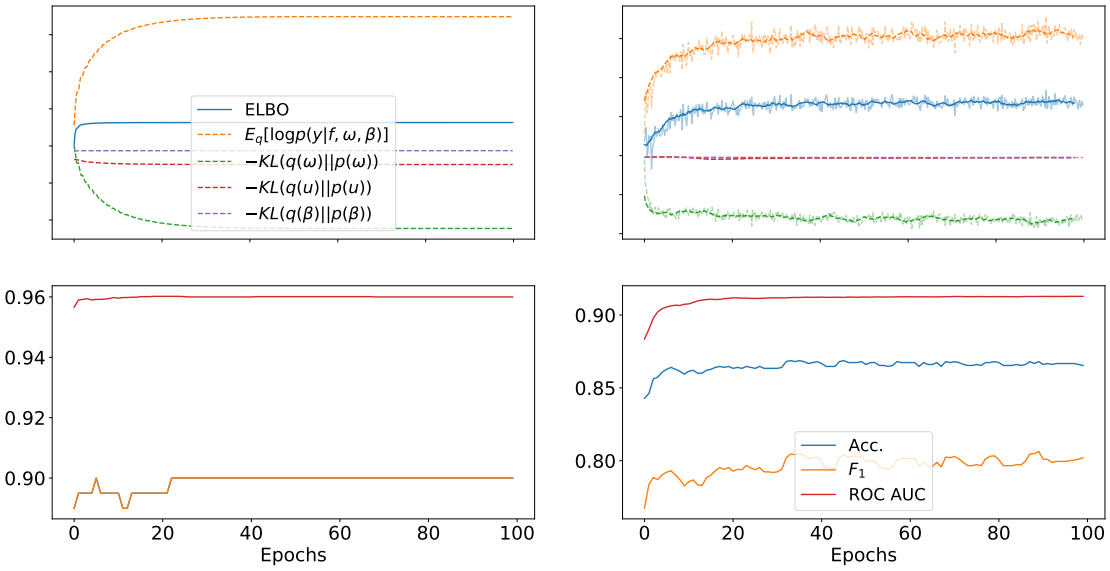

Figure 4.1.: General behavior with full-batch (left) and mini-batch updates (right). In the left image, the F_1 and the accuracy score are equal, because both classes have the same size.

Note that the scores in the lower row of Figure 4.1 are based on the MAP predictions and not on the numerical approximations. This is because the former ones are much faster to calculate and thus a good proxy for the real prediction scores. The accuracy and F_1 scores are the same as for the final predictions, while the ROC AUC differs marginally.

4.4. Practicalities

Because the **labels** have to be in $\{-1, 1\}$, they possibly have to be transformed during initialization.
If the data matrix is not sparse, it is **normalized** to mean 0 and a variance of 1. Normalizing a sparse matrix would destroy the sparsity of the data. To mitigate this, one dimension with ones is added for sparse data, so a intercept term can be learned and applied.
Inducing points are found with k-Means clustering. For practical reasons, the inducing points are sampled from the data, if the data matrix has more than 10^{10} entries.
If not specified previously, the **length scale** for the RBF kernels is set to the median Euclidean distance between the samples. For more than 500 samples, the length scale is set to the median of 500 randomly sampled data points.
For small mini-batches – with less than 500 samples – and a high variance – $\tau > 0.3$ – in the prior $p(\boldsymbol{\beta})$, the MF approaches can get **unstable**. Because the different $q(\beta_j)$ are updated individually, the KL divergences for $\boldsymbol{\beta}$ and the local parameters $\boldsymbol{\omega}$ grow without convergence, leading to worse ELBO scores. Thus for small mini-batches with $s < 500$, the updates for $q(\boldsymbol{\beta})$ are repeated several times ($\lceil \frac{-s}{100} + 6 \rceil$). Furthermore, the weights $\boldsymbol{\beta}$ are limited to be in $[-50, 50]$.
If not specified otherwise the hyperparameter σ_β for the variance $p(\boldsymbol{\beta})$ are set to 0.05. This value, showed the best performance for selecting features on the training-sets. Furthermore experiments are usually repeated five times.

4.5. Datasets

4.5.1. Toy Data

This synthetic data set helps us understand the characteristics of the classifiers, because it allows us to test the classifiers in special circumstances. The distribution of data points follow the probabilistic model presented in this thesis.

$$\begin{aligned} \boldsymbol{y} &= \mathbb{1}'(X^T \tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{f}}) \\ \tilde{\boldsymbol{f}} &\sim \mathcal{N}(\mathbf{0}, \tilde{K}(X, X)), \end{aligned}$$

where $\mathbb{1}'(x) = 1 \; \forall x \geq 0$, $\mathbb{1}'(x) = -1 \; \forall x < 0$. The kernel $\tilde{K}(X, X)$ models the covariance between the data points. This is done with a GP, with the Hyperparameters log of the length scale $\tilde{\mathcal{H}}_{ls}$, log of variance for the RBF kernel $\tilde{\mathcal{H}}_{RBF}$, the linear kernel $\tilde{\mathcal{H}}_{lin}$ and the white kernel $\tilde{\mathcal{H}}_{white}$. For all experiments with Toy data, the added white noise is set to 0.1 The non zero entries of $\tilde{\boldsymbol{\beta}}$ are uniformly samples in $[-2, 2]$.

4.5.2. Tuberculosis Disease Outcome Prediction

This dataset by Berry et al. [3] uses gene expression levels to predict the outcome of Tuberculosis. It has 143 samples with 103 healthy controls. For each patient there are 48,803 gene expression levels, which we use as features, and seven additional features, which consist of age, gender and race, one hot encoded into 5 dimensions. This additional information is used as side information. This dataset consists of few samples with many features, which suggests the use of a sparse prior.
On TBC, the LMMs were trained for 100 epochs without inducing points or mini-batches, the test set are 10% of the data set randomly sampled.

4.5.3. Drebin

For this dataset Arp et al. [1] analyzed Android applications and found 5,560 malware samples from 179 different families, resulting in more than 120,000 samples with over 500,000 dimensions. The features are binary encoded information on the requested hardware components, permissions, app components and inter- or intra-process communication. This dataset shows that the developed algorithm scales to a large number of features as well as samples.
Because of the different malware families, we can test our algorithm for generalization over different families.
Training was done on this dataset with a batch-size of 2000 and 200 inducing points for 40 epochs. For the test set we randomly sample 5% of the data.

4.5.4. Adult

The Adult Data Set[1] gives the task of predicting whether an adult earns more than 50,000 $ per year, based on 14 different attributes. After one-hot encoding the categorical features, we get 103 attributes. The dataset contains more than 30,000 samples. Attributes contain race and sex, which are ideal for usage as side information. In this setting our linear classifier does not discriminate, but rather can marginalize over the these confounders and find the underlying causalities for income.
Here the classifiers were trained for 200 epochs with a mini-batch size of 2000 and 200 inducing points.

4.6. Comparing the different variants

4.6.1. Convergence and Runtime

In order to evaluate the viability of the Mean Field and the MAP approximations, we first look at the convergence and runtime with respect to the number of features of the different models on the Toy dataset without hyperparameter updates[2]. The results on

[1] http://archive.ics.uci.edu/ml/datasets/Adult

[2] For the different dimensionalities d, $n = 1000$ is fixed and the number of non-zero entries in $\tilde{\boldsymbol{\beta}}$ grows with d by using $\lfloor \frac{d}{10} \rfloor$.

