

Master's Thesis

Title of Master's Thesis:	Triple-Gamma-Regularization: A Flexible Non-Convex Regularization Penalty based on the Triple-Gamma-Prior
Author (last name, first name):	Unterwiesingh Lukas Paul, BSc (WU)
Student ID number:	11913169
Degree program:	Master in Economics
Examiner (degree, first name, last name):	Peter Knaus, PhD

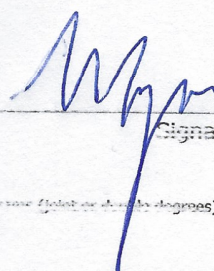
I hereby declare that:

1. I have written this master's thesis myself, independently and without the aid of unfair or unauthorized resources. Whenever content has been taken directly or indirectly from other sources, this has been indicated and the source referenced. I am familiar with the regulations specified in the Directive on Plagiarism and Other Types of Academic Fraud in Academic Theses.
2. This master's thesis has not been previously presented as an examination paper in this or any other form in Austria or abroad*.
3. This master's thesis is identical with the thesis assessed by the examiner.
4. (only applicable if the thesis was written by more than one author): this master's thesis was written together with

The individual contributions of each writer as well as the co-written passages have been indicated.

24/09/2024

Date



Signature



Student-ID: 11913169
Degree Program: Master of Science in Economics (Science Track)
Examiner: Peter Knaus, PhD
Vienna University of Economics and Business
Harvard University

Submission date: 25.09.2024

Triple-Gamma-Regularization

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

by

Lucas Paul Unterweger, B.Sc. (WU)  GitHub
(Student-ID: 11913169)

Abstract

This thesis introduces the Triple-Gamma Regularization (TGR), a novel non-convex penalty term inspired by the Triple-Gamma prior, which provides increased flexibility in handling under- and overfitting in high-dimensional regression problems. The proposed method offers a unifying framework that bridges convex penalties like LASSO and non-convex penalties such as Gaussian and Arctan, allowing for fine-tuning of the regularization process through adjustable hyperparameters. First, a detailed theoretical derivation of the TGR is provided, highlighting its properties and advantages over existing regularization methods. Then, a comprehensive simulation study is conducted to evaluate the performance of TGR under various high-dimensional scenarios. The results indicate that TGR can outperform traditional regularization techniques in several settings, particularly in terms of minimizing mean squared error. This thesis also addresses the computational challenges associated with non-convex optimization problems and explores potential solutions through gradient-based optimization techniques. These findings contribute to the growing body of literature on regularization methods and offer practical insights for future applications in sparse modeling and high-dimensional data analysis.

Acknowledgements

This completion of this 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	1
2	Theoretical Section	3
2.1	Model Complexity and the Problem of Under- and Overfitting	3
2.1.1	Regularization	5
2.2	Bayesian View on Battling Model Complexity using Shrinkage Priors . .	7
3	Literature Review	8
4	Bayesian-Frequentist Duality of Ridge and LASSO Regression	11
4.1	Triple-Gamma-Prior by Cadonna et al. (2020)	13
5	Model Setup and Derivation	15
5.1	Varying the Hyperparameters	18
5.2	Comparison to already existing Penalty Terms	24
6	Simulation Study	26
6.1	Approaches to Estimation	26
6.2	Scenario Simulation	27
6.3	Simulation Results	28
6.4	Computational Efficiency	32
6.5	Implementation as Python Package	37
7	Possible Extensions and Criticism	38
8	Conclusion	39
9	Appendix	40
9.1	Technical Specification of System used for Simulation	40
	References	43

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. While his name may not be widely known, the principle that bears his name is well recognized: *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. Nonetheless, since then, the principle has long been used by statisticians and other researchers as a scientific credo to capture the notion that “the simpler of two explanations is to be preferred” (Lazar, 2010). Recent decades and their 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 enabled 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 trouble 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, which incorporate a penalty term into the optimization problems used to estimate the coefficients of linear models. In recent decades, various penalty terms have been published each with its own advantages and disadvantages, yet to this day no 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. Before that it is necessary to have a closer look at what the exact problem is that Bayesian shrinkage priors 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 structured 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 \mathbf{Y} with a set of input variables $\mathbf{X}_1, \dots, \mathbf{X}_p$ by trying to find a functional form f which uses the given information in the inputs and - ideally - describes the hidden relationship between \mathbf{Y} and the inputs \mathbf{X}_p 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 \mathbf{X}_i and the output \mathbf{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 we 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

