Student-ID:	11913169
Degree Program:	Master of Science in Economics (Science Track)
Examiner:	Peter Knaus, PhD
Submission date:	TBA

Triple-Gamma-Regularization

A Flexible Non-Convex Regularization Penalty based on the Triple-Gamma-Prior

by

Lucas Unterweger GitHub
(Student-ID: 11913169)

Abstract

Add abstract as soon as the thesis is 100% finished.

Acknowledgements

This completion oft his thesis would not have been possible without so many amazing people in my life who supported me throughout the process of writing this master's thesis and the master program as a whole. I want to thank my mum, Daniela, and my dad, Werner, who always picked me up when I lost my motivation to continue and who convinced me to keep going time after time. I want to thank my brother, Tobias, who always readily listened to me complaining for hours and hours on end about things he probably barely understood and yet always agreed with me about my computational issues while doing the simulations. And most of all I want to thank my advisor, Peter, who always supported me from the day I randomly dropped an e-mail into his e-mail inbox about becoming his tutor in econometrics, although not having met once during the distance learning phase of the COVID pandemic, up until today. From working as a tutor in his team, writing a bachelor thesis, and to the support during this master thesis now. Even though sometimes living 12h time differences apart, one in Boston the other in Manila, he always answered any questions I had and supported me in whatever way he could and I want to sincerely thank him for that!

Contents

1 Introduction

Willam of Ockham, born in Ockham, Surrey, probably lived between 1287 and 1348 and is nowadays recognized as a pre-eminent philosopher of the middle ages. Although his name itself is no common knowledge, a principle carrying his name is: *Ockham's Razor*. Interestingly, the main formulation of the principle (*Entia non sunt multiplicanda praeter necessitatem* [plurality should not be posited without necessity]) can not be traced back to Ockham directly, but variations of it can be found in Ockham's writings. Since then nonetheless, the principle has long been used by statisticians and other researchers as a a scientific credo to capture the notion that "the simpler of two explanations is to be preferred" (Lazar, 2010). Recent decades and its advancements in information technology have opened the gates to seemingly unlimited amounts of data and information. Whether it is data about the carbon dioxide levels near bus stations in a city, the intraday trading volume of a specific financial asset or the rhythm of a beating heart throughout a day, new techniques and approaches in measuring digital and natural phenomena have enables researchers all around the globe to study previously unknown effects, test novel hypotheses and find relationships between real world process that would have otherwise remained hidden from the eye of human civilization.

Yet, these advancements come at a cost. Conventional statistical models like the *ordinary least squares (OLS)* have troubles handling statistical problem in higher dimensions. When the relative number of features to the number of data points in a statistical model is very high, the estimates of these models suffer from high variance and thus generalize poorly. Frequentist statisticians have developed penalized regression approaches to battle these issues by incorporating a penalty term into the optimization problems used to estimate the coefficients of linear models. In recent decades, various penalty terms have been published with each of its own advantages and disadvantages, yet to this day no a single method has emerged to triumph above them all (F. Wang et al., 2020).

Bayesian statisticians have developed their own tools to handle high-dimensional statistical problems and commonly use specific probability distributions to battle model complexity by shrinking the estimates of less important variables towards zero (thus the name *shrinkage prior*). Although these two fields of statistical thought use different approaches to tackle the same problem, a striking mathematical connection exists between those two methods. A bridge which allows mathematicians to derive a regularization penalty from specific prior distributions, which raises the question: "Does there exist a already known prior distribution in the Bayesian setting which can be used to derive a regularization penalty which might perform better than existing penalties?".

A prior distribution which might have these suitable properties is the unifying shrinkage prior developed by Cadonna et al. (2020) called *Triple-Gamma-Prior*. Its unifying properties and the fact that a closed-form solution for the marginal prior exists might lead to favourable properties in the realm of frequentist statistics, but before that it is necessary to have a closer look at what the exact problem is that Bayesian shrinkage prior and regularization approaches are trying to solve. An overview of this along with a theoretical basis of the necessary concepts will be presented in section 2.

The remaining thesis is structure as follows: Section 3 will provide a literature review about the history of regularization and advancements in the field along with an overview of established concepts and their respective advantages and disadvantages. Section 4 will quickly discuss the mathematical building block for this thesis by showcasing the bridge between regularization and Bayesian shrinkage prior as well as introduce the Triple-Gamma-Prior by Cadonna et al. (2020) as the shrinkage prior of choice. Section 5 will then derive the proposed penalty by utilizing the bridge explained in section 4. Section 6 will then apply the newly derived regularization penalty and apply it in the context of a simulation study to see its effects and possible fields of application. The remaining sections 7 and 8 will discuss shortcomings of the method and possible extensions and end with a summarizing conclusion.

2 Theoretical Section

Generally speaking, a major part of statistical learning deals with trying to describe a certain output variable Y with a set of input variables $X_1, \cdots, X_p$ by trying to find a functional form f which uses the given information in the inputs and - ideally - describes the hidden relationship between Y and the inputs X_i as accurately as possible. However, it comes as no surprise that the functional form f depends on the statistical problem at hand. What type of data has been collected? Are we assuming a linear or non-linear relationship between the predictors? How much data is available and can its quality be guaranteed? But more importantly, it is necessary to ask the question whether the goal of the statistical learning method is *inference* or *prediction*.

The first of these two goals - *inference* - aims at understanding the relationship that may or may not exist between the input variables X_i and the output Y. Especially applied sciences like Economics, Psychology and Medicine often try to find a (causal) relationship within their theoretical framework to evaluate a policy, a medication or a new form of therapy. Linear models for example often provide a simple and straightforward framework which provide the scientist with interpretable effects. *Prediction* on the other hand aims at forecasting the output variable Y as accurately as possible and using every bit of information that is available, but disregards interpretability. (Hastie et al., 2009, p. 21) These goals can be summarized by viewing it as a decision between *prediction accuracy* and *model interpretability*, which has thoroughly been explained by Hastie et al. (2009).

For the purpose of this thesis, the following chapters will restrict itself to the case of linear models, hence where assume that the functional form f is linear in its inputs. The commonly known *least squares estimator* is one way of estimating such a functional form and due to its many desirable properties has gained popularity in various scientific fields.

2.1 Model Complexity and the Problem of Under- and Overfitting

- BIAS VARIANCE TRADE-OFF - DIMENSIONALITY PROBLEM
General problem: Fit vs. Complexity. What to choose and where is the optimal balance?

In general, a linear regression model fitted with least squares on a sufficient amount of data points ($n \gg p$) will produce estimates which both have low bias and low variance and thus tends to perform well out of sample. However, problems arise with this approach arise when the sample size decreases, because the variance in the estimates will increase.

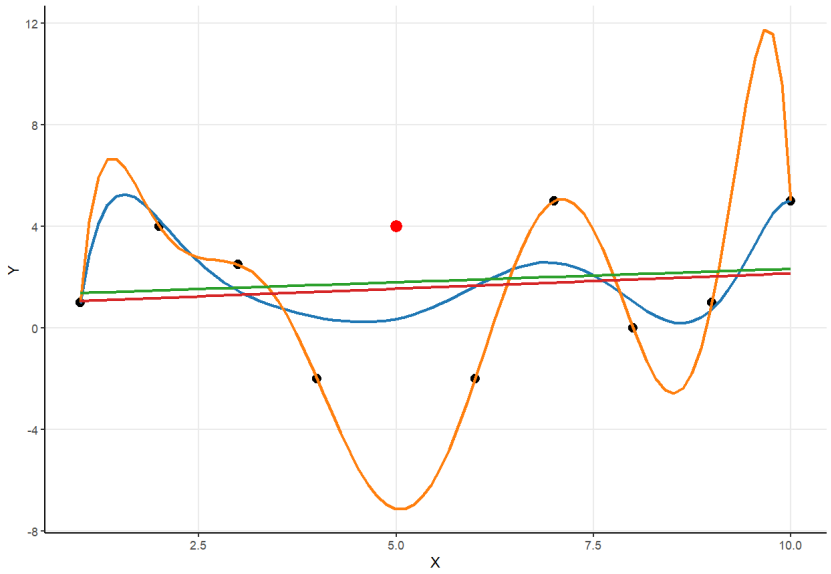


Figure 1: 1st and 8th order polynomial fit to data
(Green & Blue Lines fitted with red data point; Orange & Red Lines fitted without).

The main idea behind this can be easily be visualized by trying to fit a polynomial curve on a two dimensional space and then altering the available sample for polynomials of different order.

Such a plot can be found in figure 1. Here, the black points represent the main data sample plotted on a two-dimensional Y-X-space. The red dot plotted at $(5, 4)$ represents the additional data point which is used to alter the respective sample. Polynomials of specific orders can now be used to emulate certain levels of model complexity. For example, a first order polynomial has two coefficients to estimate: the intercept β_0 and the slope β_1. A polynomial of order of order eight has nine coefficients to estimate. What can now be seen in figure 1 is the change in estimates if we include/exclude the additional red data point. A first-order polynomial needs at least two data points and as $n = 9 > p = 2$, the overall fit of the polynomial doesn't change drastically with the additional tenth data point. In the case of the eighth-order polynomial, we need at least nine data points to create a fit as we have nine predictors (one intercept and one coefficient for each of the eight powers). In this case $n = p$ and adding another data point drastically changes the estimates coefficients, which means that the model won't perform well in out-of-sample scenarios. This emphasizes the importance a sufficient sample size when fitting linear models. And this leads us to the core field of study of this thesis: What are ways to improve the generalization of a linear model in scenarios where the sample size is not sufficiently large enough or high-dimensionality becomes a problem?

As already pointed out by Fan and Li (2006) almost 20 years ago, "high dimensional data analysis will be the most important research topic in statistics in the 21st century"

and the advancements in data availability in recent decades have supported their hypothesis. The main problem to solve is how to accurately approach the problem of feature selection and make models generalize better. A first approach could be to increase the sample size to re-establish the unbiasedness of methods like OLS, however either the problem at hand involves a scenario with only few data points to begin with, or due to an extensive number of features, the collection of a sufficient amount of data points is practically not feasible. Another approach could be to use common variable selection methods like *AIC* and *BIC*, but as Fan and Li (2006) point out, "[t]raditional variable selection such as C_p, AIC and BIC involves a combinatorial optimization problem, which is NP-hard, with computational time increasing exponentially with the dimensionality." A third and computationally more feasible approach involves a concept called *penalized regression*, more commonly known as *Regularization*.

2.1.1 Regularization

Focusing on option three to battle model complexity, regularization methods have been studied thoroughly since the 1990s. However, the emergence of data science - and especially machine learning - as a standalone field of study has led to a broader meaning of the term *regularization*. This phenomenon has been discussed by Kukačka et al. (2017), where the authors establish a taxonomy to distinguish between multiple different definitions. In the traditional sense, as can be seen in Hastie et al. (2009, pp. 167–170), *regularization* refers to a general class of problems of the form

$$\min_{f \in \mathcal{H}} \left\{ \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \right\}$$

where $L(.)$ refers to a loss function defined as some function of the true values and the predicted values and $J(f)$ is penalty based on the chosen functional from a space of functions $\mathcal{H}$. In the context of penalized linear regression, this is equivalent to finding the set of risk minimizing coefficients $\hat{\beta}$ from the set of all possible combinations of coefficients β. **(Missing Citation)**Thus, resulting in the general class of regularization problems of the form:

$$\min_{\beta \in \boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} L(y_i, f_\beta(x_i)) + \lambda J(\beta) \right\}$$

where $f_\beta(x_i)$ is a linear function of the inputs x_i parametrized by the coefficients β. Hence, in this setting, regularization deals with penalizing the risk function based on the value of the chosen set of coefficients.