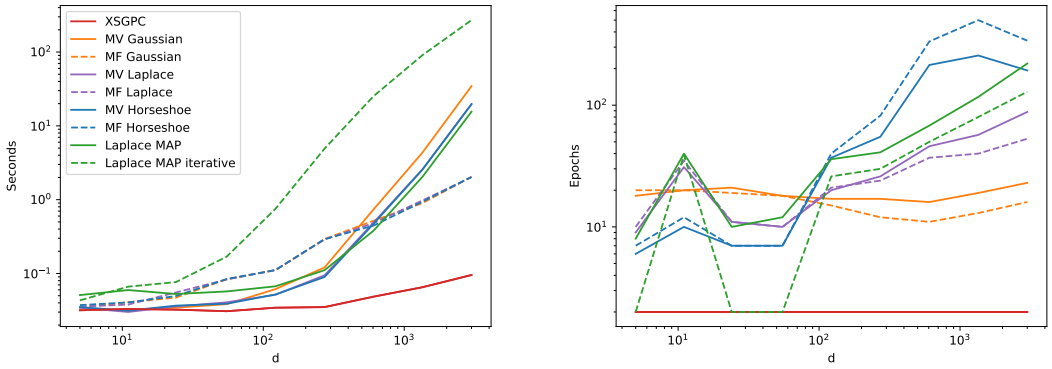

Figure 4.2.: Convergence of the different variants: Average time per epoch and number of epochs until convergence.

this can be seen in Figure 4.2. It shows that the hierarchical priors generally converge slower than the simple Gaussian prior. The overhead for the MF approaches means that, in a low-dimensional setting, an epoch takes longer than the MV variant. Overall we can see that the additional linear term $X^T\boldsymbol{\beta}$ entails far slower convergence than the XGPC, which maximizes $q(\boldsymbol{u})$ in one step. Apart from the convergence, one epoch also takes longer with the added linear term.
The MAP approximations to the Laplace priors – shown in green – behave similarly to the fully Bayesian approaches. One epoch with the iterative approach (cf. Section 3.2.1) takes the longest because one update consists of coordinate ascent for the individual β_j until convergence. The Laplace MAP approximation using ADMM (Section 3.2.1) essentially behaves like the multivariate approaches, runtime-wise.
Furthermore we can see that the Horseshoe prior converges the slowest. This is possibly because of the use of numerical approximation for $q(\boldsymbol{\lambda})$, which was avoided for all other priors.
These results give an indication when to use Mean Field and when to use a multivariate variational distribution for $\boldsymbol{\beta}$. Up to a dimensionality of roughly 500 the MV is faster, but with more dimensions the multivariate approach is less and less feasible.

4.6.2. Performance

In order to evaluate the performance of the MAP and Mean Field approximations, we look at the performance with full batch updates on the Toy dataset [3] and the Adult Data Set with mini-batch training with $s = 2000$.
Table 4.1 shows the performances for both experiments. In the full-batch run, the MF and MV approaches lead to exactly the same results in F_1 and nearly identical ones in ROC AUC. For the mini-batches, the Mean Field approaches behave similarly to the

[3] The parameters were $n = 500$, $d = 50$, with the first 5 entries in $\tilde{\boldsymbol{\beta}}$ being non-zero, the kernel parameters were $\mathcal{H}_{lin} = \exp(-3)$, $\mathcal{H}_{rbf} = \exp(-2)$ and $\mathcal{H}_{ls} = \exp(-0.5)$

	Toy dataset		**Adult dataset**	
	F_1	**ROC AUC**	F_1	**ROC AUC**
MV Gaussian	.93	.98	.79	.91
MF Gaussian	.93	.98	.80	.91
MV Laplace	.96	.99	.70	.91
MF Laplace	.96	.99	.80	.91
MAP Laplace	.90	.97	.80	.91
MAP Laplace Iterative	.95	.99	.80	.91
MV Horseshoe	.94	.99	.80	.91
MF Horseshoe	.94	.99	.80	.91

Table 4.1.: Performance comparison for the Mean Field approximation and the MAP estimate. The standard error for the Adult dataset was less than 0.02 for all entries, the results on the Toy data are deterministic.

multivariate ones. The MAP approximations perform worse than their fully Bayesian counterparts and thus are not used for further performance benchmarks.

4.7. Evaluation of Empirical Bayes

In order to evaluate the success of the ML-II optimization, first the ELBO($\mathcal{H}_{ls}$) and $F_1(\mathcal{H}_{ls})$ are obtained via grid search. Here the all hyperparameters, but the length scale, are fixed and ELBO($\mathcal{H}_{ls}$) is the ELBO of the converged CAVI algorithm as a function of the length scale, similarly for $F_1(\mathcal{H}_{ls})$.
A classifier then is trained using the presented algorithm with ML-II updates and the course of the optimization is recorded. Figure 4.3 shows the result of this experiment [4]. Because only hyperparameters for the GP are optimized, the form of $p(\boldsymbol{\beta})$ does not have an effect on the Empirical Bayes updates. We can see that maximizing the ELBO does not necessarily mean optimizing the classification performance. This means that ML-II has to be handled with care.

4.8. Comparison with baselines

The performance benchmarks for TBC and Adult are found in Table 4.2.
The MF Gaussian and SGPC do not perform better than random for TBC as can be seen by the ROC AUC score of 0.5. Thus the non-sparse Gaussian prior is not a good choice for the high-dimensional datasets, like TBC. The Benchmark for XGPC was added for TBC, because it fits its model substantially better than SGPC to the dataset. We can

[4] The plot was done a dataset with the following parameters: 800 datapoints (600 train/200 test) with two dimensions where one dimension of $\tilde{\beta}$ is zero; the Kernel parameters were $\tilde{\mathcal{H}}_{ls} = -1$, $\tilde{\mathcal{H}}_{RBF} = 2$, $\tilde{\mathcal{H}}_{lin} = 0$, $\exp(\tilde{\mathcal{H}}_{white}) = 0.1$ and $\epsilon \sim \mathcal{U}(0, 0.2)$. The prior on $p(\boldsymbol{\beta})$ was a Gaussian, $q(\boldsymbol{\beta})$ was multivariational.

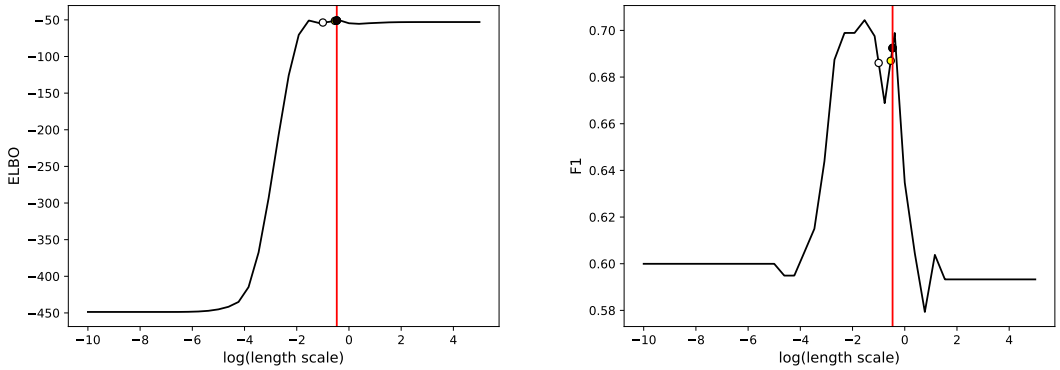

Figure 4.3.: The outcome of hyperparameter optimization. The red line shows the found value of length scale $\mathcal{H}_{ls}$, the black graph shows ELBO($\mathcal{H}_{ls}$) and $F_1(\mathcal{H}_{ls})$ respectively. The markers show the start value (white) up to the found value (black).

see that the presented models with sparse priors perform best in this dataset.
Apart from the predictions using the full LMM, classifiers based on the linear model using only $\boldsymbol{\beta}$ is evaluated. These linear classifiers are denoted with 'only $\boldsymbol{\beta}$' and will be further assessed in the following sections. For TBC, we can see that the linear models for the sparse priors, while having a lower score than the corresponding full LMM, still outperform the Logistic Regressor in F_1. Interestingly, the linear models for the Laplace and Horseshoe perform similarly, while the corresponding full LMMs differ by 0.1 in F_1.