In supervised learning, a model is fitted on an observed dataset where both inputs and outputs are given in pairs. Using this data a statistical model aims to find a functional form which maps these inputs into the output space and thus describes a relationship between those two spaces. Naturally, one would assume to find a functional form which manages to describe the relationship as closely as possible by minimizing a chosen loss function. Yet, this often ignores a problem called “overfitting” which can loosely be described as using “too much information” from the training dataset. As described by Ying (2019), this problem arises from the fact “that [an] over-fitted model has difficulty coping with pieces of the information in the testing set, which may be different from those in

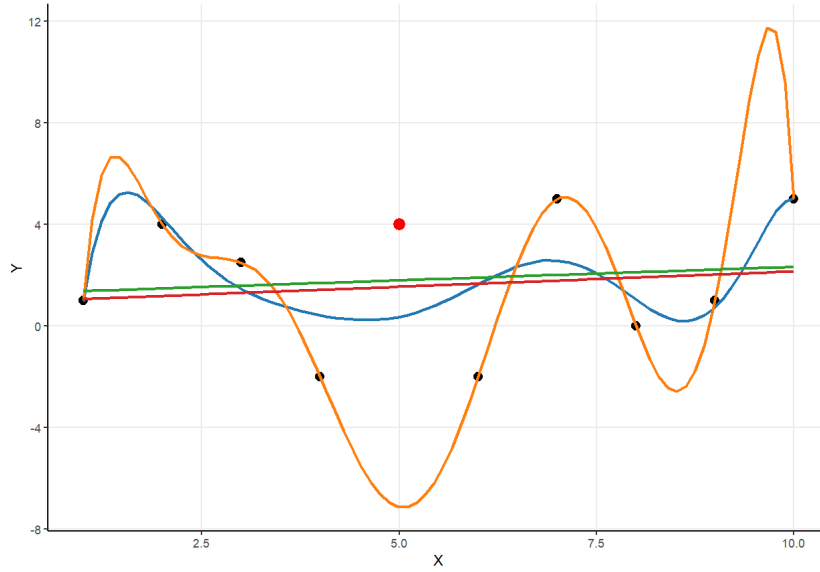


Figure 1: 1st and 8th order polynomial fit to data
(Green & Blue lines fitted with red data point; Orange & Red lines fitted without).

the training set”. A more formal definition of overfitting has been defined by Mitchell and Mitchell (1997):

Definition 1: Given a hypothesis space \mathcal{H} , a hypothesis $h \in \mathcal{H}$ is said to overfit the training data if there exists some alternative hypothesis $h' \in \mathcal{H}$ such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

Theoretically, this problem can emerge from a multitude of reasons. One of which being that learning every piece of information in the training dataset can lead to fitting the noise rather than fitting the underlying pattern. Another reason can be ill-chosen degrees of model complexity, because “having many hypotheses intuitively increase the risk of randomly finding a solution that learns the training set “by heart”, with limited generalization ability.” (Paris et al., 2003).

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. 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 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 estimated 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 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 defi-

nitions. 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 $L(\cdot)$ 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 (Hastie et al., 2009). Thus, resulting in the general class of regularization problems of the form:

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

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:

Definition 2: 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. Before advancing to literature that deals with this kind of problem, it is necessary to establish 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 data point consists of a target variable $y_i \in \mathbb{R}$ along with a number of corresponding inputs $x_i \in \mathbb{R}^p$. Given a linear function $f_{\beta}(x_i)$ of the inputs parametrized by coefficients $\beta \in \mathcal{B}$, $L(y_i, f_{\beta}(x_i))$ 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

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

It is important to note here that $J(\beta)$ is only a function of the coefficients β and neither the targets y_i nor the inputs x_i . Therefore, 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. Shrinkage priors, such as the *horseshoe* prior (Carvalho et al., 2010) or the *Triple-Gamma* prior (Cadonna et al., 2020), place a large amount of probability mass near zero while allowing for heavier tails, meaning that most parameter estimates are small, but the prior still allows for larger values when strongly supported by the data. This behavior structure of a probability distribution reduces the influence of irrelevant variables while retaining the effect of significant variables, thus improving both the predictive performance of the model in high-dimensional data settings, where often only a few variables are relevant in a set of multiple irrelevant variables. Unlike standard normal prior distributions, shrinkage priors actively encourage sparsity, making them particularly effective for feature selection and regularization in high-dimensional models (Piironen & Vehtari, 2017).