However, this only describes a subset of *regularization* methods as stated by Kukačka et al. (2017). The authors use a more general definition of regularization:

Defintion 1. Regularization is any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set.

Building on that, they split up the majority of *regularization* methods into (1) methods applied to the data set like transformations or modifications of the inputs, (2) methods altering the selected model family , (3) methods applied to the error/loss function $L(y_i, f_\beta(x_i))$, (4) methods applied to regularization/penalty term as described above and (5) alterations of the optimization procedure itself.

Unsurprisingly, this thesis is concerned with the fourth group of regularization methods, which add a penalty/regularization term $J(\beta)$ into the risk function, but before advancing to literature that deals with this kind of problem, it is necessary to establish a terminology which will be used throughout this thesis. This terminology will be the following:

Let $\mathcal{D}$ be a training data set with $n \in \mathbb{N}$ observations, where every consists of a target variable $y_i \in \mathbb{R}$ along with a number of corresponding inputs $x_i \in \mathbb{R}$. Given a linear function $f_\beta(x_i)$ of the inputs parametrized by coefficients $\beta \in \boldsymbol{\beta}$, $L(y_i f_\beta(x_i))$ is the **Loss** function measuring the discrepancy between the actual target y_i and the output of the linear function $f_\beta(x_i)$. According to Vapnik (1991), the **Empirical Risk Functional** is then

$$R_{emp}(\beta) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\beta(x_i)).$$

Regularization in this thesis' context refers then to adding some penalty function $J(\beta)$ dependent on the set of parameters β, multiplied by some weighting parameter λ, to the empirical risk functional $R_{emp}(\beta)$. Thus, $R_{reg} = R_{emp}(\beta) + \lambda \cdot J(\beta)$. This results in the overall optimization problem

$$\begin{aligned} &\arg\min_{\beta \in \boldsymbol{\beta}} \{R_{reg}(\beta)\} \\ &= \arg\min_{\beta \in \boldsymbol{\beta}} \{R_{emp}(\beta) + \lambda \cdot J(\beta)\} \\ &= \arg\min_{\beta \in \boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} L(y_i, f_\beta(x_i)) + \lambda \cdot J(\beta) \right\} \end{aligned} \tag{1}$$

It is important to note here that as $J(\beta)$ is only a function of the coefficients β and neither the targets y_i nor the inputs x_i. It only affects the generalization error of the model, not the training error given by the empirical risk functional $R_{emp}(\beta)$.

2.2 Bayesian View on Battling Model Complexity using Shrinkage Priors

To build on the mathematical connection between the Bayesian and Frequentist approach to battle model complexity, which will be used as the building block of this thesis

later on, it is necessary to make a quick detour and quickly recap on how Bayesian statisticians tackle model complexity.

Explain Shrinkage Prior

3 Literature Review

The concept of a penalized regression has been around for quite some time and been studied widely in various fields of scientific research. Arguably, this methodological approach to penalized regression started with the publication of two pieces of literature published by Arthur Hoerl and Robert Kennard in 1970 (A. E. Hoerl & Kennard, 1970a, 1970b). With these two papers the authors introduced the widely known *Ridge Regression*, which has been developed from the previously known concept of Ridge analysis. In its core, the authors were trying to tackle the problem of high variances of the regression coefficients in high-dimensional problem settings. This shrinkage estimator, which uses the squared coefficient as a penalty term, "attempt[s] to shrink the coefficients to reduce these variances, while adding some bias." (R. W. Hoerl, 2020) This closely resembles the previously discussed issue of the *Bias-Variance-Tradeoff*, which has been discussed in the *Under- and Overfitting* chapter in section 2 (Roger W. Hoerl, Arthur Hoerl's son, published a historical overview of the development of the concept of *Ridge Regression* in 2020 (R. W. Hoerl, 2020)). The closed from solution of the Ridge estimator is given by

$$\hat{\beta}_{Ridge} = \left(X^T X + \lambda I\right)^{-1} X^T y$$

, which adjusts the OLS estimator by shifting the main diagonal entries of the design matrix by λ ($\lambda \geq 0$). It can be shown that this closed form estimator is equivalent to a Lagrangian problem of the following form (van Wieringen, 2015):

$$\hat{\beta}_{Ridge}(\lambda) = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\}$$

This resembles an example of the above defined regularization framework with squared residual loss and a penalty term of the form $\|\beta\|_2^2$, which only depends on the parameter β. In case of λ being equal to zero, this reduces to the *maximum likelihood (ML) estimator*.

The publications of Arthur Hoerl and Robert Kennard have led to further advancements, although it took more than 25 years, in shrinkage estimation or related concepts. One concept which is almost as famous *Ridge Regression* is the *Least Absolute Shrinkage and Selection Operator*, more commonly know as *LASSO*, developed by Tibshirani (1996). He argues that the two at the time most prominent shrinkage methods - Ridge and Subset Selection - both have their drawbacks. Ridge regression on the one hand is an optimization problem which continuously shrinks coefficients towards zero, but doesn't select them in a discrete sense, which makes it hard to interpret these models. Subset Selection on the other hand chooses variables in a discrete sense - a variables either stays within the model or it doesn't - and thus creates easily interpretable models, but "[s]mall changes in the data can result in very different models being selected and this can reduce its prediction accuracy." (Tibshirani, 1996) *LASSO* is trying to combine both methods'

advantages by using $\|\beta\|_1$ as a penalty term.

LASSO and to some extend *Ridge* can be viewed as a special case of a l_p-norm regularization with corresponding values for p ($p = 1$ for LASSO and $p = 2$ for Ridge) (Frank & Friedman, 1993).

$$\|\beta\|_p = \left(\sum_{i=1}^{p} |\beta_i|^p \right)^{1/p}$$

Work published by researchers in the nineties, like the previously mentioned Frank and Friedman (1993) or Fu (1998), as well as more recent literature like F. Wang et al. (2020) have repeatedly shown there is no go-to-method to tackle regularization problems, as the effectiveness of a specific approach highly depends on the data siuation at hand. Due to this particular situation in the literature, several other methods have been proposed in the recent years and decades. An approach combining *Ridge* and *LASSO* two methods is called *Elastic Net* regularization and has been developed by Zou and Hastie (2005). The authors there elaborate on some of the shortcomings of the LASSO method. For example, in a special case where there are more predictors p than data points n ($p > n$), LASSO only selects up to n variables due to the nature of the convex optimization problem. Should several of the included variables be highly *pairwise* correlated with each other, *LASSO* tends to only select on of these variables. In its core, *Elastic Net Regularization* linearly combines the penalty terms of *Ridge* and *LASSO* regularization, yielding a loss function of the form:

$$J(\beta) = \lambda_1 \|\beta\|_2^2 + \lambda_2 \|\beta\|_1$$

The authors have shown that, especially when it comes to encouraging the aforementioned grouping effects, *Elastic Net* tends to perform better than the *LASSO*.

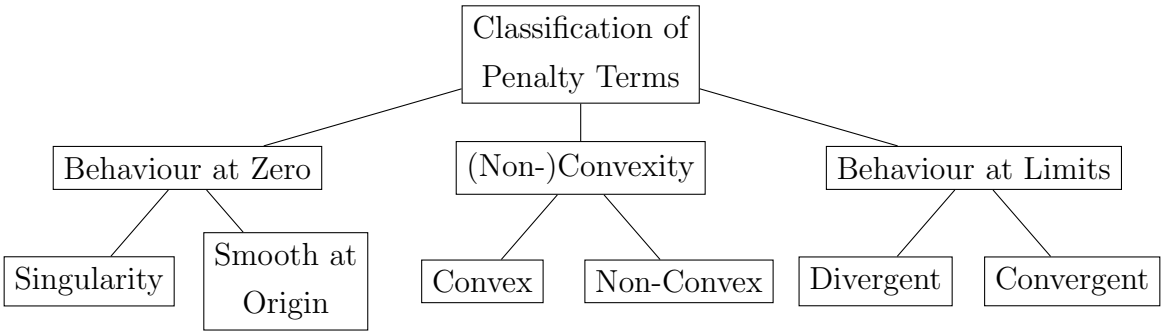
Recent years however have opened up a new subfield of approaches to regularization. Methods like *LASSO, Bridge* (Frank & Friedman, 1993), *Ridge* and *Elastic Net*[1] are convex functions in its parameters and are thus usually classified as *Convex Regularization Penalties*. Recently, the literature has shifted towards penalties which are non-convex functions in its parameters, usually called *Non-Convex Regularization Penalties*[2]. One recent example of such a penalty includes John et al. (2022), who proposed a penalty structure called *Gaussian penalty* and which is based on a Gaussian-like function by using $J(\beta) = 1 - e^{-\kappa\beta^2}$. Another method proposed by Y. Wang and Zhu (2016) is called

[1] Elastic Net, due to its mathematical definition, can be view as a generalization of *Ridge* and *LASSO*.

[2] Note: They are called *non-convex* penalties and not *concave* penalties, because non-convexity does not necessarily imply concavity.

the *Atan penalty* - or *Arctan* penalty - and makes use of the favourable properties of the *Arctan* function by using the penalty $J(\beta, \gamma) = (\gamma + \frac{2}{\pi}) \arctan(\frac{|\beta|}{\gamma})$. Several others include *SCAD* (Fan & Li, 2001), *MCP* (Zhang, 2010) or *Laplace* (Trzasko & Manduca, 2009) penalties.

Seeing this vast array of different pieces of literature immediately raises the question on advantages and disadvantages of specific methods and why so many have established itself in this particular field of study. Keeping in mind that the effectiveness of a method still highly depends on the data situation at hand, it is still important to distinguish methods based on its mathematical properties. John et al. (2022) have distinguished methods based on two broad criteria: (1) How does a penalty behave in small neighbourhoods around zero? (Is it smooth or singular at origin?) and (2) Is the underlying function convex or non-convex? To better incorporate the proposed concept of this thesis into this body of literature, I am proposing a third property to distinguish penalty terms: (3) How does the function behave in the limits? (Does it converge or diverge?)



As already mentioned earlier in section 2, a penalty function is a function dependent on the parameter of the model and it does not depend on the data at hand. Thus, changes in your data set does not effect the penalty directly, but only the overall optimization problem through the empirical risk functional R_{emp}. The following table classifies some of the existing concepts based on the three criteria mentioned earlier. A visualisation of them can be found in figure 7 in chapter 5.2.

Depending on what property a penalty has, it can perform specific tasks or provide challenges in application. As thoroughly described by (John et al., 2022), penalties with a singularity at origin tend to be suitable when the goal of the statistical analysis is variable selection but "could pose theoretical and computational challenges when the focus is on regularization alone without variable selection." In addition, it has been shown in the literature that "objective functions which are a sum of a nonconvex, singular-at-origin penalty function coupled with either a smooth nonconvex loss function or a nonsmooth convex loss function, often fails to satisfy a theoretical condition known as

'Clarke regularity'." (A result presented in **QiCuiLiuPang2021<empty citation>**) Convex penalties are generally easier to implement using common optimization algorithms than non-convex penalties and, more importantly, penalties which are divergent and its limits, which most convex penalties are, yield biased results. Due to the fact that, when using divergent and convex penalties like LASSO and Ridge, the additional penalty for larger estimates keeps rising with estimates which deviate from zero, estimates are getting artificially pulled towards zero and are thus biased. Non-Convex penalties, which are usually convergent in its limits towards $-\infty$ and $+\infty$, tend to yield unbiased results as the additional penalty becomes zero as soon as a certain threshold is crossed (Again, referring to the visualisation in figure 7).