In the Adult dataset, all LMMs and Logistic Regression are tied for the first place in performance. Neither the sparse priors nor the presented design show an advantage on this comparatively low-dimensional dataset. The linear components using $\boldsymbol{\beta}$, perform worse than the full models and Logistic Regression.

For the evaluation on Drebin, two different scenarios were used, firstly the standard setting, where the dataset was randomly separated into train- and test-set, with a ratio of 0.95/0.05. For the second experiment, the malware subfamilies were separated, with 90% of the subclasses being used for training and the other for testing. The benign samples were distributed to the two sets with the ratio of the malware samples. This was done in order to test the hypothesis, that the linear model is better poised to finding the characteristics of malign software. Table 4.3 shows the results for both experiments. We can see that in the standard setting Logistic Regression performs best, closely followed by the Mean Field Horseshoe and Laplace.
When testing on the generalization over different subfamilies, we can see that the linear components perform significantly better in F_1, while the other classifiers, do not show a good generalization for new, unseen categories.

	TBC		**Adult**	
	F_1	ROC AUC	F_1	ROC AUC
MF Gaussian	$.23_{\pm.02}$	$.50_{\pm.00}$	$\mathbf{.80}_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
MF G. only $\boldsymbol{\beta}$	$.23_{\pm.02}$	$.54_{\pm.06}$	$.69_{\pm.01}$	$.89_{\pm.00}$
MF Laplace	$\mathbf{.75}_{\pm.08}$	$\mathbf{.88}_{\pm.05}$	$\mathbf{.80}_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
MF L. only $\boldsymbol{\beta}$	$.64_{\pm.06}$	$.75_{\pm.06}$	$.69_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
MF Horseshoe	$.65_{\pm.06}$	$.87_{\pm.04}$	$\mathbf{.80}_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
MF H. only $\boldsymbol{\beta}$	$.64_{\pm.06}$	$.75_{\pm.06}$	$.70_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
Log. Reg.	$.59_{\pm.08}$	$.77_{\pm.07}$	$\mathbf{.80}_{\pm.00}$	$\mathbf{.91}_{\pm.00}$
XGPC	$.57_{\pm.07}$	$.71_{\pm.07}$	-	-
SGPC	$.23_{\pm.02}$	$.50_{\pm.00}$	$.76_{\pm.00}$	$.89_{\pm.01}$
Black Box	$.41_{\pm.01}$	$.65_{\pm.08}$	$.43_{\pm.02}$	$.86_{\pm.01}$

Table 4.2.: Results for experiments on TBC and Adult with 5 repetitions. The test/train splits are 0.9/0.1 and 0.95/0.05 respectively.

	Drebin		**Drebin separate classes**	
	F_1	ROC AUC	F_1	ROC AUC
MF Laplace	$.89_{\pm.01}$	$\mathbf{.98}_{\pm.00}$	$.59_{\pm.05}$	$.94_{\pm.04}$
MF L. only $\boldsymbol{\beta}$	$.85_{\pm.01}$	$.97_{\pm.00}$	$\mathbf{.79}_{\pm.10}$	$.93_{\pm.07}$
MF Horseshoe	$.90_{\pm.01}$	$\mathbf{.98}_{\pm.00}$	$.58_{\pm.04}$	$.93_{\pm.03}$
MF H. only $\boldsymbol{\beta}$	$.86_{\pm.01}$	$.97_{\pm.00}$	$.74_{\pm.11}$	$.88_{\pm.05}$
Log. Reg.	$\mathbf{.91}_{\pm.00}$	$\mathbf{.98}_{\pm.00}$	$.56_{\pm.05}$	$\mathbf{.95}_{\pm\mathbf{.03}}$
XGPC	$.78_{\pm.01}$	$.96_{\pm.00}$	$.48_{\pm.00}$	$.85_{\pm.07}$

Table 4.3.: Results for experiments on TBC with 0.9/0.1 train-test split and 5 repetitions.

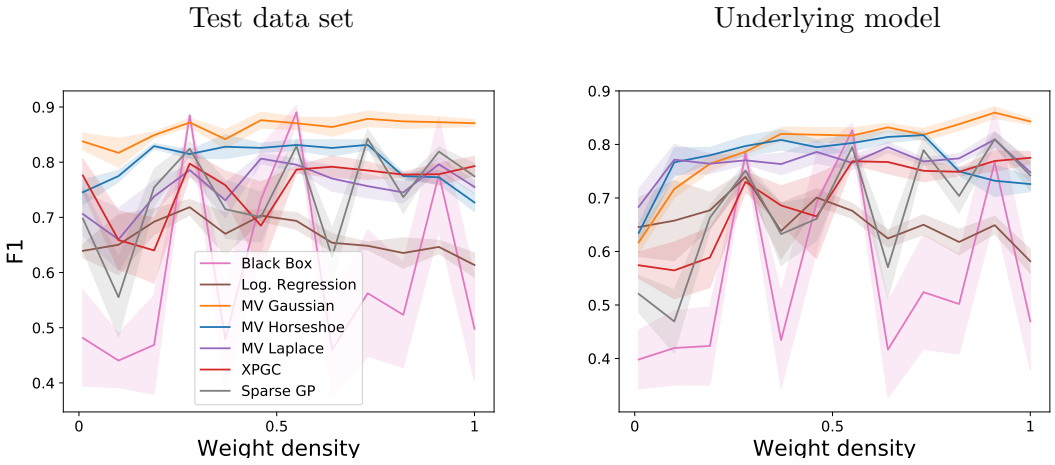

Figure 4.4.: Performance as a function of sparsity in $\boldsymbol{\beta}$ on the Toy dataset with $d = 100$. On the left the F_1 score on the confounded test set is shown, on the right the score for the underlying model $y_i = \mathbb{1}'(\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}})$.

4.8.1. Performance different ratios non-zero weights

In order to evaluate the quality of the linear model, the performance of the different classifiers as a function of the sparsity of $\tilde{\boldsymbol{\beta}}$ evaluated. This was done by randomly sampling a $\tilde{\boldsymbol{\beta}}$ with a given sparsity and creating a toy dataset[5]. All classifiers were trained on this dataset and their performance on a test set with and without correlated noise is evaluated. Figure 4.4 shows the results with 5 repetitions, the scores for the linear components are not shown, but here they show the same behavior as their full LMMs.
Even though the dense Gaussian prior performs best on the test set with correlated noise, the sparse priors best predict the underlying model in a sparse setting. This hints at the LMM incorporating effects from the confounder when using the Gaussian prior, which is contrary to our set goal.
Furthermore we can see that XGPC and SGPC perform similarly to each other. The approaches with sparse priors – the Laplace and Horseshoe prior, as well as Logistic Regression – perform best on the underlying model for sparse settings, while the approaches with dense priors – Gaussian prior and GP approaches – work best in a dense setting. The Black Box approach shows the most unstable performance.