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 was derived from the previously known concept of Ridge analysis. At 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 form solution of the Ridge estimator is given by

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

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):

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

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 known as *LASSO*, developed by Tibshirani (1996). He argues that the two most prominent shrinkage methods at the time - 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 variable 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 extent *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 situation 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 the *Ridge* and *LASSO* 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 one 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

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

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

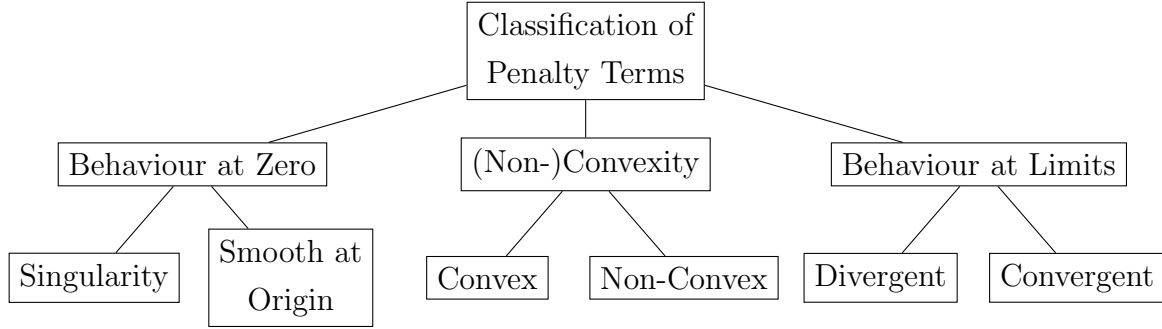


Figure 2: A possible classification of regularization methods based on the mathematical properties of the penalty function

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?) Such a classification tree can be found in figure 2.

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 1 classifies some of the existing concepts based on the three criteria mentioned earlier. A visualisation of them can be found in figure 8 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 Qi et al. (2022)) Convex penalties are generally easier to implement using common optimization algorithms than non-convex penalties but, more importantly, penalties which are divergent in 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 8).

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

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 overfitting 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 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 priori and the errors of the standard linear model are drawn from a normal distribution), that the mode 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 p variables and errors $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, then the likelihood of the data is given by

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

Under the previously mentioned assumption that $p(\beta) = \prod_{i=1}^k \beta_i$ 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 and taking the log of the distribution yields

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

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, shrinkage priors are a Bayesian approach to dealing 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 its representation is given by three gamma distributions. A key advantage of this prior is its unifying property. As can 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, the Bayesian LASSO can be seen as a special case of the triple-gamma-prior. Another advantage is its function as a bridge between spike-and-slab 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 be 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.

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 $\mathbf{y} = [y_1 \ y_2 \cdots y_n]^T$ and $\mathbf{x}_i = [x_{i1} \ x_{i2} \cdots x_{ip}]^T$ with $\forall i \in \{1, \dots, n\} : y_i, x_{ij} \in \mathbb{R}$. Here, x_i is the i -th data point, thus resulting in the design matrix $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \cdots \mathbf{x}_n]^T$.

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

$$y_i = \mathbf{x}_i^T \cdot \beta + \varepsilon_i, \quad i \in \{1, \dots, n\},$$

with the assumed distribution of $\varepsilon_i \sim N(0, \sigma^2)$. Thus it follows that $y \sim N_n(\mathbf{X}\beta, \sigma^2 I)$. 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.

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

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

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

The log of the likelihood function is then given by

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

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

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

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

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

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 ($\Re(a) > 0$) and again apply proportionality.

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

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

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

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

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

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

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

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

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**:

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

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 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 result, 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 3. As already mentioned earlier, the necessary restriction 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 becomes 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).

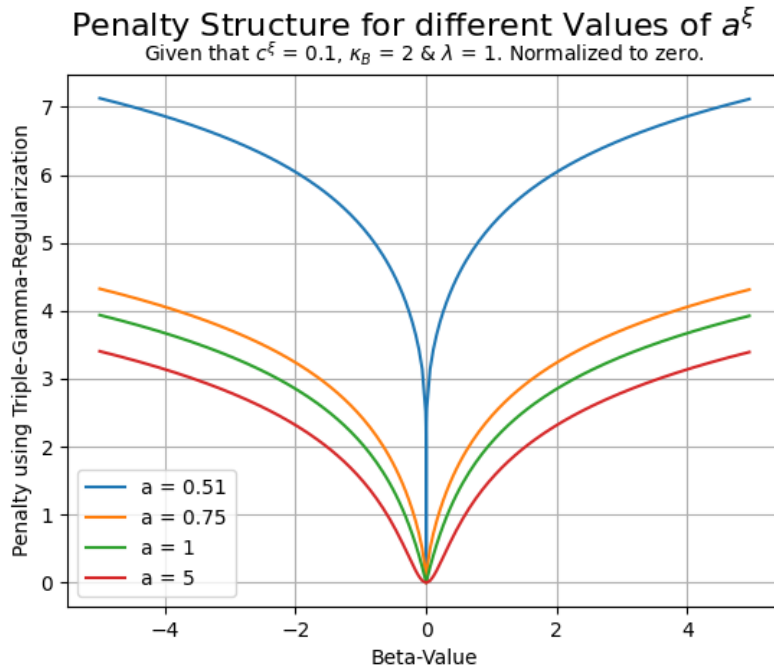


Figure 3: Triple-Gamma-Penalty using different values of a^ξ

Seeing this, it is apparent that a value of a^ξ close but strictly larger than 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 4 (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 even more convincing that the value is significantly different from zero.