Still have to figure out how to fit this table.

Penalty		Classification		
Penalty	Reference	Behaviour at Origin	(Non-)Convexity	Limits
LASSO	Tibshirani (1996)	Singular	Convex	Divergent
Ridge	A. E. Hoerl and Kennard (1970b)	Smooth	Convex	Divergent
Gaussian	John et al. (2022)	Smooth	Non-Convex	Convergent
Ar(c)tan	Y. Wang and Zhu (2016)	Singular	Non-Convex	Convergent
Elastic-Net	Zou and Hastie (2005)	Singular	Convex	
Bridge	Frank and Friedman (1993)	Both		
MCAP	Zhang (2010)	Singular	Non-Convex	
SCAD	Fan and Li (2001)	Singular	Non-Convex	
Laplace	Trzasko and Manduca (2009)	Non-Convex		
Triple-Gamma	-	Depending on parameters	Non-Convex	Divergent

Table 1: Classification of several Penalties

4 Bayesian-Frequentist Duality of Ridge and LASSO Regression

The main motivation for this thesis stems from a striking duality that exists between Bayesian shrinkage priors (discussed in chapter 2.2) and the frequentist approach using regularization. Both approaches aim at tackling the issue of model complexity and over-fitting by making it necessary for the data to be more convincing that the value of an estimate is statistically significant different from zero. As mentioned, Bayesian statistics

use specific prior distributions with a usually a lot of mass around zero and heavy tails, whereas frequentist statisticians usually alter their optimization problems incorporating penalty terms into into their loss functions. On a first glance, these two approaches have no immediate mathematical connection, however this turns out to be wrong.

In a Bayesian setting, one usually assumes that the parameter vector β has a prior distribution $p(\beta$. By multiplying it with the likelihood of the data $f(y|X,\beta)$ and by utilizing Bayes' rule, one retrieves the posterior distribution of the parameter distribution, up to a proportionality constant:

$$p(\beta|y,X) \propto f(y|X,\beta) \times p(\beta)$$

It turns out, as can be seen in Hastie et al. (2009, pp. 248–250) and van Wieringen (2015) among others, that when choosing certain prior distributions and using standard assumptions in Bayesian modelling (ie the individual parameters of the parameter vector are independent a prior and the errors of the standard linear model are drawn from a normal distribution), that the moments of the posterior distributions correspond to the point estimates of a regularization approach in a frequentist setting. As this idea is essential to the entire thesis, a shortened derivation of this duality for the case of *Ridge* regression will now be shown.

Assuming that a response variable y_i follows a linear model with k variables and errors $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$, then the likelihood of the data is given by

$$\mathcal{L}(y|X,\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^2\right).$$

Under the previously mentioned assumption that $p(\beta) = \prod_{i=1}^{k}$ and by choosing a Gaussian distribution with mean zero and variance τ^2, we gain the prior distribution

$$p(\beta) = \frac{1}{(2\pi\tau^2)^{p/2}} \exp\left(-\frac{1}{2\tau^2}\|\beta\|_2^2\right).$$

Putting both together using Bayes' theorem yields and taking the log of the distribution yields

$$\log(p(\beta|y,X)) \propto -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^2 - \frac{1}{2\tau^2}\|\beta\|_2^2.$$

By viewing $\frac{1}{2\tau^2}$ as the weighting parameter λ, which controls the strength of the regularization. The *maximum-a-priori* estimate $\hat{\beta}_{MAP}$, which can be viewed as trying to find the $\arg\max_{\beta}$ of the log-posterior, can be used to construct the following optimization problem:

$$\hat{\beta}_{MAP} = \arg\min_{\beta} \left\{ \frac{1}{2\sigma^2} \sum_{i=1}^{n} \epsilon_i^2 + \frac{1}{2\tau^2} \|\beta\|_2^2 \right\}$$

This optimization problem constitutes the connection between the Bayesian and the frequentist approach as it can be interpreted as (1) finding the mode of the posterior distribution when using a Gaussian prior distribution and (2) finding the parameters which minimize the loss when using a Ridge regression with $\lambda = 1/(2\tau^2)$. The same connection exists for the case of *LASSO* regression which can be derived using the same procedure and a double-exponential - also called *Laplace* - distribution with mean zero and some scale parameter τ as the prior for the parameter vector β.

4.1 Triple-Gamma-Prior by Cadonna et al. (2020)

This opens up a path of possibility to utilize this connection and derive a new regularization penalty by using a different shrinkage prior. Recently, a new development in the are of shrinkage priors has been made by Cadonna et al. (2020), who proposed a new shrinkage prior called the *Triple-Gamma-Prior* which has several advantageous properties and also comes with a closed-form solution of the marginal prior distribution for β, which sets the building block for this thesis' research question:

> Can the closed-form marginal distribution of the Triple-Gamma-Prior be used to derive a new regularization penalty and do its advantages carry over into the frequentist framework?

Yet, before diving into the mathematical core of this thesis which derives said concept, it is necessary to have a look at the Triple-Gamma-Prior in depth to explain why this particular prior might prove itself to provide a useful building block for a novel regularization penalty.

As already mentioned above, BMA and shrinkage priors are two Bayesian approaches to deal with statistical problems in high-dimensional settings. The prior is generally constructed for variance selection in sparse state space and TVP models and is named the Triple-Gamma-Prior as it's representation is given by three Gamma distributions. A key advantage of this prior is its unifying property. As cam be seen in table 1 in Cadonna et al. (2020), special cases of the prior, depending on certain values in its hyperparameters, are the Normal-Exponential-Gamma, the Horseshoe and the double Gamma prior. In addition, it can also be seen as a special case of the LASSO prior. Another advantage is its function as a bridge between spike-and-slap and continuous shrinkage prior as the hyperparameters can be used both for steering the shrinkage effect on noise while also preventing overshrinking the signal. Computationally the shrinkage prior performs better

than commonly used spike-and-slab priors as it is a continuous shrinkage prior which can more easily implemented using Markov-Chain-Monte-Carlo (MCMC) methods.

Mathematically, the prior builds on the work by Bitto and Frühwirth-Schnatter (2019) in which the authors introduced the *double gamma prior* with the following hierarchical representation:

$$\theta_j|\xi_j^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{2\xi_j^2}\right), \quad \xi_j^2|a^\xi, \kappa_B^2 \sim \mathcal{G}\left(a^\xi, \frac{a^\xi\kappa_B^2}{2}\right)$$

It is important to note here, that the *double gamma prior* is a global-local shrinkage prior, where the global hyperparameter κ_B^2 is the same for all parameters θ_j and the local hyperparameter ξ_j^2 is the local component. The *Triple-Gamma-Prior* now adds a third Gamma prior into this framework by replacing κ_B^2 in the second-level prior with κ_j^2, which follows

$$\kappa_j^2|c^\xi, \kappa_B^2 \sim \mathcal{G}\left(c^\xi, \frac{c^\xi}{\kappa_B^2}\right)$$

The core property which is relevant to this thesis can be found in *Theorem 1 (b)* in Cadonna et al. (2020) which shows that the *Triple-Gamma-Prior* has a closed-form marginal prior distribution for the parameter θ_j, or to stick with the terminology of this thesis β_j. The prior is given by

$$p(\sqrt{\beta_j}|\phi^\xi, a^\xi, c^\xi) = \frac{\Gamma(c^\xi + \frac{1}{2})}{\sqrt{2\pi\phi^\xi} \cdot B(a^\xi, c^\xi)} \cdot U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j}{2\phi^\xi}\right),$$

where $\phi^\xi = \frac{2c^\xi}{\kappa_B^2 a^\xi}$ and $U(a, b, z)$ is the confluent hyper-geometric function of the second kind of Tricomi (1947) given by

$$U(a, b, z) = \frac{1}{\Gamma(a)} \int_0^\infty e^{-zt} t^{a-1}(1+t)^{b-a-1} dt$$

According to the mathematical connection described above, this prior can now be used to derive a novel regularization penalty.

Should I talk more extensively about the TGP? Is there something important I forgot to mention?

5 Model Setup and Derivation

Coming to the theoretical framework of the *triple-gamma-regularization*, let's assume we have a response variable y and p predictors along with n data points. More formally, let $y = [y_1 \quad y_2 \cdots y_n]^T$ and $x_i = [x_{i1} \quad x_{i2} \cdots x_{in}]^T$ with $\forall i \in \{1, ..., n\} : y_i, x_i \in \mathbb{R}$. Here, x_i is the i-th predictor, thus resulting in the design matrix $X = [x_1 \quad x_2 \cdots x_p]$.

Starting from the Bayesian framework, the standard linear regression model is given by

$$y_i = x_i^T \cdot \beta + \varepsilon_i \quad i \in \{1, ..., n\}.$$

with the assumed distribution of $\varepsilon_i \sim N(0, \sigma^2)$. Thus it follows that $y \sim N_n(X\beta, \sigma^2 I)$. The posterior distribution of the parameter vector β, according to Bayes' Rule, is then proportional to the product of the likelihood of the data and the prior distribution, which can be seen in equation 2.

$$p(\beta|y, X, \sigma^2) \propto \mathcal{L}(y|\beta, \sigma^2, X) \times p(\beta) \tag{2}$$

As stated above, each data point y_i is assumed to be identically and independently drawn from a normal distribution with mean $X\beta$ and variance σ_i^2, thus:

$$\begin{aligned}
\mathcal{L}(\mathbf{y}|\beta, \sigma^2, X) &= \prod_i^n p(y_i|\beta, \sigma^2, X_i) \\
&= \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x_i - \mu}{\sigma}\right)^2\right) \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta)\right)
\end{aligned} \tag{3}$$

The log of the likelihood function is then given by

$$\begin{aligned}
\log \mathcal{L}(\mathbf{y}|\beta, \sigma^2, \mathbf{X}) &= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta)\right)\right) \\
&= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}}\right) + \log\exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta)\right) \\
&= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) \\
&\propto -\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) = -\frac{1}{2\sigma^2}\left\|\mathbf{y} - \mathbf{X}\beta\right\|_2^2.
\end{aligned}$$

The marginal prior distribution for the parameter vector β stems from the Triple-Gamma-Prior constructed in Cadonna et al. (2020) given in Theorem 1 (a) and is given

by

$$p(\sqrt{\beta_j}|\phi^\xi, a^\xi, c^\xi) = \frac{\Gamma(c^\xi + \frac{1}{2})}{\sqrt{2\pi\phi^\xi} \cdot B(a^\xi, c^\xi)} \cdot U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j}{2\phi^\xi}\right)$$
$$\propto U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j}{2\phi^\xi}\right).$$

Here, $U(a, b, z)$ refers the confluent hyper-geometric function of the second kind which was introduced by Tricomi (1947). As this prior is specified for the parameter $\sqrt{\beta_j}$, we transform the prior by squaring the parameter to gain

$$p(\beta_j|\phi^\xi, a^\xi, c^\xi) \propto U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right).$$

Now, assuming that the parameters are independent a priori, the prior distribution is given by

$$\begin{aligned}
p(\beta) &= \prod_j^p p(\beta_j) \\
&= \prod_j^p p(\beta_j|\phi^\xi, a^\xi, c^\xi) \\
&\propto \prod_j^p U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right) \\
&= \prod_j^p \frac{1}{\Gamma(c^\xi + \frac{1}{2})} \int_0^\infty e^{-(\frac{\beta_j^2}{2\phi^\xi})t} t^{c^\xi + \frac{1}{2} - 1} (1+t)^{\frac{3}{2} - a^\xi - c^\xi + \frac{1}{2} - 1} dt \\
&\propto \prod_j^p \int_0^\infty \exp\left(-\frac{\beta_j^2}{2\phi^\xi} t\right) t^{c^\xi - \frac{1}{2}} (1+t)^{1 - a^\xi - c^\xi} dt. \qquad (4)
\end{aligned}$$

Here, in line 1 the assumption of independence between the parameters has been used to describe the distribution of the parameter vector as the product of its individual parameter distributions. In line 2, the marginal prior from Cadonna et al. (2020) has been used as the prior distribution for each individual parameter β_j. In line 3, scaling parameters have been removed by using the proportionality assumption. The last two lines of the derivation insert the integral representation of the confluent hyper-geometric function of the second kind, $U(a, b, z)$, which is valid in the case of a positive real part for the first parameter ($\mathfrak{Re}(a) > 0$) and again apply proportionality.