4.8.2. Correlation with Confounder

The correlation of the linear classifiers with the confounders can be tested by looking at the features deemed most important by the classifiers, i.e. high absolute value for the corresponding weight, and their correlations to the confounding variable. The presented model uses the confounding variables only for the GP and thus the linear weights

[5]The log of the hyperparameters used are $\tilde{\mathcal{H}}_{RBF} : 0$, $\tilde{\mathcal{H}}_{lin} : -2$ and $\tilde{\mathcal{H}}_{ls} : 2$, n was 1000 with $d = 100$

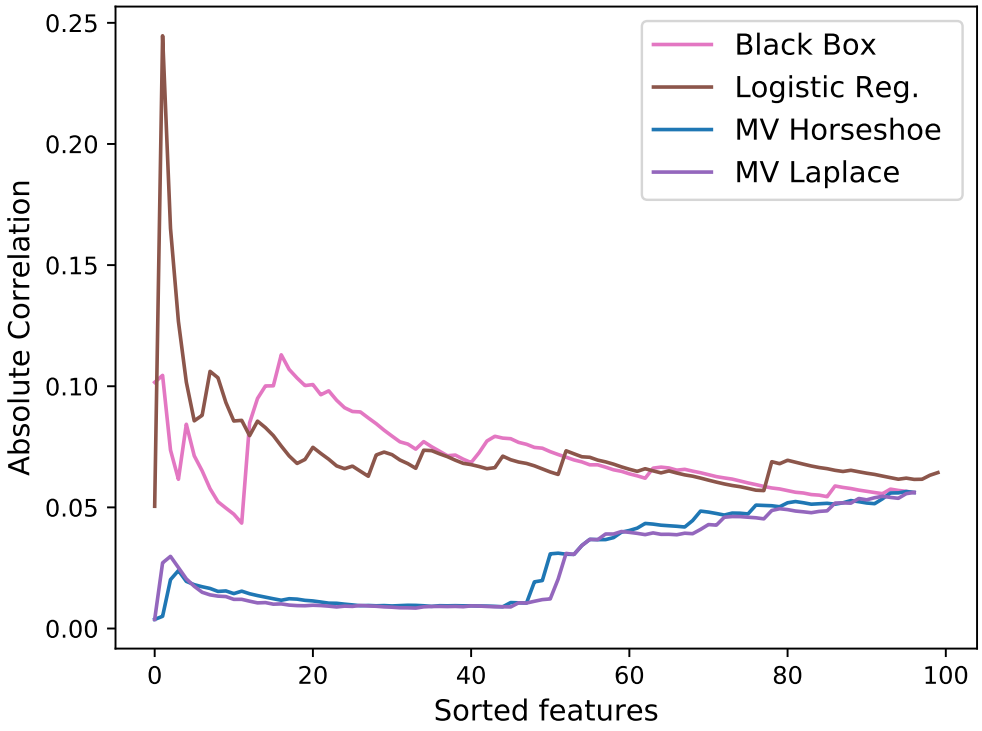

Figure 4.5.: Correlation to confounding variable sex. The weights are sorted to average absolute weight, the correlation to the confounder is calculated on the whole dataset. The shown y-value at $x = j$ is the average correlation of the j first weights.

neither needs nor has access to the confounding variables. The confounding attribute we investigate here is the sex of the person in the Adult Data Set. Figure 4.5 shows the smoothed absolute correlations of the features as sorted by the different classifiers, following the experimental design of [28]. We can see that the presented model has less correlation to the confounders than the baselines. This underlines the usefulness of the presented model.

4.8.3. Selection Stability

In order to test the selection stability, the classifiers were trained 20 times on random subsets of the data for the Adult and TBC data sets. For the former the subset contained 40% of the dataset and for the latter 75%. In every iteration first a LMM with a Laplace prior was optimized with Hyperparameter updates. This was done to find the optimal Kernel for the GP component. Then the models were trained using the fixed optimized

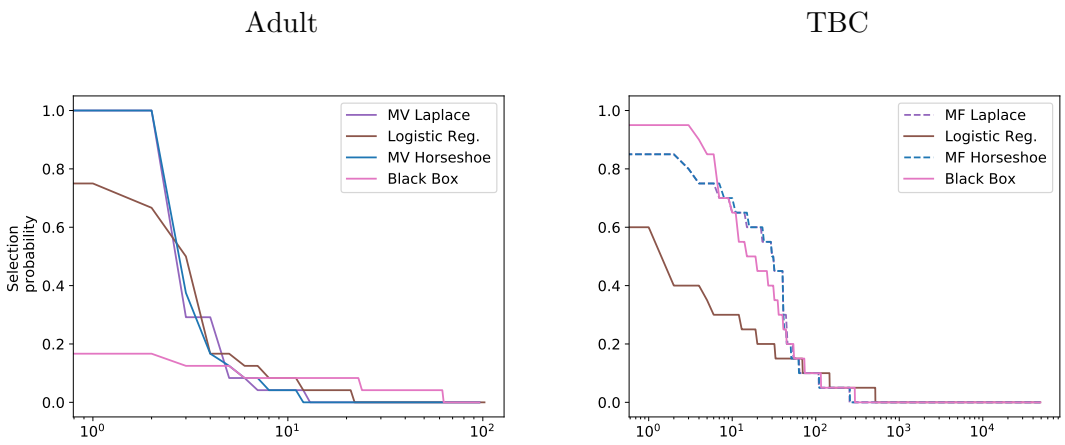

Figure 4.6.: Feature Selection stability for the top 3 and 40 features on Adult and TBC respectively.

kernel with a Laplace and a Horseshoe prior. Because it is low-dimensional, multivariate $q(\boldsymbol{\beta})$ are used for the Adult dataset and the three most weighted features count as selected. For TBC, we used MF distributions and selected the top 40 features.
Figure 4.6 shows that the presented LMMs with sparse priors have a higher selection stability than Logistic Regression in both datasets. The Black Box approach works best in TBC and worst in the Adult set. This again shows unstable behavior and hints at unused potential in the architecture of this approach.

5. Conclusion

This thesis presents an approach to linear classification in the presence of correlated noise. The approach scales to the number of features by using inducing points and mini-batches. With the Mean Field approximation it scales to the number of dimensions as well. This linear complexity in d is the minimum achievable complexity, because the d different distributions $q(\beta_j)$ have to be estimated.
Two Maximum A Posteriori approximations for the use of the Laplace prior with an own Variational Expectation Maximization algorithm are derived. In practice, these approximations have no advantage over the probabilistic models. For these fully Bayesian approaches, a theoretical framework is presented that allows the easy integration of further Gaussian scale mixture models. The sparse priors Laplace and Horseshoe outperform their dense counterparts in sparse settings, thus showing a desired behavior. With an adequate prior the algorithm is able to identify the linear dependencies and is robust against correlated noise. The Python implementation can be found on Gitlab[1].

As shown in the experimental section, the model allows an estimation of the latent linear model without suffering in performance, when using the full LMM. The linear component shows a better generalization to unseen subfamilies in the Drebin dataset and the full LMM performs substantially better in a high-dimensional setting with few data-points, compared to a simple linear model or Sparse Gaussian Process. It thus shows the desired properties of isolating correlated noise.
The cost of these advantages are a longer runtime and slower convergence, because of the interactions between the two components of the model. As these interactions happen between two latent models, they have not yet been properly investigated. It is likely, that the performance of the linear component can be improved after better understanding these latent effects. Ideally the performance of the latent linear model can be improved without impairing the behavior of the full LMM. The influence of hyperparameter optimization also plays an important role here. It might further be beneficial to first train both or one component separately, before optimizing them jointly.
The presented Black-Box approach shows an unstable behavior; improving it could be a simple and possibly fruitful extension of this thesis.

An ideal application for the presented LMM is statistical genetics, with the task of identifying genetic causes for diseases. Our model can also be used to avoid using a classifier, which shows a bias against minorities, as has happened recently. We are planning on further investigating the behavior of the LMM on new data for these applications.

[1] https://gitlab.tubit.tu-berlin.de/lenz3000/Sparse-Probit-GP-LMM

Bibliography

[1] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[2] William Astle, David J Balding, et al. Population structure and cryptic relatedness in genetic association studies. *Statistical Science*, 24(4):451–471, 2009.