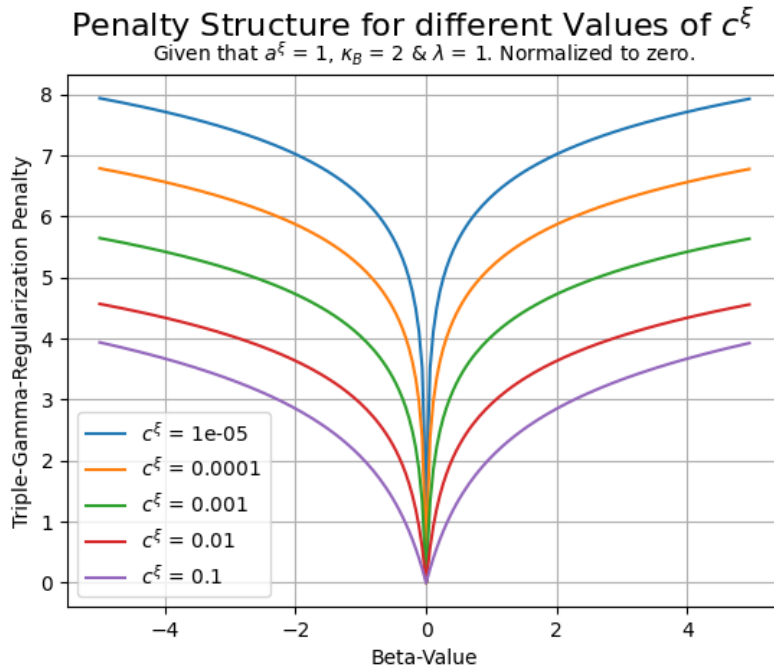


Figure 4: Triple-Gamma-Penalty using different values of c^ξ with $0 < c^\xi \leq 0.1$

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 5, 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.

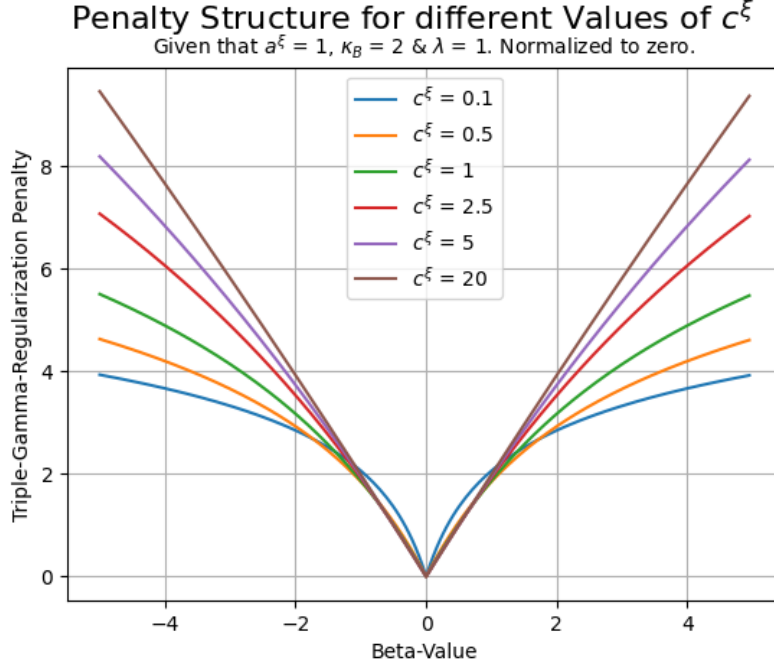


Figure 5: Triple-Gamma-Penalty using different values of c^ξ , with $c^\xi \geq 0.1$

Variations of the Hyperparameter κ_B

The third and final hyper-parameter κ_B^2 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 6. 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.