Taking the log of the prior distribution and using the properties of the logarithmic function yields the general result

$$\log(p(\beta)) = \log(\prod_j^p p(\beta_j|\phi^\xi, a^\xi, c^\xi)) = \sum_j^p \log(p(\beta_j|\phi^\xi, a^\xi, c^\xi)).$$

A common approach to estimation in regularization settings is the *maximum a posteriori probability (MAP)* estimator, which is defined as

$$\hat{\beta}_{MAP}(x) = \underset{\beta \in \mathbb{R}^p}{\arg\max} \left\{ f(x|\beta) g(\beta) \right\}.$$

where $f(x|\beta)$ describes the the probability density function of a variable x, which is parametrized by the parameter vector β. The second function $g(\beta)$ incorporates our prior information about the parameter vector β into the optimization problem.

Returning to our specific problem at hand, the posterior distribution of our parameter vector β can be retrieved by applying Bayes' theorem and the previously gained results in equations 4 and 3. Thus, the posterior distribution of the parameter vector β is proportional to

$$\begin{aligned} p(\beta|y, X, \sigma^2) &\propto p(y|X, \beta, \sigma) \times p(\beta) \\ &\propto \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}(y - X\beta)^T (y - X\beta)} \times \prod_j^p U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right). \end{aligned} \tag{5}$$

Making use of the monotonicity of the logarithmic function and seeing that it is easier to optimize the log-posterior, Taking the log of the posterior probability distribution, we take the log of result 5.

$$\begin{aligned} \log(\beta|X, y, \sigma^2) &= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}}\right) - \frac{1}{2\sigma^2} \left\|\mathbf{y} - \mathbf{X}\beta\right\|_2^2 + \log\left(\prod_j^p U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right)\right) \\ &= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}}\right) - \frac{1}{2\sigma^2} \left\|\mathbf{y} - \mathbf{X}\beta^2\right\|_2^2 + \sum_j^p \log\left(U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j}{2\phi^\xi}\right)\right) \\ &\propto -\frac{1}{2\sigma^2} \left\|\mathbf{y} - \mathbf{X}\beta\right\|_2^2 + \sum_j^p \log\left(U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right)\right) \end{aligned}$$

To align with the general specification structure of regularization problems, which can be seen from equation 1, a parameter λ will be multiplicatively added in front of the penalty term, which makes it possible to adjust the strength of the influence that the penalty has on the chosen parameters. By minimizing the negative log-posterior adjusted with λ, we can retrieve the *maximum a posteriori probability (MAP)* estimator using **Triple-Gamma-Regularization**:

$$\hat{\beta}_{MAP} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left(\frac{1}{2\sigma^2} \left\|\mathbf{y} - \mathbf{X}\beta\right\|_2^2 + \lambda \sum_j^p - \log\left(U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right)\right)\right) \tag{6}$$

5.1 Varying the Hyperparameters

After closer inspection of equation 6, it can easily be seen that this resembles the general penalized regression already seen in Hastie et al. (2009, p. 398) and in section 2 as $R(\beta) + \lambda \cdot J(\beta)$. The first term, also called the empirical loss in machine learning literature, is the widely known residual sum of squares:

$$R(\beta) = \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

The second part of the optimization problem can be viewed as a penalty imposed on the total risk based on the size of the estimates:

$$J_{TG}(\beta) = \sum_{j}^{p} - \log\left(U\left(c^\xi + \frac{1}{2}, \frac{3}{2} - a^\xi, \frac{\beta_j^2}{2\phi^\xi}\right)\right)$$

In contrast to the *LASSO* penalty, which uses the the absolute value of the coefficient, or the *Ridge* penalty, which uses the square of the estimate, this penalty derived from Cadonna et al. (2020) is based on the log of the confluent hyper-geometric of the second kind introduced by Tricomi (1947). Notably, this penalty term has three additional hyper-parameters: c^ξ, a^ξ and κ_B as $\phi^\xi = (2c^\xi)/(\kappa_B^2 a^\xi)$. Here, the restrictions $a^\xi > 0.5$ and $0 < c^\xi < \infty$ are necessary to ensure that the penalty for a β_j being equal to zero remains finite and not diverges to negative infinity at zero. This results, which has already been presented and proven as part of Theorem 2 in Cadonna et al. (2020, pp. 5–6), ensures that the negative log of the hypergeometric function remains finite and thus does not produce parameter estimates which are zero for every variables.

In contrast to penalties like *LASSO* and *Ridge* which have a pre-defined structure, the hyperparameters of *Triple-Gamma* penalty make it possible to adjust the structure of the penalty and thus its influence on the overall optimization problem. Of course, the other penalties have simpler mathematical structures, but are very limited in adapting to different settings. Thus, the *Triple-Gamma* penalty trades some of the mathematical simplicity to gain more flexibility by introducing these hyperparameters. The next few pages will elaborate on the different effects that different values of these hyperparameters have on the overall structure of the penalty.

Variations of the Hyperparameter a^ξ

The first hyper-parameter which can be adjusted is a^ξ. A plot with a set of different values for a^ξ can be found in figure 2. As already mentioned earlier, the necessary restriction for for this hyper-parameter is that it has to be strictly greater than 0.5 to guarantee the finiteness of the penalty. To demonstrate the effects of changes in a^ξ, the other parameters have been set to $c^\xi = 0.1$ and $\kappa_B = 2$. It can easily be seen from the figure that a^ξ steers the sharpness of the penalty in small neighbourhoods around $\beta = 0$. As a^ξ increases, the penalty because smoother at $\beta = 0$ with it eventually converging a *Gaussian Penalty* like behaviour. From a modelling perspective, this opens up the possibility of steering the degree of variable selection the penalty performs. Nonetheless, the overall structure of the penalty in the tails does not change systematically apart from a parallel shift, which can be readjusted by specifying a different weighting parameter λ or a different value for κ_B (more on effect of κ_B on the penalty can be found later in this chapter).

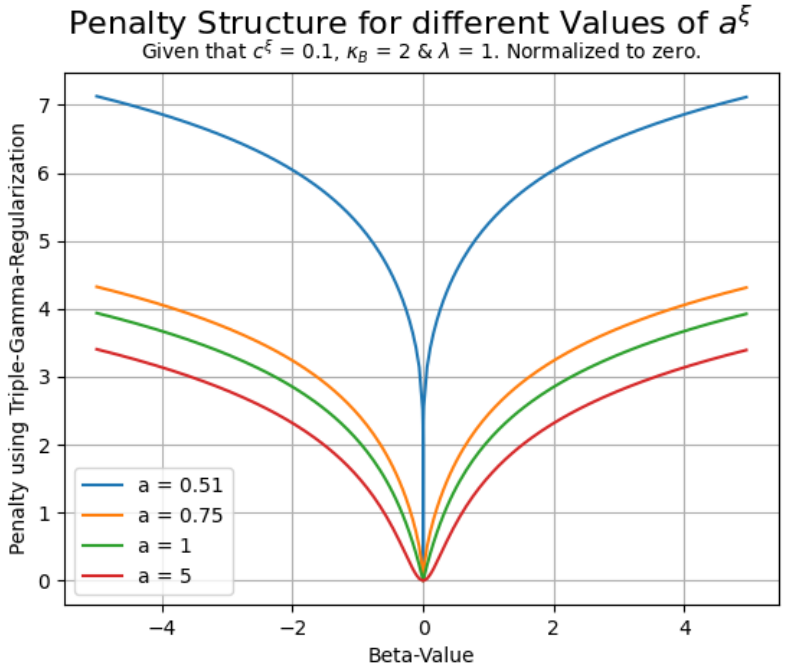


Figure 2: Triple-Gamma-Penalty using different values of a^ξ

Seeing this, it is apparent that a value of a^ξ close but strictly larger that 0.5 mimics the behaviour of the *Arctan Penalty* by Y. Wang and Zhu (2016) in small neighbourhoods of $\beta = 0$. Similar, large positive values for a^ξ lead to a *Gaussian Penalty* like behaviour in small neighbourhoods of $\beta = 0$ as recently proposed by John et al. (2022).

Variations of the Hyperparameter c^ξ

In contrast to the hyperparameter a^ξ, which mainly affects the behaviour at and around $\beta = 0$, changes in c^ξ mainly affect the behaviour in the tails. However, the effect that a change in c^ξ has on the penalty structure can be split up in two rough subsets of $(0, \infty)$. The effect of the first subset of values for c^ξ which are strictly greater than 0 but less or equal than 0.1 can be found in figure 3 (as already mentioned before, by definition, c^ξ has to be strictly greater than zero: $c^\xi > 0$). Here, it can be seen that as the values for c^ξ become smaller, a shifting effect takes place which generally does not influence the overall structure of the penalty, but increases the amount of penalty which is added to the risk function for β-values which are different from zero (In a sense, this has a similar effect to changes in κ_B, which will be explained later). Or, to put it differently, with values of c^ξ closer to zero, the data has be become even more convincing that the value is significantly different from zero.