[3] Matthew PR Berry, Christine M Graham, Finlay W McNab, Zhaohui Xu, Susannah AA Bloch, Tolu Oni, Katalin A Wilkinson, Romain Banchereau, Jason Skinner, Robert J Wilkinson, et al. An interferon-inducible neutrophil-driven blood transcriptional signature in human tuberculosis. *Nature*, 466(7309):973, 2010.

[4] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.

[5] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518): 859–877, 2017.

[6] Stephen Boyd. Alternating direction method of multipliers. In *Talk at NIPS workshop on optimization and machine learning*, 2011.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[8] Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling sparsity via the horseshoe. In *Artificial Intelligence and Statistics*, pages 73–80, 2009.

[9] Carlos M Carvalho, Nicholas G Polson, and James G Scott. The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480, 2010.

[10] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[11] Christian Donner and Manfred Opper. Inverse ising problem in continuous time: A latent variable approach. *Physical Review E*, 96(6):062104, 2017.

[12] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

[13] Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[14] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33 (1):1, 2010.

[15] Yarin Gal and Richard Turner. Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning*, pages 655–664, 2015.

[16] Andrew Gelman et al. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.

[17] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.

[18] Federico Girosi. Models of noise and robust estimation. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1991.

[19] Chris Hans. Bayesian lasso regression. *Biometrika*, 96(4):835–845, 2009.

[20] Charles R Henderson. Estimation of genetic parameters. In *Biometrics*, volume 6, pages 186–187. International Biometric Soc 1441 I ST, NW, Suite 700, Washington, DC 20005-2210, 1950.

[21] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[22] James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

[23] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[24] Hemant Ishwaran, J Sunil Rao, et al. Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Hannes Leeb and Benedikt M Pötscher. The finite-sample distribution of post-model-selection estimators and uniform versus nonuniform approximations. *Econometric Theory*, 19(1):100–142, 2003.

[27] Fred B Lempers. Posterior probabilities of alternative linear models. 1971.

[28] Limin Li, Barbara Rakitsch, and Karsten Borgwardt. ccsvm: correcting support vector machines for confounding factors in biological data classification. *Bioinformatics*, 27(13):i342–i348, 2011.

[29] Stephan Mandt, Florian Wenzel, Shinichi Nakajima, John Cunningham, Christoph Lippert, and Marius Kloft. Sparse probit linear mixed model. *Machine Learning*, 106(9-10):1621–1642, 2017.

[30] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017. URL `http://jmlr.org/papers/v18/16-537.html`.

[31] Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

[32] Trevor Park and George Casella. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook (version: November 15, 2012), 2012.

[36] Nicholas G. Polson and James G. Scott. On the half-cauchy prior for a global scale parameter. *Bayesian Anal.*, 7(4):887–902, 12 2012. doi: 10.1214/12-BA730. URL `https://doi.org/10.1214/12-BA730`.

[37] Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.

[38] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6 (Dec):1939–1959, 2005.

[39] Barbara Rakitsch, Christoph Lippert, Oliver Stegle, and Karsten Borgwardt. A lasso multi-marker mixed model for association mapping with population structure correction. *Bioinformatics*, 29(2):206–214, 2012.

[40] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[41] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[42] Jun Shao. Bootstrap model selection. *Journal of the American Statistical Association*, 91(434):655–665, 1996.

[43] Alex J Smola and Peter L Bartlett. Sparse greedy gaussian process regression. In *Advances in neural information processing systems*, pages 619–625, 2001.

[44] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[45] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[46] Michalis K Titsias and Miguel Lázaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in neural information processing systems*, pages 2339–2347, 2011.

[47] Florian Wenzel, Theo Galy-Fajou, Christan Donner, Marius Kloft, and Manfred Opper. Efficient gaussian process classification using polya-gamma data augmentation. *arXiv preprint arXiv:1802.06383*, 2018.

[48] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

A. Appendix

A.1. Updates for $q(u)$

In order to recover the optimal distribution $q^*(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}_u, S_u)$, we find the gradients of $\mathcal{L}$ w.r.t. $\boldsymbol{m}_u$ and S_u. The lower bound $\mathcal{L}$ is defined in Equation 3.3 as

$$\begin{aligned}\mathcal{L} := \mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] \\ - KL(q(\boldsymbol{u})||p(\boldsymbol{u})) - KL(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - KL(q(\boldsymbol{\beta})||p(\boldsymbol{\beta})).\end{aligned}$$

The last two KL divergences do not depend on $\boldsymbol{u}$, so we only need to find the gradients of the other two terms. The first one is computed in Equation 3.4:

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{f})q(\boldsymbol{\omega})q(\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] = \frac{1}{2}\Big[&\boldsymbol{y}^T(X^T\boldsymbol{m}_\beta + \hat{\boldsymbol{m}}_f) - \boldsymbol{m}_\beta^T X\Theta X^T\boldsymbol{m}_\beta - Tr(X^T\Theta X S_\beta) \\ &- \hat{\boldsymbol{m}}_f{}^T\Theta\hat{\boldsymbol{m}}_f - Tr(\Theta\tilde{K}) - Tr(K'_{m,n}\Theta K'_{m,n}S_u) \\ &- 2\boldsymbol{m}_\beta^T X\Theta\hat{\boldsymbol{m}}_f\Big]. \qquad \text{(A.1)}\end{aligned}$$

The Kullback Leibler divergence $KL(q(\boldsymbol{u})||p(\boldsymbol{u}))$ for the m-dimensional Gaussian distributions $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}_u, S_u)$ and $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, K_{m,m})$ is

$$KL(q(\boldsymbol{u})||p(\boldsymbol{u})) \overset{c}{=} \frac{1}{2}\left(Tr(K_{m,m}^{-1}S_u) + \boldsymbol{m}_u^T K_{m,m}^{-1}\boldsymbol{m}_u + \log\frac{|K_{m,m}|}{|S_u|}\right)$$

we find the derivative of this term w.r.t. S_u and $\boldsymbol{m}_u$

$$\begin{aligned}\frac{\mathcal{L}}{\partial S_u} &= \frac{\partial}{\partial S_u}\mathbb{E}_{q(\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] - \frac{\partial}{\partial S_u}KL(q(\boldsymbol{u})||p(\boldsymbol{u})) \\ &= \frac{1}{2}\frac{\partial}{\partial S_u}\Big[-Tr(K'_{m,n}\Theta K'_{m,n}S_u)\Big] - \frac{1}{2}\frac{\partial}{\partial S_u}\left[Tr(K_{m,m}^{-1}S_u) - \log|S_u|\right] \\ &= \frac{1}{2}\left(-K'_{m,n}\Theta K'_{n,m} - K_{m,m}^{-1} + S_u^{-1}\right) \\ \frac{\mathcal{L}}{\partial \boldsymbol{m}_u} &= \frac{1}{2}\frac{\partial}{\partial \boldsymbol{m}_u}\Big[\boldsymbol{y}^T\hat{\boldsymbol{m}}_f - \hat{\boldsymbol{m}}_f{}^T\Theta\hat{\boldsymbol{m}}_f - 2\boldsymbol{m}_\beta^T X\Theta\hat{\boldsymbol{m}}_f\Big] \\ &\quad - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{m}_u}\left[\boldsymbol{m}_u^T K_{m,m}^{-1}\boldsymbol{m}_u\right] \\ &= \frac{1}{2}\left[K'_{m,n}\boldsymbol{y} - 2K'_{m,n}\Theta K'_{n,m}\boldsymbol{m}_u - 2K'_{m,n}\Theta X^T\boldsymbol{m}_\beta - 2K_{m,m}^{-1}\boldsymbol{m}_u\right].\end{aligned}$$