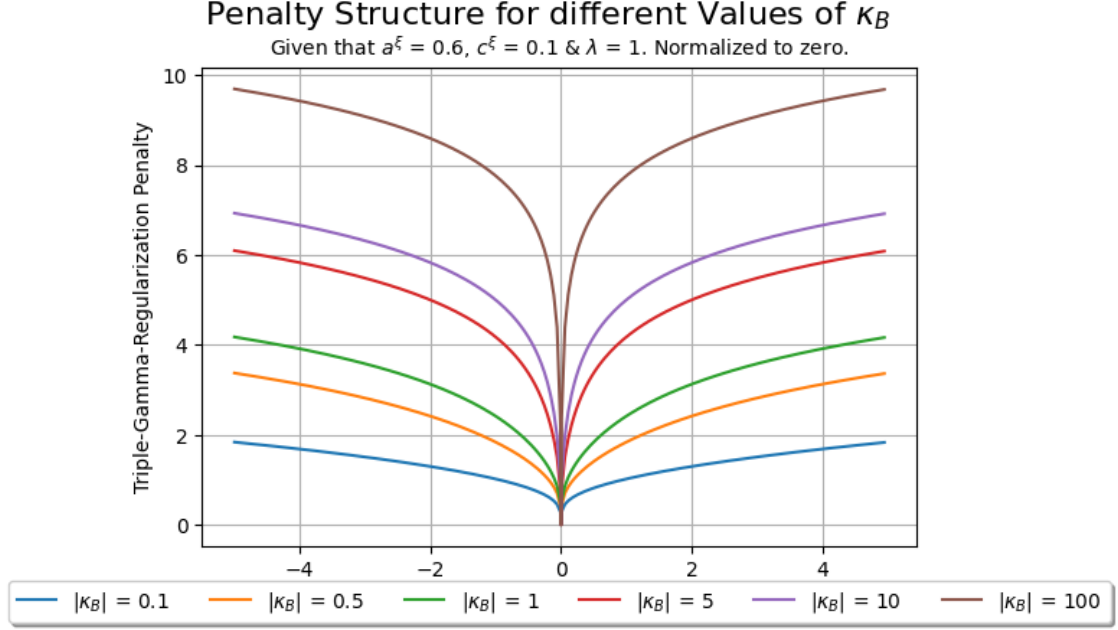


Figure 6: Triple-Gamma-Penalty using different values of κ_B

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.

Variable	Change		Mathematical Properties	
	Positive Change	Negative Change	Defined Range	Misc.
a^ξ	Shifting towards <i>Gaussian</i> -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 <i>prior knowledge</i> 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 themselves 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 7. 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.

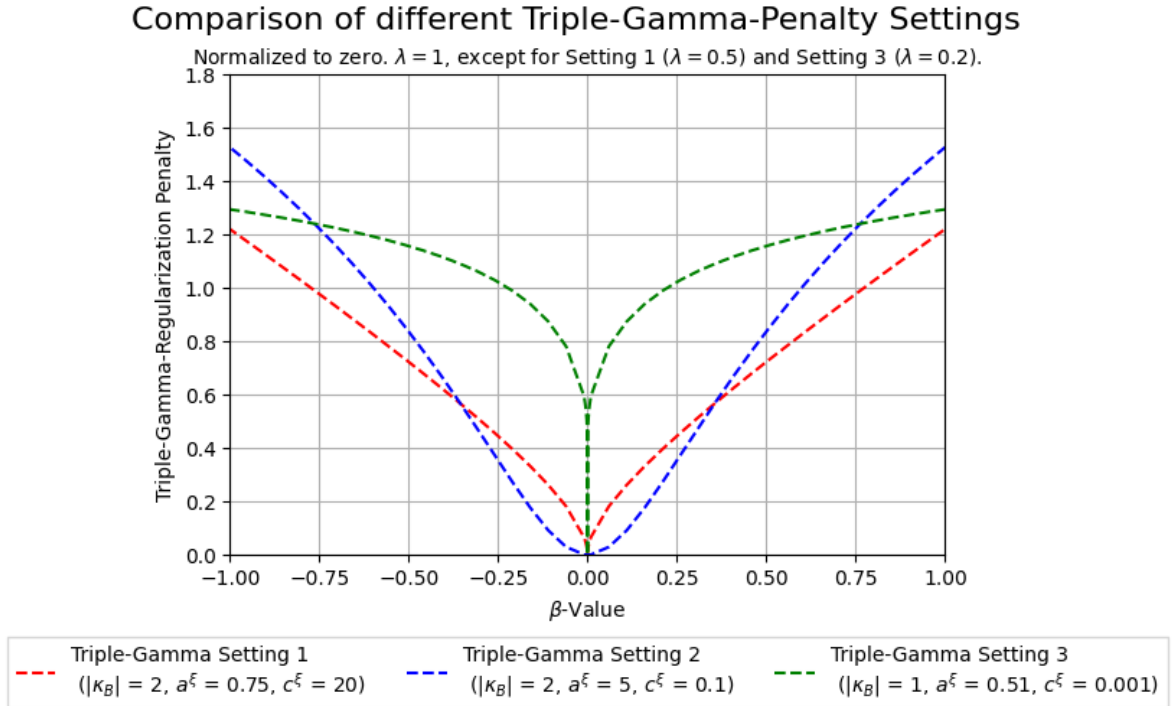


Figure 7: Different settings of the triple-gamma-penalty

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 off very quickly and thus adds relatively little penalty for parameters that are above 1 in absolute terms.