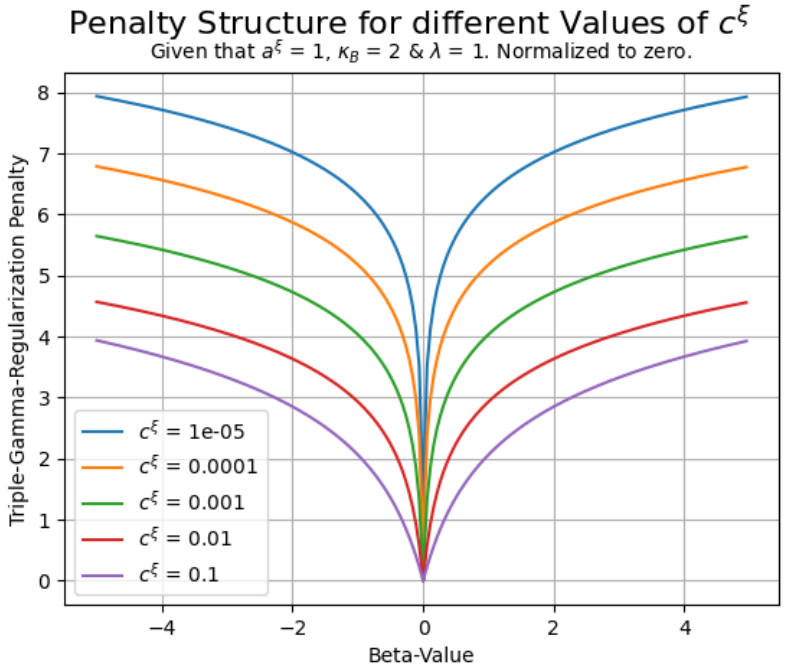


Figure 3: Triple-Gamma-Penalty using different values of c^ξ with $0 < c^\xi \leq 0.1$

However, the more interesting effect that a change in c^ξ has on the penalty structure can be seen for values of c^ξ that are greater than 0.1. In figure 4, a plot can be found with the Triple-Gamma-Penalty for larger values of c^ξ. Again, starting from the baseline with $c^\xi = 0.1$, higher values for this hyper-parameter mainly change the behaviour of the penalty in the tails. A result that has already been shown by Cadonna et al. (2020) in Table 1, where multiple different hyper-parameter settings are presented, is that with

an increasing value for c^ξ and with $a^\xi = 1$ as well as $\kappa_B = 2$, the Triple-Gamma-Prior converges to a *LASSO* like shrinkage behaviour. A property which carries over to the proposed regularization setting when using the proposed hyper-parameter values, thus showing that the Triple-Gamma-Penalty can be used both as a non-convex penalty as well as a *LASSO* penalty, creating increased flexibility in modelling approaches.

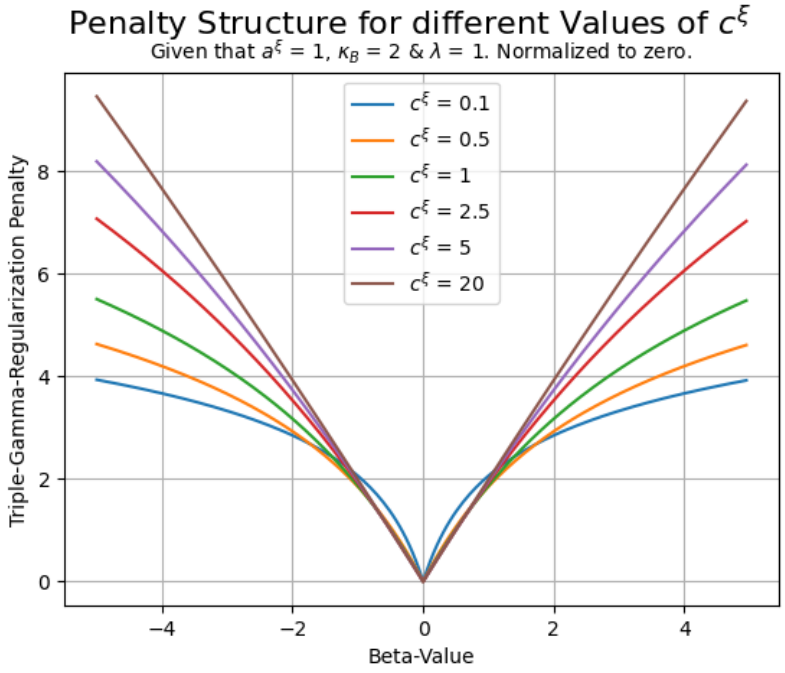


Figure 4: Triple-Gamma-Penalty using different values of c^ξ with $c^\xi \geq 0.1$

Variations of the Hyperparameter κ_B

The third and final hyper-parameter κ_B enters the Triple-Gamma-Penalty $J_{TG}(\beta)$ as part of $\phi^\xi = \frac{2c^\xi}{\kappa_B^2 a^\xi}$ as can be seen from equation 6. Notably, κ_B is squared and thus only the absolute value of κ_B, $|\kappa_B|$, influences the structure of the penalty. The overall third function value is defined as $\frac{\beta_j^2}{2\phi^\xi}$ and by plugging in ϕ^ξ we get $\frac{\beta_j^2 \kappa_B^2 a^\xi}{4c^\xi}$, it can be seen that a value of $\kappa_B = 0$ leads to the entire parameter value being zero for all values of β_j. Hence, a change in β_j won't influence the penalty and furthermore won't have an influence on the overall risk minimization problem, the result being that the optimal set of parameters will only depend on the chosen loss function.

For all values of $\kappa_B \neq 0$, the value of the hyper-parameter will influence the penalty structure. A plot with several different values for the absolute value of κ_B can be found in figure 5. It can be seen that κ_B has a scaling effect on the strength of the penalty, thus playing a similar role as the generally multiplicatively added regularization parameter λ. For small values of κ_B, the penalty only has a weak singularity in small neighbourhoods

around 0, thus making it more likely for the parameters to be different from zero. As κ_B increases, the spike becomes more pronounced and thus increases the added penalty for parameters different from zero.

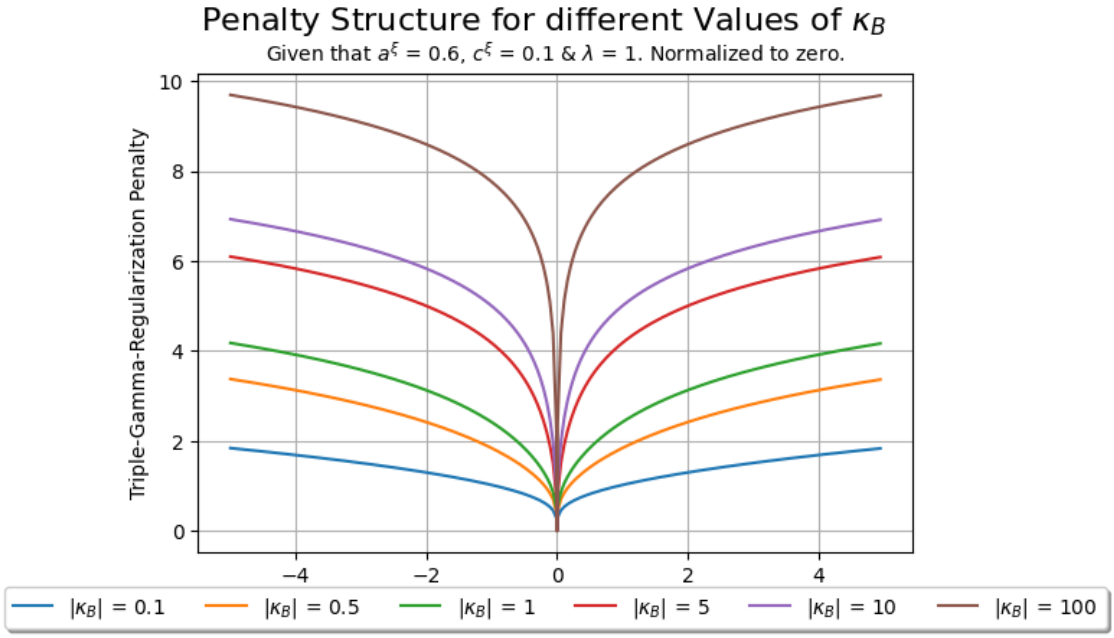


Figure 5: Triple-Gamma-Penalty using different values of κ_B

Although it has a scaling effect like λ, it still has this structural effect as well. Allowing myself to use a Bayesian analogy in this setting, it can be viewed as follows: Increasing κ_B corresponds to a way of putting more prior knowledge on zero and thus pulling coefficients stronger towards zero.

A summary of all these effects can be found in table 2 below where the effects of positive and negative changes in the hyperparameters as well as defined ranges for these parameters are defined.

	Change		Mathematical Properties	
Variable	**Positive** Change	**Negative** Change	Defined Range	Misc.
a^ξ	Shifting towards *Gaussian*-like behaviour at $\beta = 0$	Shifting towards singularity at $\beta = 0$; Higher immediate penalty for coefficients $\beta \neq 0$	$(0.5, \infty)$	-
c^ξ	Generally, convergence towards convexity and, given certain settings for a^ξ and κ_B, LASSO. Higher values increase the additional penalty for higher absolute values of β.	For values smaller than 0.1, similar effect to increase in a^ξ	$(0, \infty)$	-
κ_B	The singularity at $\beta = 0$ becomes more pronounced, putting more *prior knowledge* on β being zero.	The reverse effect to the positive effect.	$(-\infty, \infty)$	When $\kappa_B = 0$, the value of β doesn't affect the overall penalty anymore, thus reverting the optimization problem back to the non-penalized one.

Table 2: Summary of the effects of changes in the hyperparameters a^ξ, c^ξ and κ_B on the penalty structure

5.2 Comparison to already existing Penalty Terms

As already mentioned in section 3, several other penalty terms have already been widely studied in the literature. Convex penalties like *Ridge* (A. E. Hoerl & Kennard, 1970b) or non-convex penalties like the *Ar(c)tan* (Y. Wang & Zhu, 2016) and *Gaussian* (John et al., 2022) have managed to establish itself as prominent approaches to regularization. It is now certainly of interest to see how the *Triple-Gamma* penalty compares to the established methods. Seeing that, due to its flexibility, there is not *one* *Triple-Gamma* penalty, three distinct hyper-parameter setting have been chosen to represent the proposed penalty term.

	Hyperparameter		
	a^ξ	c^ξ	κ_B
Setting 1	0.75	0.1	2
Setting 2	5	0.01	2
Setting 3	0.51	0.001	1

Table 3: Three Example Settings of the *Triple-Gamma* Penalty

A visualisation of these settings can be found in figure 6. It can be seen that all settings differ heavily in its structure and thus its influence on the optimization problem and in the end the resulting parameters.

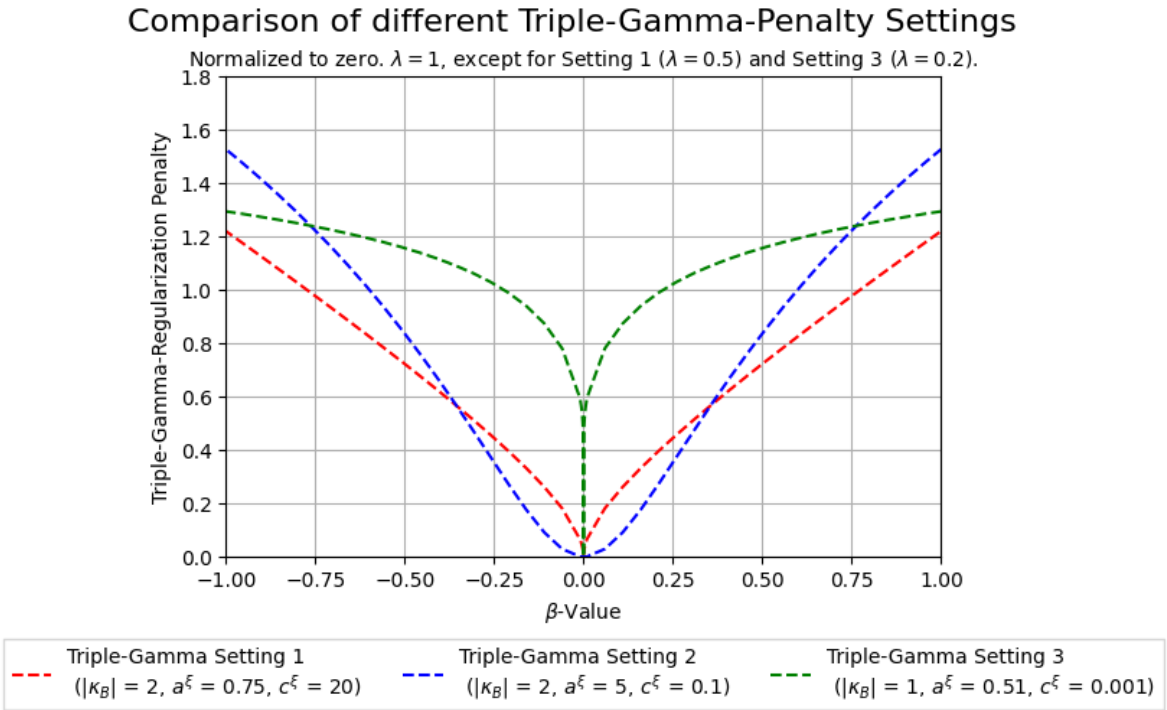


Figure 6: Different Settings of the Triple-Gamma-Penalty

Setting 1 is a *LASSO* like regularization penalty, but still keeps the non-convex shape of the Triple-Gamma penalty. *Setting 2* introduces the smooth-at-origin property but keeps a similar shape than *setting 1* in the tails. *Setting 3* has the most distinct structure with a strong singularity at origin but tails which fall of very quickly and thus adds relatively little penalty for parameters that are above 1 in absolute terms.