Setting the gradients to zero, we find the optimal parameters:

$$S_u^* = \left(K_{m,m}^{-1} + K'_{m,n}\Theta K'_{n,m}\right)^{-1} \tag{A.2}$$

$$\boldsymbol{m}_u^* = S_u^*\left(\frac{1}{2}K'_{m,n}\boldsymbol{y} - K'_{m,n}\Theta X^T\boldsymbol{m}_\beta\right) \tag{A.3}$$

A.2. Updates for $q(\omega)$

We have a variational distribution for ω_i of

$$q(\omega_i) = PG(\omega_i|1, c_i),$$

which entails a Kullback-Leibler Divergence between $q(\omega_i)$ and $p(\omega_i)$ of

$$\begin{aligned}
KL(q(\omega_i)||p(\omega_i)) &= \mathbb{E}_{q(\omega_i)}\left[\log q(\omega_i) - \log p(\omega_i)\right] \\
&= \mathbb{E}_{q(\omega_i)}\left[\log\cosh(\frac{c_i}{2}) - \frac{c_i^2}{2}\omega_i + \log PG(\omega_i|1,0) - \log PG(\omega_i|1,0)\right] \\
&= \log\cosh(\frac{c_i}{2}) - \frac{c_i^2}{2}\mathbb{E}_{q(\omega_i)}\left[\omega_i\right] \\
&= \log\cosh(\frac{c_i}{2}) - \frac{c_i}{4}\tanh\left(\frac{c_i}{2}\right).
\end{aligned}$$

As before we look at the gradient of $\mathcal{L}$ with respect to ω_i, where only the gradients of the likelihood term from Equation 3.4 and the KL of ω_i are non zero.

$$\begin{aligned}
\frac{\mathcal{L}}{\partial\omega_i} &= \frac{\partial}{\partial\omega_i}\mathbb{E}_{q(\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})}\left[\log p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})\right] - \frac{\partial}{\partial\boldsymbol{\omega_i}}KL(q(\boldsymbol{\omega_i})||p(\boldsymbol{\omega_i})) \\
&= -\frac{1}{2}\frac{\partial}{\partial\omega_i}\mathbb{E}_{q(\boldsymbol{f},\boldsymbol{\omega},\boldsymbol{\beta})}\Big[\boldsymbol{m}_\beta^T\boldsymbol{x}_i\theta_i\boldsymbol{x}_i^T\boldsymbol{m}_\beta + \theta_i\boldsymbol{x}_i^T S_\beta\boldsymbol{x}_i + \hat{\boldsymbol{m}}_{f_i}^2\theta_i + \theta_i\tilde{K}_{i,i} \\
&\quad + \theta_i\boldsymbol{k}'_{i,m}S_u\boldsymbol{k}'_{m,i} + 2\boldsymbol{m}_\beta^T\boldsymbol{x}_i\theta_i\hat{\boldsymbol{m}}_{f_i}\Big] \\
&\quad - \frac{\partial}{\partial\boldsymbol{\omega_i}}\left[\log\cosh(\frac{c_i}{2}) - \frac{c_i}{4}\tanh\left(\frac{c_i}{2}\right)\right],
\end{aligned}$$

where $\boldsymbol{x}_i$ is the i^{th} column of X. We use $\boldsymbol{k}_{i,m}$ for the i^{th} row of $K_{n,m}$. Further we note $\boldsymbol{k}_{i,m}K_{m,m}^{-1}$ as $\boldsymbol{k}'_{i,m}$, $\boldsymbol{k}'^T_{i,m}$ as $\boldsymbol{k}'_{m,i}$ and the i^{th} entry on the diagonal of $\tilde{K}$ as $\tilde{K}_{i,i}$. Setting in

the first moment of $q(\omega_i)$ from Equation 2.8, we have $\theta_i = \mathbb{E}_{PG(\omega_i|1,c_i)}[\omega_i] = \frac{1}{2c_i}\tanh(\frac{c_i}{2})$

$$\begin{aligned}
\frac{\mathcal{L}}{\partial\omega_i} &= -\frac{1}{2}\frac{\partial}{\partial\omega_i}\Big[2\log\cosh(\frac{c_i}{2}) - \frac{c_i}{2}\tanh\left(\frac{c_i}{2}\right) + \\
&\quad \frac{1}{2c_i}\tanh(\frac{c_i}{2})\Big(\underbrace{\boldsymbol{m}_\beta^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{m}_\beta + \boldsymbol{x}_i^T S_\beta \boldsymbol{x}_i + \hat{\boldsymbol{m}}_{f_i}^2 + \tilde{K}_{i,i} + \boldsymbol{k}'_{i,m}S_u\boldsymbol{k}'_{m,i} + 2\boldsymbol{m}_\beta^T\boldsymbol{x}_i\hat{\boldsymbol{m}}_{f_i}}_{a_i}\Big)\Big] \\
&= -\frac{1}{4}\Bigg[2\tanh(\frac{c_i}{2}) - \tanh(\frac{c_i}{2}) - \frac{c_i}{2}\left(1-\tanh^2(\frac{c_i}{2})\right) \\
&\quad + \frac{a_i}{c_i}\left(-\frac{\tanh(\frac{c_i}{2})}{c_i} + \frac{1}{2}\left(1-\tanh^2(\frac{c_i}{2})\right)\right)\Bigg] \\
&= \frac{1}{4}\left(\frac{a_i}{c_i^2} - 1\right)\left(\tanh(\frac{c_i}{2}) - \frac{c_i}{2}\left(1-\tanh^2(\frac{c_i}{2})\right)\right)
\end{aligned}$$

This gradient is zero if one of the two terms involving c_i becomes zero. The first factor becomes zero if

$$\begin{aligned}
c_i^* = \sqrt{a_i} &= \sqrt{\boldsymbol{m}_\beta^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{m}_\beta + \boldsymbol{x}_i^T S_\beta\boldsymbol{x}_i + \hat{\boldsymbol{m}}_{f_i}^2 + \tilde{K}_{i,i} + \boldsymbol{k}'_{i,m}S_u\boldsymbol{k}'_{m,i} + 2\boldsymbol{m}_\beta^T\boldsymbol{x}_i\hat{\boldsymbol{m}}_{f_i}} \\
&= \sqrt{(\hat{\boldsymbol{m}}_{f_i} + \boldsymbol{x}_i^T\boldsymbol{m}_\beta)^2 + \boldsymbol{x}_i^T S_\beta\boldsymbol{x}_i + \tilde{K}_{i,i} + \boldsymbol{k}'_{i,m}S_u\boldsymbol{k}'_{m,i}}
\end{aligned}$$

Since S_β, S_u and $\tilde{K}$ are definite positive, this is valid. Using the second derivative we can see that this update leads to the optimal $q^*(\omega_i)$.

A.3. Dynamic Programming for Mean Field approach

The updates for the mean and variance for the fully Bayesian MF approach in Equation 3.12 contains the calculation of $X_{-j}^T\boldsymbol{m}_{\beta_{-j}}$ which is the product of a $s \times (d-1)$ matrix with a column vector with $(d-1)$ entries. We can reduce the computational complexity by using

$$X_{-j}^T\boldsymbol{m}_{\beta_{-j}} = X^T\boldsymbol{m}_\beta - \boldsymbol{x}_j^T m_{\beta_j}.$$

Even though the calculation of $X^T\boldsymbol{m}_\beta$ is more more expensive than $X_{-j}^T\boldsymbol{m}_{\beta_{-j}}$, we can update this sum after every update for $q(\beta_j)$. In every update the mean m_{β_j} changes by Δm_{β_j} and thus after every CAVI step for m_{β_j} we update $X^T\boldsymbol{m}_\beta$ with

$$X^T\boldsymbol{m}_\beta \leftarrow X^T\boldsymbol{m}_\beta + \boldsymbol{x}_j^T\Delta m_{\beta_j}.$$

Apart from the calculation of the initial $X\boldsymbol{m}_\beta$, these updates do not depend on the dimensionality d.