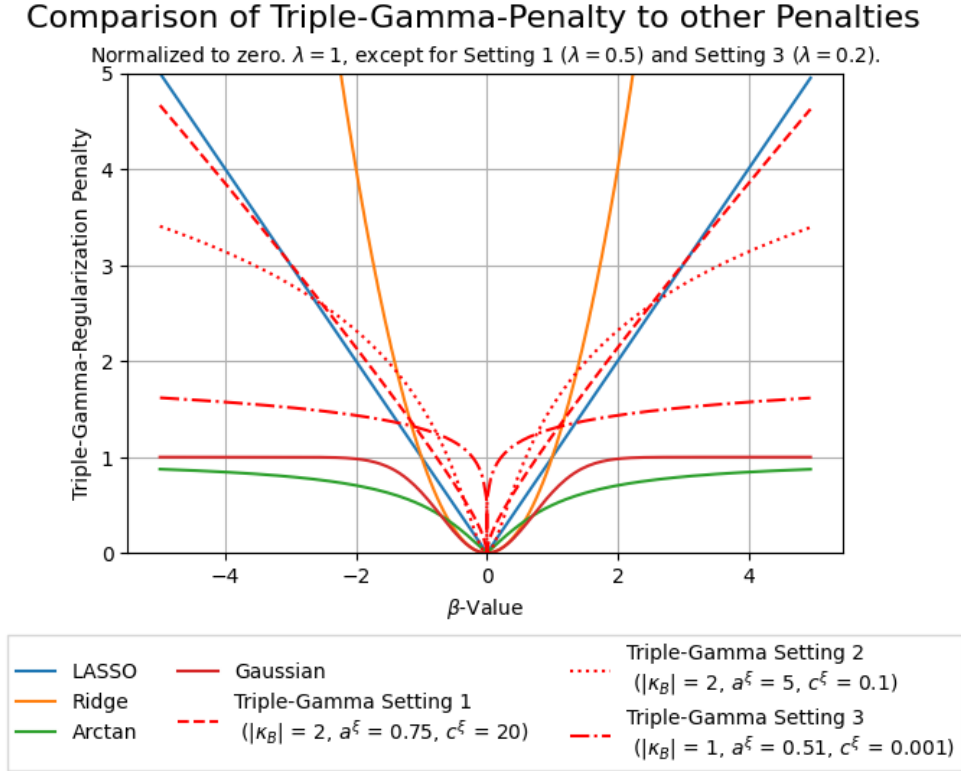


Figure 8: Basic representation of the triple-gamma-penalty using hyperparameters $c^\xi = 0.1, \kappa_B = 2, a^\xi = 0.75$ compared to *LASSO* and *ridge* penalties

Additionally, a comparison of these *Triple-Gamma* settings to a selection of other regularization penalties can be found in figure 8. 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 8 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 equation 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 (Grippo & Sciandrone, 2023). 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 *Stochastic 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)*. 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 these values, all possible combinations of these variable settings will be formed, resulting in a total of 27 possible scenarios.

Variable	Values
Sample Size N	10, 100, 250
Dimensionality p	10, 25, 50
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. After this run, these optimal values will be used in the overall

simulation. For the application of the gradient descent algorithm a maximum number of 500 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 Results

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 sd_i , the first $s_i * p_i$ coefficients, rounded to the next larger integer, will be set to 3 and the remaining will be set to 0. 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 generated from a normal distribution with mean 0 and variance σ_i . According to F. Wang et al. (2020), the respective σ_i will be computed based on the *Signal-To-Noise Ration (SNR)*. The *SNR* is given by $SNR = \sqrt{\beta^T X^T X \beta / (N_i \sigma_i^2)}$ and by setting the *SNR* to a specific value, one can adjust how strong the noise in the data is relative to the sound of the actual data generating process. For the purpose of this study, and similar to F. Wang et al. (2020), a *SNR* equal to one will be used. In terms of the regularization parameter λ , the search for the optimal λ will be performed by using five-fold-cross-validation on the a set of possible λ values. For every method and every run in every scenario separately, the optimal λ_i^* will be chosen which generates the lowest validation loss on the testset. This grid-search procedure will be performed for every $\lambda_i \in \{0.001, 0.01, 0.1, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9, 0.95\}$.