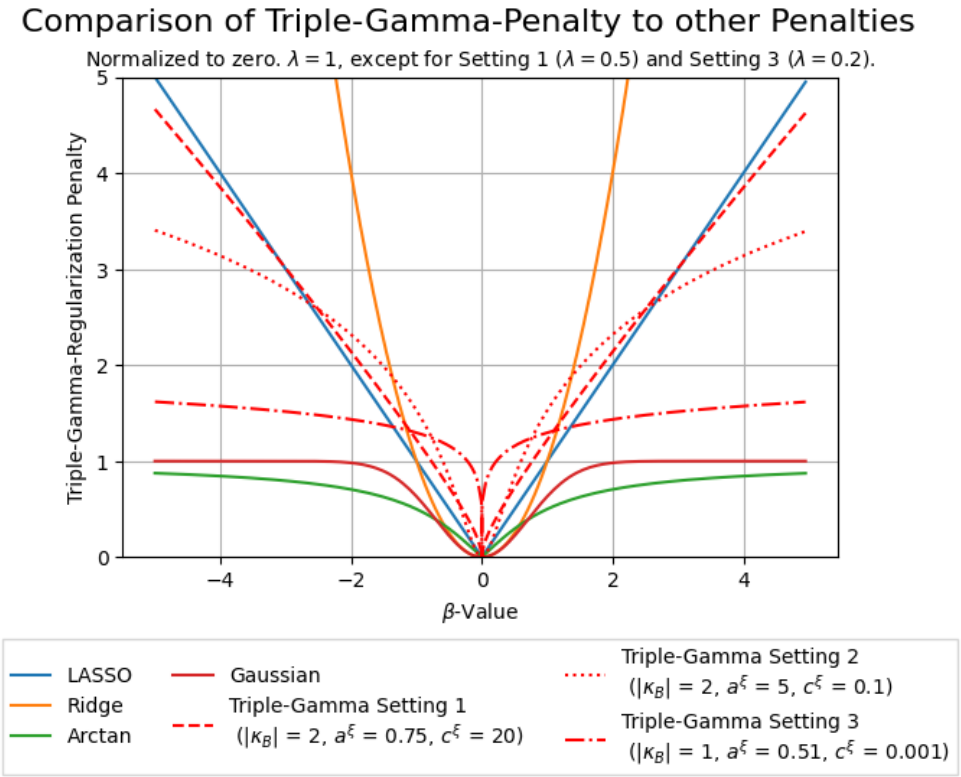


Figure 7: Basic Representation of the Triple-Gamma-Penalty using Hyperparameters $c^\xi = 0.1, \kappa_B = 2, a^\xi = 0.75$ compared to *LASSO* and *Ridge* Penalties

Additionally, a comparison of these *Triple-Gamma* settings to a selection of other regularization penalties can be found in figure 7. As can be seen here, *setting 1* of the *Triple-Gamma* penalty imitates a *LASSO* penalty with a slightly decreasing additional penalty, resulting in a weakened bias which is common to *LASSO* estimates. *Setting 3* has a similar structure than the non-convex penalties *Gaussian* and *Arctan*, but with a string singularity at zero. Here, it can clearly be seen that the *Triple-Gamma* penalty is not a penalty which converges in the tails, as do *Gaussian* and *Arctan*, but still has a considerably weaker bias as convex methods like *LASSO* and *Ridge*. Overall, figure 7 showcases the variety of possible penalties that can be implemented using the *Triple-Gamma* penalty.

6 Simulation Study

To test the proposed method of section 5 a simulation study will now be conducted whether, and if so, in what scenarios, the *Triple-Gamma* penalty might be valuable addition to the literature. Of course several previously introduced methods have also been tested using simulation studies, but the methodology behind these studies usually varies from paper to paper. However, before moving on to this section of the thesis, the question remains on how to solve the optimization problem in 6.

6.1 Approaches to Estimation

As already mentioned in chapter 3, non-convex penalties like the *Triple-Gamma* penalty are usually harder to solve than convex optimization problems. The reason being that convex functions have a unique global minimum, whereas non-convex functions can have multiple local minima, which are not necessarily a global minimum. **(Missing Citation)**In addition, unlike for example *Ridge* regression, the *Triple-Gamma* penalty like many other established penalties in the literature does not have a closed form solution for this optimization problem. Thus, it is necessary to apply an optimization algorithm which solves this problem iteratively. An algorithm which has established itself to solve these kinds of problems is called *Gradient Descent* (Grippo & Sciandrone, 2023, pp. 229–248). In its essence, the algorithm tries to minimize a loss function by iteratively adjusting the model parameters in the direction that reduces the loss most. It calculates the gradient of the loss function with respect to the model parameters and then updates the parameters by a learning rate. This iterative process can either be repeated until a specified number of iterations is reached or a local (or global) minimum has been found. As already mentioned, this can lead to problems when dealing with non-convex optimization problems as multiple local minima might exist, which do not necessarily need to be global ones. Various approaches to deal with this have been proposed. One method uses random starting points to cover various areas of the optimization space to cover potential multiple local minima. Another method is called *Stochastic Gradient Descent (SGD)* and uses different batches of the dataset to estimate the model parameters (Amari, 1993). The *Gradient Descent* method will be used for the simulation study in the following chapter.

For the sake of completeness and as a stepping stone for further research, it is necessary to mention a different approach to solve the optimization problem which is called *Local Linear Approximation (LLA)* algorithm and has been proposed by Zou and Li (2008). Their resulting *one-step LLA estimates* have several computational advantages and "alleviate the computation burden in the iterative algorithm and overcome the potential local maxima problem in maximizing the nonconcave penalized likelihood" (Zou & Li, 2008). Still, to the best of my knowledge, no easily accessible programming packages

exist for this algorithm.

6.2 Scenario Simulation

Various simulation studies comparing different regularization penalties have been published so far. Vettam and John (2022) for example have compared *LASSO, Ridge, MCP, SCAD, Laplace* and *Arctan* in the context of the estimation of neural networks. F. Wang et al. (2020) have tested a range of penalties in the context of linear models and analysed various the performance of penalties using levels of sparsity, dimensionality and data availability. The authors tested the performance based on three criteria: *prediction*, *variables selection* and *variables ranking*. This simulation study will focus on the the first of these metrics.

Prediction is usually measured by splitting a simulated dataset into a training dataset and a test dataset. The model is fitted on the training dataset and then tested on the test dataset using a metric like the *mean squared error (MSE). Variable Selection is measured in various ways. F. Wang et al. (2020) used true-positive and false-positive rates as their metric of choice. As several of the tested methods in this simulation study do not perform variable selection naturally due to its smooth-at-origin property and thus report estimates larger than zero, a different metric will be chosen for this simulation study. To test whether models accurately predict the underlying data generating process and also result in estimates close to its true coefficients, the square of the absolute deviation between the true coefficient and the estimated coefficient of each variable will be computed and summed up. Using this metric, models whose estimates are close the true coefficients will perform than models whose estimates are far off.*

Some of the previous paragraph is in italic, because I initially planned on running a second simulation study on this metric as well, but implementation and runtime issues made it difficult. Will probably just mention it.

To evaluate the models in a wide range of contexts, every simulation scenario will vary in terms off three variables: *sample size* N (ie $N = 100$ meaning that for every run, a total of 100 observations will be generated), *dimensionality* p (ie $p = 10$ meaning a total of ten covariates will included in the model) and *sparsity* s (ie $s = 0.1$ meaning that 10% of the covariates have a true coefficients different from zero). The set of values chosen for each scenario variable can be found in table 4. Using this values, all possible combinations of these variable settings will be formed, resulting in a total of 36 possible scenarios.

Variable	Values
Sample Size N	$50, 100, 500, 1000$
Dimensionality p	$3, 10, 25$
Sparsity s	$0.1, 0.5, 0.9$

Table 4: Scenarios Values

The remaining issue to cover deals with the weighting/regularization parameter λ (see section 2) and how it is set. This key parameter controlling the strength of the influence of the penalty on the overall optimization result is usually set by the user, however another *hands-off* method is cross-validation, which will be used for this simulation (Hastie et al., 2009, pp. 250–251). A grid of possible values for λ is chosen, the model is being estimated for every possible value, and the best model will be chosen based on the chosen cross-validation metric. For this simulation study, all values between 0 and 1 based on 0.01 steps will be tested on the first generated dataset. To save time when it comes to computation, the search will be stopped if the chosen metric hasn't improved for more than 10 new values for λ. After his run, these optimal values will be used in the overall simulation. **For the application of the gradient descent algorithm a maximum number of 750 epochs has been set.** The simulations have been run with Python (Version 3.11.4) along with scikit-learn (Version 1.4.1.post1) and PyTorch (Version 2.2.2).[3]

6.3 Simulation Study for Prediction

For the simulation study testing the models prediction accuracy the key metric to test performance will be the *mean-squared-error (MSE)*. The data is being generated the following way: Given a set of scenario parameters N_i, p_i and s_i, the first $s_i * p_i$ coefficients, rounded to the next larger integer, will be generated from a normal distribution with mean 0 and variance 1, the remaining coefficients will be set to zero. Then, N_i draws will be generated from a normal distribution with mean 0 and variance 1 for each of the p_i features. Using these coefficients and the data from the features a response variable y will be computed and noise drawn from a normal distribution with mean 0 and variance 1 multiplied with 0.5 will be added as noise. First, the grid search for the optimal λ will be performed by generating a sample dataset using the above generating process. Then, every method that is part of the simulation will be estimated on the dataset for every λ_i in the set $\lambda_i \in \{0, 0.01, 0.02, \cdots, 1\}$. For every method separately, the λ_i which generates the lowest validation loss on the testset will be used for the overall simulation.

[3]More information on the packages used and its installed version can be found in the accompanying GitHub repository.