One possibility that was further explored, but not used for the experiments, is doing intermediate updates for $q(\boldsymbol{\omega})$. This is not in a standard CAVI algorithm, but theoretically it is possible to leverage the knowledge over Δm_{β_j} and Δs_{β_j}. Calculating the

local parameters from scratch has a complexity of $O(s^2(d+m)+sm^2)$. When updating the local parameters between every update of $q(\beta_j)$, this entails a complexity of at least $O(s^2d^2)$, which does not scale well. Using the same method as above we can reduce the complexity, leading to local updates with complexity $O(s^2)$. From Equation 3.7 we have:

$$c_i^* = \sqrt{\hat{m}_{f_i}^2 + 2\mathbb{E}_q[\boldsymbol{\beta}^T]\boldsymbol{x}_i\hat{\boldsymbol{m}}_{f_i} + \mathbb{E}_q[\boldsymbol{\beta}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{\beta}] + \boldsymbol{k}'_{i,m}S_u\boldsymbol{k}'_{m,i} + \tilde{K}_{i,i}}$$

Here again we have the calculation of

$$\mathbb{E}_q[\boldsymbol{\beta}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{\beta}] = \boldsymbol{m}_\beta^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{m}_\beta + Tr(\boldsymbol{x}_i\boldsymbol{x}_i^TS_\beta)$$

and

$$\mathbb{E}_q[\boldsymbol{\beta}^T]\boldsymbol{x}_i\hat{\boldsymbol{m}}_{f_i} = \boldsymbol{m}_\beta^T\boldsymbol{x}_i\hat{\boldsymbol{m}}_{f_i},$$

which again can be solved by using dynamic programming for these terms. This means we can update our current estimates for c_i^{*2} and updating them after each update of $q(\beta_j)$.
In practice one update for all $q(\beta_j)$ takes longer than several updates with the original approach, but for small mini-batches the updates are more stable.

A.4. Kullback Leibler divergence for augmented Laplace distribution

The KL divergence $\mathrm{KL}(q(\boldsymbol{\beta},\boldsymbol{\lambda})||p(\boldsymbol{\beta},\boldsymbol{\lambda}))$:

$$\begin{aligned} KL(q(\boldsymbol{\beta},\boldsymbol{\lambda})||p(\boldsymbol{\beta},\boldsymbol{\lambda})) &= \mathbb{E}_{q(\boldsymbol{\beta},\boldsymbol{\lambda})}\left[\log q(\boldsymbol{\beta})q(\boldsymbol{\lambda}) - \log p(\boldsymbol{\beta}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})\right] \\ &= \mathbb{E}_{q(\boldsymbol{\beta})}\left[\log q(\boldsymbol{\beta})\right] + \mathbb{E}_{q(\boldsymbol{\beta},\boldsymbol{\lambda})}\left[\log q(\boldsymbol{\lambda}) - \log p(\boldsymbol{\beta}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})\right] \end{aligned}$$

Because of the independence in the priors ($q(\boldsymbol{\lambda}) = \prod q(\lambda_j)$ and $p(\boldsymbol{\beta},\boldsymbol{\lambda}) = \prod p(\beta_j|\lambda_j)p(\lambda_j)$) we can look at one summand of the second term independently of the variational distribution $q(\boldsymbol{\beta})$:

$$\begin{aligned} \mathbb{E}_{q(\beta_j,\lambda_j)}\left[\log q(\lambda_j) - \log p(\beta_j|\lambda_j)p(\lambda_j)\right] \overset{c}{=} \frac{1}{2}E_q\Big[&- 2\log K_{-1/2}(\sqrt{a_j}) - \frac{1}{2}\log(a_j) - 3\log\lambda_j \\ &- \left(a_j\lambda_j + \frac{1}{\lambda_j}\right) + 4\log(\lambda_j) + \frac{1}{\lambda_j} - \log(\lambda_j) + \frac{\beta_j^2}{\sigma_\beta^2}\lambda_j\Big] \\ = \frac{1}{2}\Big(&- 2\log K_{-1/2}(\sqrt{a_j}) - \frac{1}{2}\log(a_j) \\ &+ \mathbb{E}_q\Big[- a_j\lambda_j + \frac{\beta_j^2}{\sigma_\beta^2}\lambda_j\Big]\Big) \end{aligned}$$

We can see that if we set a_i to $\mathbb{E}_q[\frac{\beta_j^2}{\sigma_\beta^2}]$ the last part cancels out. The remaining part of the KL is $\mathbb{E}_{q(\boldsymbol{\beta})}\left[\log q(\boldsymbol{\beta})\right]$, where $q(\boldsymbol{\beta})$ is a Gaussian distribution:

$$\mathbb{E}_{q(\boldsymbol{\beta})}\left[\log q(\boldsymbol{\beta})\right] = -\frac{1}{2}\left(d\log(2\pi) + \log(\det(S_\beta)) + d\right) \tag{A.4}$$

With this we get

$$KL(q(\boldsymbol{\beta},\boldsymbol{\lambda})||p(\beta,\boldsymbol{\lambda})) \stackrel{c}{=} -\frac{1}{2}\Big(\sum_{j=0}^{d} 2\log K_{-1/2}(\sqrt{a_j}) + \frac{1}{2}\log(a_j) + \sum_{j=0}^{d}\mathbb{E}_q\Big[a_j - \frac{\beta_j^2}{\sigma_\beta^2}\Big]\lambda_j + \log(\det(S_\beta)) + \log(\sigma_\beta^2)\Big)$$

A.5. Numerical approximation for Horseshoe

Using the standard approach, we first look at the full conditional

$$\begin{aligned} q^*(\lambda_j) &\propto \exp \mathbb{E}_q[\log p(\lambda_j)p(\beta_j|\lambda_j)] \\ &= \frac{2}{\pi(1+\lambda_j^2)}\frac{1}{\sqrt{2\pi\lambda_j^2}}\exp\left(-\frac{1}{2}\frac{\mathbb{E}_q\left[\beta_j^2\right]}{\sigma_\beta^2\lambda_j^2}\right) \end{aligned}$$

The distribution is handier if we apply a reparametrization, $\gamma = \frac{1}{\lambda^2}$. This leads to

$$q^*(\gamma) = \frac{\frac{1}{1+\gamma}\exp\left(-\frac{1}{2}b_j^*\gamma\right)}{a_{(b_j^*)}},$$

where $a_{(b)}$ is the normalizer:

$$a_{(b_j)} = \int_0^\infty \frac{\gamma}{1+\gamma}\exp\left(-\frac{1}{2}b_j\gamma\right)d\gamma$$

and the optimal b_j^* is $\mathbb{E}\left[\frac{\beta_j^2}{\sigma_\beta^2}\right]$. Because in the update only $\mathbb{E}_{q(\lambda_j)}[\frac{1}{\lambda_j^2}] = \mathbb{E}_{q(\gamma_j)}[\gamma_j]$ is needed we calculate that directly.

$$\mathbb{E}_{q(\gamma_j)}[\gamma_j] = \frac{1}{a_{(b_j)}}\int_0^\infty \frac{\gamma}{1+\gamma}\exp\left(-\frac{1}{2}b_j\gamma\right)d\gamma$$

Looking at the Kullback-Leibler divergence

$$\begin{aligned} KL(q(\boldsymbol{\beta},\boldsymbol{\lambda})||p(\boldsymbol{\beta},\boldsymbol{\lambda})) &= KL(q(\boldsymbol{\beta})||p(\boldsymbol{\beta}|\boldsymbol{\lambda}) + KL(q(\boldsymbol{\lambda})||p(\boldsymbol{\lambda})) \\ &= \mathbb{E}_q(\log q(\boldsymbol{\beta})) + \sum_j KL(q(\gamma_j)||p(\gamma_j)) - \mathbb{E}_q[\log p(\beta_j|\lambda_j)]. \end{aligned}$$