Using the set of scenario variables mentioned in table 4, all possible combinations will be formed resulting in the overall 27 scenarios.⁴ The code iterates through the list of scenarios, generates 15 datasets based on these scenarios variables and all methods will be estimated on the test part of these datasets. Afterwards, the validation loss will be computed for each method and stored. This procedure results in table 6, which shows the mean and standard deviation of the distribution of the validation loss for every scenario and every method. Before advancing to the overall results of this simulation study and its interpretation, it might be useful to have a look at a specific scenario to better understand the results in table 6, which will be the scenario with $N = 250, p = 25, s = 0.9$.

In this scenario, we are dealing with a dataset of 250 observations, 25 features and

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

⁴Note: Due to the computational constraints of hardware used for this simulation and several other issues, not all 27 scenarios were estimated in one run, but rather in several separate runs. More information on the exact procedure to ensure reproducibility can be found in the relevant script on [GitHub](#).

23 non-zero coefficients.⁵ 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 9.

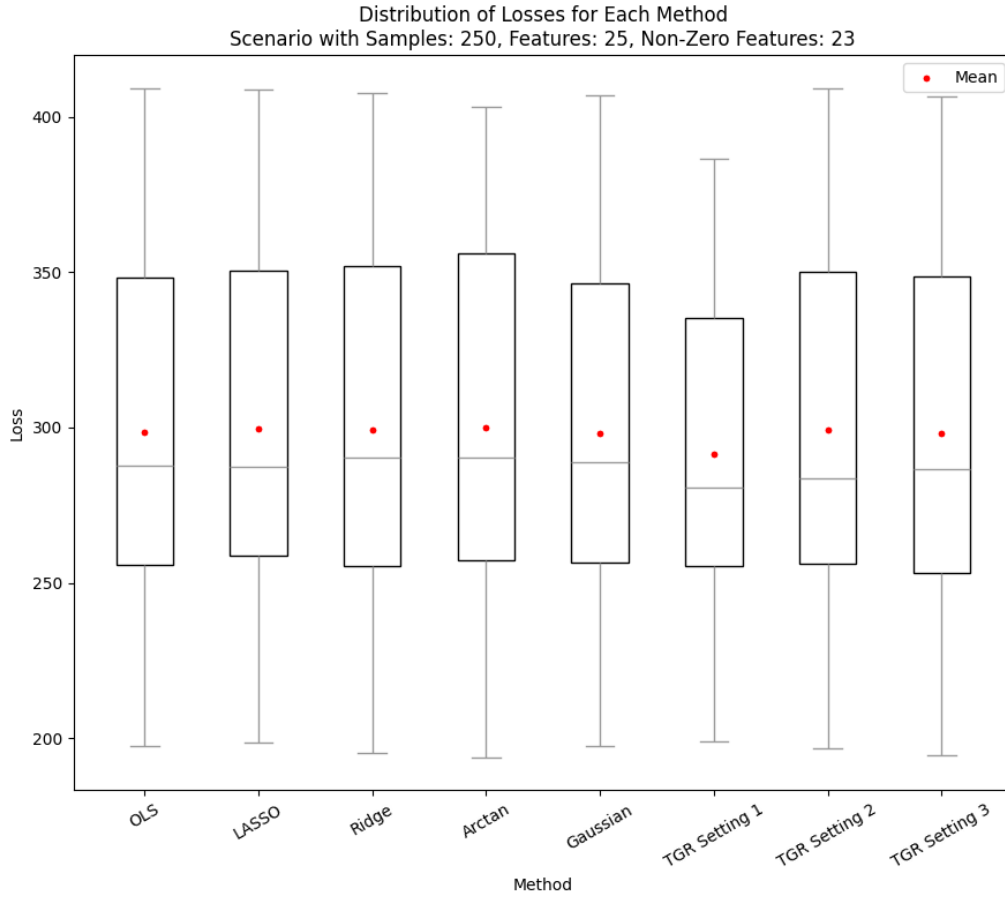


Figure 9: Distribution of MSE on validation set in the scenario with 250 observations, 25 features and 23 non-zero coefficients

This scenario, which is a good example for a dense scenario, where a lot of features are in the dataset and a major part of it is different from zero, i.e. has an effect on the outcome. Here, setting 1 of the *Triple-Gamma-Penalty* performs best as it manages to produce the smallest mean in loss with about 291.426. All other methods perform as good as or slightly better than the regular *OLS* approach. A second scenario can be found in figure 10, where the scenario with 100 observations, 50 features and 25 non-zero features is shown. Here, again the setting 1 of the *Triple-Gamma-Penalty* seems to do best when it comes to the lowest mean *MSE*.

⁵ $25 \times 0.9 = 22.5$, which is then rounded up to 23.