Using the set of scenario variables mentioned in table 4, all possible combinations will be formed resulting in the overall 36 scenarios. The code iterates through the list of scenarios, generates 25 datasets based on these scenarios variables and all tested methods will be estimated on every one the test part of these datasets. Afterwards, the validation loss will be computed for each method and stored. This procedure results in tables 5, which shows the mean and standard deviation of the distribution of the validation loss for every scenario and every methods. The values colored in red indicate the smallest means for every scenario. Before advancing to the overall results of this simulation study, it might be useful to have a look at a specific scenario to better understand the results in table 5, which will be the scenario with $N = 100, p = 25, s = 0.1$.

In this scenario, we are dealing with a dataset of 100 observations, 25 features and 3 non-zero coefficients.[4] A figure with boxplots showcasing the distributions of the Mean-Squared-Error rates on the validation set for all methods can be found in figure 8

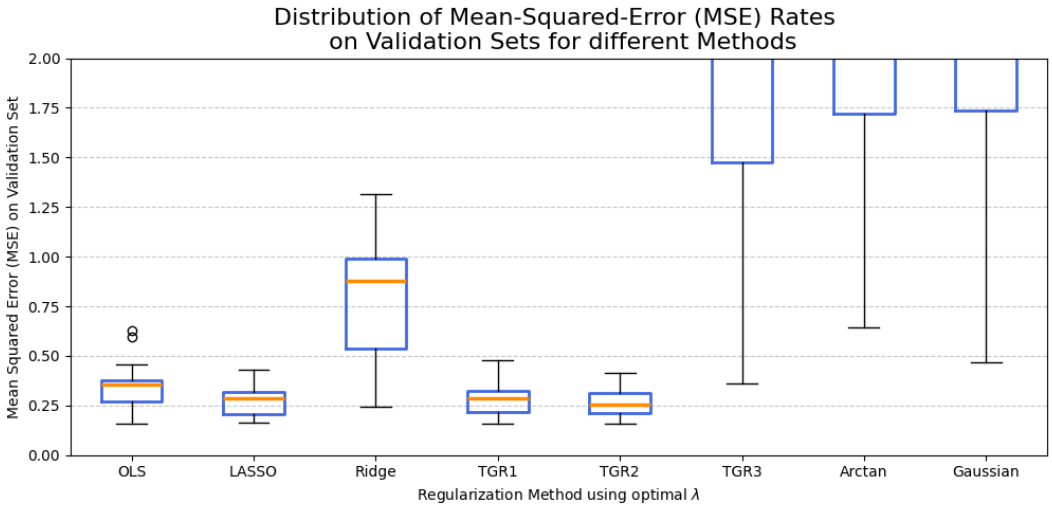


Figure 8: Distribution of MSE on Validation Set in the Scenario with 100 Observations, 25 Features and 3 non-zero Coefficients

This scenario, which is a good exmaple for a sparse scenario, where a lot of features are in the dataset but only a small part of it is different from zero, i.e. has an effect on the outcome. Here, *LASSO* performs well which was to be expected, but it can also be seen that the first and second setting of the *Triple-Gamma* Penalty can both produce similar results when it comes to predictive power. *Setting 2* even has a lower mean in MSE (0.25255) than *LASSO* regression (0.2858). *OLS* manages to perform quite well, yet has a visible shift towards higher *MSE* values. *Arctan, Gaussian* and the third setting

[4] $25 \times 0.1 = 2.5$ and $ceil(2.5) = 3$.

of the *Triple-Gamma* penalty perform significantly worse in this scenario.

Generally, it can be seen table 4 that the larger the sample size gets, the better OLS performs on the dataset as enough data is present for the OLS regression to work properly. However, in sparse data settings as well as when the sample size is low, regularization approaches regularly outperform the OLS method, which aligns with theory. Essentially, *LASSO* and the *first* and *second* setting of the *Triple-Gamma* penalty outperform most other methods in these scenarios. Especially, the two models estimated with these two settings of the *Triple-Gamma* approach manage to work best in these scenarios in most cases, highlighting the potential benefit that the *Triple-Gamma* regularization might have in these scenarios.

I will extend my discussion on the results and go a bit more into detail. But the general idea of the results should be there.

6.4 Computational Efficiency

At this point of the thesis it is necessary to quickly cover the topic of computational efficiency. As mentioned earlier, this simulation study has been implemented using Python along with PyTorch. As the *Triple-Gamma* penalty is not a simple algebraic computation, as for example computing the absolute value of a real number in the case of *LASSO* or computing the arcus tangens in the case of the *Arctan* penalty, it is apparent that the computation of the loss function given a set of parameters is not very straight-forward and computationally expensive. In its current implementation, the Python function computing the *Triple-Gamma-Loss* has to access a second function which approximates the logarithm of the confluent hyper-geometric function of the second kind. More importantly, it does so for every parameter in the model. This is, to no surprise, very inefficient and makes computations in high dimensions unnecessarily complicated. In figure 9 a plot can be found which shows the time to estimate all three settings of the *Triple-Gamma-Regularization* on a dataset generated with parameters $N = 100$ and $s = 0.1$ with an increasing number of features p.

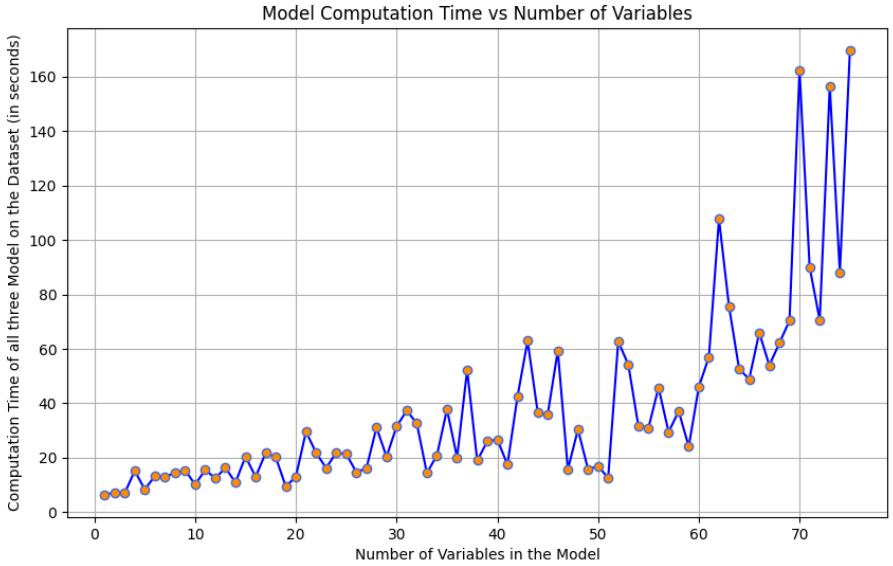


Figure 9: Time necessary to compute all three scenarios of the Triple-Gamma-Regularization with $N = 100$ and $s = 0.1$ dependent on the number of Features p

As can be seen from figure 9 above, the time to estimates these models increases drastically with an increased number of variables. Setting aside that it might even grow exponentially, it can be seen that the computation time also increasingly becomes more unstable. Several approaches might help in resolving these issues, like using *Stochastic Gradient Descend (SDG)*, which splits up the training dataset stochastically into smaller

batches, **(Missing Citation)**or choosing a different estimation algorithm altogether (see section 6.1). Another approach could be to implement the underlying estimation in a more efficient programming language when it comes to runtime like C++. All in all, this definitely opens up a field for further research to make the overall method more suitable for application.

Table 5 – continued from previous page

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	TGR Setting 1	TGR Setting 2	TGR Setting 3	Arctan	Gaussian
50	3	1	0,25234	0,2219	0,27945	0,22478	0,21905	0,35557	0,97859	0,93073
			(0,16398)	(0,16634)	(0,30662)	(0,15457)	(0,16378)	(0,9856)	(1,75094)	(1,07439)
50	3	2	0,22655	0,23138	0,39515	0,23436	0,21497	0,47195	1,3962	1,18809
			(0,11512)	(0,11184)	(0,51324)	(0,10803)	(0,11079)	(0,78161)	(2,19764)	(3,67118)
50	3	3	0,20423	0,21708	0,53198	0,19216	0,20721	0,90621	1,92277	2,41364
			(0,13742)	(0,15681)	(0,60466)	(0,14344)	(0,15232)	(1,94347)	(5,64658)	(5,04711)
50	10	1	0,28023	0,22356	0,29245	0,21951	0,2116	0,42564	0,80753	0,82095
			(0,17155)	(0,13548)	(0,54917)	(0,14297)	(0,12048)	(1,06699)	(1,6802)	(2,02455)
50	10	5	0,28965	0,28843	0,93375	0,26844	0,28782	2,89328	4,5195	4,47004
			(0,1554)	(0,15598)	(0,7604)	(0,11256)	(0,13445)	(2,27582)	(5,73895)	(4,20082)
50	10	9	0,30136	0,37676	2,00818	0,33526	0,35175	5,72024	10,64451	10,72909
			(0,1306)	(0,14906)	(1,17919)	(0,1349)	(0,16711)	(3,36261)	(5,36156)	(5,46578)
50	25	3	0,51829	0,36501	0,62776	0,3955	0,30886	1,46023	1,77643	1,88994
			(0,36764)	(0,21862)	(0,65178)	(0,29328)	(0,22148)	(1,78356)	(2,71921)	(2,21)
50	25	13	0,67486	0,54569	2,9942	0,91695	0,80402	9,23039	11,46166	12,24814
			(0,35518)	(0,36132)	(2,02239)	(0,77358)	(0,50006)	(5,06033)	(6,86366)	(6,86801)
50	25	23	0,77777	1,31854	4,98093	2,14074	2,11424	14,84778	21,84308	21,33126
			(0,47943)	(0,74119)	(3,66703)	(1,62387)	(1,4291)	(9,21278)	(11,33141)	(12,44137)
100	3	1	0,26113	0,24148	0,28637	0,24006	0,23919	0,33325	0,80169	0,62443
			(0,06292)	(0,05724)	(0,1803)	(0,06026)	(0,06046)	(0,35087)	(1,85348)	(1,84042)

Continued on next page

Table 5: Mean and Standard Deviations of MSE on the Validation Set for different Scenarios (25 runs per scenario)