The first part is easily calculable (see Equation A.4) and the rest is found after transforming λ_j to γ_j in the prior.

$$p(\gamma_j) = (\pi\sqrt{\gamma_j}(\gamma_j+1))^{-1}$$

This is then put into the KL.

$$\begin{aligned}\mathbb{E}_q[\log q(\gamma_j) - \log p(\gamma_j) - \log p(\beta_j|\lambda_j)] = \mathbb{E}_{q(\gamma_j)}\Big[&- \log(1+\gamma_j) - \frac{1}{2} b_j \gamma_j - \log a_{(b_j)} \\ &+ \log(1+\gamma_j) + \frac{1}{2}\log(\gamma_j) + \log \pi \\ &- \frac{1}{2}\log(\gamma_j) + \frac{1}{2}\frac{\beta_j^2}{\sigma_\beta^2}\gamma_j\Big] \\ \overset{c}{=} &- \log a_{(b_j)}\end{aligned}$$

Using the already calculated $\mathbb{E}_q[q(\beta_j)]$ from Equation A.4, we get

$$KL(q(\boldsymbol{\beta}, \boldsymbol{\lambda})||p(\boldsymbol{\beta}, \boldsymbol{\lambda})) \overset{c}{=} -\left(\sum_{j=0}^{d} \log a_{(b_j)} + \frac{1}{2}\log(\det(S_\beta))\right) + \frac{1}{2}\mathbb{E}\left[\frac{\beta_j^2}{\sigma_\beta^2} - b_j\right]\gamma_j$$

Where again the last part cancels out if b_j is set to $\mathbb{E}\left[\frac{\beta_j^2}{\sigma_\beta^2}\right]$

A.6. Hyperparameter gradients

Because the Hyperparameters do not occur in $\partial KL(q(\boldsymbol{\beta})||p(\boldsymbol{\beta}))$ the gradient for the ELBO in the fully Bayesian and the MAP Setting are very similar. Thus both losses are written here as $\mathcal{L}$. The equations here factor in the variances of $q(\boldsymbol{\beta})$ and $p(\boldsymbol{\beta})$. When using the MAP estimate the corresponding equations these terms are zero. In order to find the gradient of the Loss $\mathcal{L}$

$$\frac{\partial \mathcal{L}(y|X, \mathcal{H})}{\partial \mathcal{H}} = \frac{\partial \mathbb{E}_q[\log p(y|u, \omega, \beta)]}{\partial \mathcal{H}} + \frac{\partial KL(q(u)||p(u))}{\partial \mathcal{H}}$$

the gradient is broken down into sub-problems. $\mathcal{H}$ are the Hyperparameters of the Kernels here.
The only factor directly affected by $\mathcal{H}$ is the Kernel K. So $\frac{\partial K}{\partial \mathcal{H}}$ has to be found first.
The RBF kernel function is:

$$k(x, y) = \exp(-\frac{(x-y)^2}{2l^2})$$

Where l is the length-scale. The derivation to the hyperparameter is:

$$\frac{\partial k}{\partial l} = \frac{(x-y)^2}{l^3}\exp(-\frac{(x-y)^2}{2l^2})$$

Because the variance is a pre-factor for the kernel, its derivation is the kernel without the pre-factor.

With the parametrization of the hyperparameters as $\tilde{h}_k = \exp h_k$, the chain rule the derivation is:

$$\frac{\partial K}{\partial h_k} = \frac{\partial k}{\partial \tilde{h}_k}\frac{\partial \tilde{h}_k}{\partial h_k} = \frac{\partial k}{\partial \tilde{h}_k}\exp(h_k)$$

With these components retrieved, we can look at the occurrences of K.

$$\begin{aligned}
\frac{\partial K'_{n,m}}{\partial \mathcal{H}} &= \frac{\partial K_{n,m}}{\partial \mathcal{H}}K^{-1}_{m,m} - K_{n,m}K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}}K^{-1}_{m,m} \\
\frac{\partial K'_{n,m}K_{m,n}}{\partial \mathcal{H}} &= 2\frac{\partial K_{n,m}}{\partial \mathcal{H}}K^{-1}_{m,m}K_{n,m} - K_{n,m}K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}}K^{-1}_{m,m}K_{m,n} \\
\frac{\partial \log|K|}{\partial \mathcal{H}} &= Tr\left(K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}}\right) \quad \text{According to [40] Appendix}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \mathcal{H}}\log \mathbb{E}_q[p(y|u,\omega,\beta)] \propto \frac{1}{2}\frac{\partial}{\partial \mathcal{H}}\Big(& y^T\hat{m}_f \\
& - Tr(\Theta\tilde{K}) - Tr(K'_{m,n}\Theta K'_{n,m}S_u) \\
& - (\hat{m}_f + X^T m_\beta)^T\Theta(\hat{m}_f + X^T m_\beta)\Big) \qquad \text{(A.5)} \\
\propto \frac{1}{2}\Big(& y^T\frac{\partial K'_{n,m}}{\partial \mathcal{H}}m_u - Tr(\Theta\left[\frac{\partial K_{n,n}}{\partial \mathcal{H}} - \frac{\partial K'_{n,m}K_{m,n}}{\partial \mathcal{H}}\right]) \\
& - Tr(2K'_{m,n}\Theta\frac{\partial K'_{n,m}}{\partial \mathcal{H}}S_u) \\
& - 2(\hat{m}_f + X^T m_\beta)^T\Theta\frac{\partial K'_{n,m}}{\partial \mathcal{H}}m_u\Big) \qquad \text{(A.6)} \\
\frac{\partial}{\partial \mathcal{H}}KL(q(u)|p(u)) = 1/2\frac{\partial}{\partial \mathcal{H}}& \left(Tr(K^{-1}_{m,m}S_u) + m_u^T K^{-1}_{m,m}m_u + \log(|K_{m,m}|)\right) \\
= 1/2\Big(& - Tr(K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}}K^{-1}_{m,m}S_u) \\
& - m_u^T K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}}K^{-1}_{m,m}m_u + Tr(K^{-1}_{m,m}\frac{\partial K_{m,m}}{\partial \mathcal{H}})\Big) \qquad \text{(A.7)}
\end{aligned}$$

When joining this the complete gradient is as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}(y|X,\mathcal{H})}{\partial \mathcal{H}} = \frac{1}{2}\Bigg(& y^T \frac{\partial K_{n,m} K_{m,m}^{-1}}{\partial \mathcal{H}} m_u - Tr(\Theta \left[\frac{\partial K_{n,n}}{\partial \mathcal{H}} - \frac{\partial K_{n,m} K_{m,m}^{-1} K_{m,n}}{\partial \mathcal{H}}\right]) \\ & - Tr(2K_{m,m}^{-1} K_{m,n} \Theta \frac{\partial K_{n,m} K_{m,m}^{-1}}{\partial \mathcal{H}} S_u) \\ & - 2(m_u^T K_{m,m}^{-1} K_{m,n} + X^T m_\beta)^T \Theta \frac{\partial K_{n,m} K_{m,m}^{-1}}{\partial \mathcal{H}} m_u \\ & - Tr(K_{m,m}^{-1} \frac{\partial K_{m,m}}{\partial \mathcal{H}} K_{m,m}^{-1} S_u) - m_u^T K_{m,m}^{-1} \frac{\partial K_{m,m}}{\partial \mathcal{H}} K_{m,m}^{-1} m_u \\ & - Tr(K_{m,m}^{-1} \frac{\partial K_{m,m}}{\partial \mathcal{H}}) \Bigg) \end{aligned} \tag{A.8}$$