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	TGR Setting 1	TGR Setting 2	TGR Setting 3	Arctan	Gaussian
100	3	2	0,23982	0,23845	0,41608	0,2366	0,24922	0,53486	1,56864	1,62051
			(0,06316)	(0,06489)	(0,15747)	(0,06092)	(0,06462)	(0,6425)	(1,05884)	(1,20262)
100	3	3	0,29648	0,27556	0,44406	0,27967	0,29763	0,81765	2,19614	1,87073
			(0,09054)	(0,08227)	(0,48781)	(0,08737)	(0,08619)	(1,02153)	(3,74058)	(3,38036)
100	10	1	0,2734	0,26898	0,40486	0,26611	0,26677	0,55277	1,05805	1,13716
			(0,07944)	(0,08039)	(0,13091)	(0,07539)	(0,07919)	(0,78647)	(0,81199)	(0,95467)
100	10	5	0,26237	0,251	1,06942	0,243	0,25305	2,7111	5,93076	5,18044
			(0,09054)	(0,09586)	(0,5515)	(0,08831)	(0,0991)	(1,97029)	(3,65107)	(3,10735)
100	10	9	0,25386	0,32494	1,72644	0,2797	0,30738	5,6015	9,87634	9,13047
			(0,10201)	(0,12919)	(0,9834)	(0,1147)	(0,11993)	(4,00655)	(5,05172)	(5,53228)
100	25	3	0,35328	0,2858	0,87797	0,28449	0,25255	2,90378	3,90851	3,94007
			(0,113)	(0,07274)	(0,56232)	(0,07773)	(0,0729)	(1,76077)	(2,74017)	(2,62225)
100	25	13	0,39073	0,43176	2,56021	0,5217	0,35142	8,24937	11,40495	10,7303
			(0,1303)	(0,12362)	(1,25941)	(0,31652)	(0,1927)	(4,35583)	(5,75347)	(5,60251)
100	25	23	0,34696	0,55041	4,83633	1,02786	0,97566	20,17376	25,35184	24,57685
			(0,1123)	(0,18314)	(2,02043)	(0,97606)	(0,9903)	(7,27886)	(10,40436)	(10,40762)
500	3	1	0,25492	0,25986	0,34586	0,25577	0,2552	0,39464	0,9333	1,14691
			(0,03172)	(0,03346)	(0,39479)	(0,03298)	(0,0348)	(0,80202)	(2,63126)	(2,72861)
500	3	2	0,26374	0,26873	0,54005	0,26467	0,2691	0,75057	2,1697	2,26932
			(0,0221)	(0,02096)	(0,23233)	(0,02099)	(0,02182)	(1,08099)	(1,72071)	(2,01686)

Continued on next page

Table 5: Mean and Standard Deviations of MSE on the Validation Set for different Scenarios (25 runs per scenario)

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	TGR Setting 1	TGR Setting 2	TGR Setting 3	Arctan	Gaussian
500	3	3	0,26158	0,27067	0,61755	0,26586	0,26434	1,05504	2,96089	3,16231
			(0,04296)	(0,04119)	(0,27379)	(0,04229)	(0,03968)	(0,79724)	(2,38658)	(2,07472)
500	10	1	0,24745	0,25079	0,33238	0,244	0,2541	0,64	1,06196	1,20499
			(0,04425)	(0,03818)	(0,1129)	(0,03922)	(0,03966)	(0,43651)	(0,93851)	(1,01908)
500	10	5	0,24921	0,26579	0,81008	0,25288	0,26072	2,6974	3,61381	3,85227
			(0,04205)	(0,04301)	(0,57001)	(0,04222)	(0,03949)	(2,56921)	(4,61234)	(4,56494)
500	10	9	0,24397	0,28519	1,44095	0,25363	0,28111	5,84618	8,56159	9,97515
			(0,03147)	(0,043)	(0,7961)	(0,07933)	(0,10408)	(3,07682)	(6,16018)	(5,99551)
500	25	3	0,26085	0,25511	0,57272	0,2472	0,24907	2,09905	2,7201	2,72876
			(0,03684)	(0,03815)	(0,29767)	(0,03757)	(0,04021)	(1,77082)	(2,04823)	(2,27602)
500	25	13	0,27144	0,32268	1,86148	0,29349	0,30065	9,38381	12,09481	12,38708
			(0,02889)	(0,04375)	(0,69901)	(0,11735)	(0,05267)	(3,4034)	(4,82712)	(4,64426)
500	25	23	0,2755	0,39447	3,1937	0,41115	0,45642	16,75672	21,00113	20,08944
			(0,03552)	(0,06411)	(1,276)	(1,11575)	(1,0544)	(6,37604)	(9,35346)	(9,5086)
1000	3	1	0,25053	0,25757	0,34022	0,25349	0,25861	0,41004	1,07548	1,1043
			(0,02649)	(0,0281)	(0,20817)	(0,02671)	(0,02731)	(0,69077)	(1,89567)	(1,5998)
1000	3	2	0,25114	0,25515	0,46823	0,24898	0,24944	1,07539	2,16733	1,7779
			(0,02077)	(0,02314)	(0,17607)	(0,02115)	(0,02262)	(0,93913)	(1,47574)	(1,61607)
1000	3	3	0,25034	0,25653	0,53652	0,25239	0,24992	0,78222	3,06023	2,33984
			(0,0289)	(0,031)	(0,28536)	(0,02764)	(0,02859)	(0,9846)	(1,5519)	(2,22089)

Continued on next page

Table 5: Mean and Standard Deviations of MSE on the Validation Set for different Scenarios (25 runs per scenario)

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	TGR Setting 1	TGR Setting 2	TGR Setting 3	Arctan	Gaussian
1000	10	1	0,25044	0,25084	0,33524	0,25095	0,24915	0,7077	1,27558	1,51029
			(0,01971)	(0,02246)	(0,27203)	(0,02019)	(0,02237)	(1,02835)	(1,87087)	(1,54743)
1000	10	5	0,24012	0,26252	0,68128	0,24393	0,25507	2,60964	4,16242	4,29199
			(0,02191)	(0,02371)	(0,39154)	(0,02244)	(0,02319)	(1,90423)	(3,10798)	(2,78838)
1000	10	9	0,2563	0,28626	1,32663	0,25664	0,26883	4,90589	8,38851	8,28297
			(0,02596)	(0,02975)	(0,38829)	(0,02786)	(0,03265)	(1,77821)	(3,19513)	(3,29618)
1000	25	3	0,25279	0,25831	0,78625	0,26149	0,25134	3,05865	3,65552	3,84833
			(0,02674)	(0,02874)	(0,34754)	(0,03574)	(0,02827)	(1,72991)	(2,50772)	(2,32238)
1000	25	13	0,2587	0,31177	2,03451	0,30109	0,28809	9,19749	12,10087	12,51176
			(0,02206)	(0,03114)	(0,70594)	(0,07099)	(0,02876)	(3,68473)	(5,07376)	(5,42773)
1000	25	23	0,26145	0,35083	3,77492	0,41261	0,4022	19,54978	26,04359	25,08614
			(0,02966)	(0,03516)	(1,0891)	(0,36625)	(0,33202)	(5,65596)	(6,87056)	(7,12848)

6.5 Implementation as Python Package

As mentioned before, this simulation study has been run using Python (Version 3.11.4) along with scikit-learn (Version 1.4.1.post1) and PyTorch (Version 2.2.2). The accompanying code, a list of packages used and its version and other material can be found on GitHub. The script *tgr.py* can be used as a script to apply *Triple-Gamma-Regularization* on a dataset of choice. To make its application easier, the following subsections provides important information on how to use it.

General Functions

Function Name	Description
`TripleGamma-Regularization`	Here a class is defining a simple neural network. It uses the `nn.Module` and initializes a linear model with 10 input features and 1 output feature without an intercept.
`TripleGammaReg_Loss`	This function computes the loss for the Triple Gamma Regularization. It takes the predicted values (`Y_HAT`), the true values (`Y`), the coefficients of the model (`coefficients`), the regularization parameter (`lamda`), and the regularization parameters `a`, `c` and `kappa` as input. The loss is computed according to the theoretical formulation by using the script `log_hyperu.py`, which approximates the log of the confluent hyper-geometric function of the second kind.
`TripleGammaModel`	This function trains the *Triple-Gamma* Regularization model. It takes the feature matrix (`X`), the target values (`y`), the regularization parameter (`penalty`), regularization hyper-parameters `a`, `c`, `kappa`, the number of epochs (`num_epochs`), and the learning rate (`lr`) as input. It initializes the `PyTorch` model and optimizer, uses Gradient Descent to train the model for the specified number of epochs, and returns the trained model, a list of model coefficients for each epoch, and a list of loss values for each epoch.

Dependencies

To successfully run the script, the following dependencies are required:

- `torch`: PyTorch library including sub modules `torch.nn` and `torch.optim` for building and training neural networks with at least version 2.2.2
- `log_hyperu as hyperu`: The custom module for computing the logarithm of the hypergeometric function, which can be found on GitHub.

Data Types of Inputs and Outputs of the Functions

- `TripleGammaModel`
  - **Input:** `X` (Tensor), `y` (Tensor), `penalty` (float), `a` (float), `c` (float), `kappa` (float), `num_epochs` (int, default=1000), `lr` (float, default=0.01)
  - **Output:** `model` (Initialization of the class TripleGammaRegularization), `coef_list` (list of Tensors), `loss_list` (list of Tensors)

7 Possible Extensions and Criticism

I will write the last two paragraphs as soon as the rest of the thesis is 100% air tight to avoid having to rewrite the conclusions multiple times. It should take long though.

8 Conclusion

ENDE

9 List of Figures

List of Figures

10 List of Tables

List of Tables

11 References

References

Amari, S.-i. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, *5*(4-5), 185–196.

Bitto, A., & Frühwirth-Schnatter, S. (2019). Achieving shrinkage in a time-varying parameter model framework. *Journal of Econometrics*, *210*(1), 75–97.

Cadonna, A., Frühwirth-Schnatter, S., & Knaus, P. (2020). Triple the gamma—a unifying shrinkage prior for variance and variable selection in sparse state space and tvp models. *Econometrics*, *8*(2), 20.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Fan, J., & Li, R. (2006). Statistical challenges with high dimensionality: Feature selection in knowledge discovery. *arXiv preprint math/0602133*.

Frank, L. E., & Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, *35*(2), 109–135.

Fu, W. J. (1998). Penalized regressions: The bridge versus the lasso. *Journal of computational and graphical statistics*, *7*(3), 397–416.

Grippo, L., & Sciandrone, M. (2023). *Introduction to methods for nonlinear optimization* (Vol. 152). Springer Nature.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

Hoerl, A. E., & Kennard, R. W. (1970a). Ridge regression: Applications to nonorthogonal problems. *Technometrics*, *12*(1), 69–82.

Hoerl, A. E., & Kennard, R. W. (1970b). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hoerl, R. W. (2020). Ridge regression: A historical context. *Technometrics*, *62*(4), 420–425.

John, M., Vettam, S., & Wu, Y. (2022). A novel nonconvex, smooth-at-origin penalty for statistical learning. *arXiv preprint arXiv:2204.03123*.

Kukačka, J., Golkov, V., & Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*.

Lazar, N. (2010). Ockham's razor. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(2), 243–246.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288.

Tricomi, F. (1947). Sulle funzioni ipergeometriche confluenti [Paper for hypergeometric function of second kind.]. *Annali di Matematica Pura ed Applicata*, *26*(1), 141–175. https://doi.org/10.1007/BF02415375

Trzasko, J., & Manduca, A. (2009). Relaxed conditions for sparse signal recovery with general concave priors. *IEEE Transactions on Signal Processing*, *57*(11), 4347–4354.

van Wieringen, W. N. (2015). Lecture notes on ridge regression. *arXiv preprint arXiv:1509.09169*.

Vapnik, V. (1991). Principles of risk minimization for learning theory [Definition of Empirical Risk Minimization]. *Advances in neural information processing systems*, *4*.

Vettam, S., & John, M. (2022). On two recent nonconvex penalties for regularization in machine learning. *Results in Applied Mathematics*, *14*, 100256.

Wang, F., Mukherjee, S., Richardson, S., & Hill, S. M. (2020). High-dimensional regression in practice: An empirical study of finite-sample prediction, variable selection and ranking [Paper with simulation study]. *Statistics and computing*, *30*, 697–719.

Wang, Y., & Zhu, L. (2016). Variable selection and parameter estimation with the atan regularization method. *Journal of Probability and Statistics*, *2016*.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *67*(2), 301–320.

Zou, H., & Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, *36*(4), 1509.