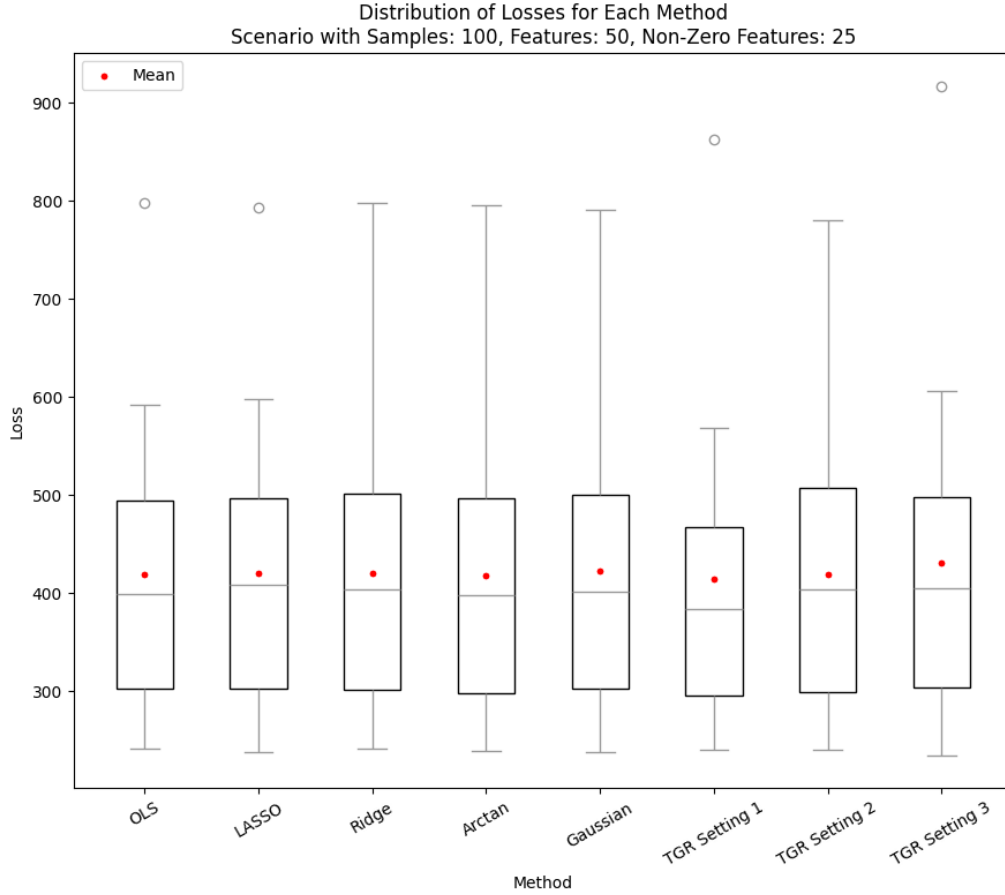


Figure 10: Distribution of MSE on validation set in the scenario with 100 observations, 50 features and 25 non-zero coefficients

An overview of the basic results for each of the 27 scenarios can be found in table 5. In the left half of the table the scenario parameters are presented and in the right half the best method based on the method having the lowest mean as well as median validation loss. In general, the results confirm the results from F. Wang et al. (2020) that no go-to method in regularization has been found. In terms of the mean loss, Setting 1 and 2 of the *Triple-Gamma-Penalty* as well as the *Gaussian* approach seem to work well. Setting 3 of the *Triple-Gamma-Penalty* only comes out on top in two scenarios. *LASSO* only manages to perform best in one scenario when using the median loss as the method of comparison. *Arctan* interestingly never performs best when it comes to the lowest mean loss, but shows up a few times in the median column. The *OLS* approach, hence a regularization parameter of $\lambda = 0$ performs best in a few of the scenarios with sample size 10, however seeing that these scenarios are trying to estimate 25 and 50 parameters based on ten samples, these results are suffering from high variance and thus should not be taken too serious. As previously mentioned, a table with the means and standard deviations of all methods in all scenarios can be found in table 6.

Generally, it can be said that certain settings of the *Triple-Gamma-Penalty* can indeed

Scenario Parameters			Moment	
Samples	Features	Non-Zero Features	Best Method (Mean)	Best Method (Median)
10	10	1	TGR Setting 2	TGR Setting 2
10	10	5	TGR Setting 2	TGR Setting 2
10	10	9	Gaussian	Arctan
10	25	3	TGR Setting 1	Arctan
10	25	13	TGR Setting 2	Gaussian
10	25	23	OLS	TGR Setting 3
10	50	5	TGR Setting 1	Arctan
10	50	25	OLS	Gaussian
10	50	45	OLS	TGR Setting 2
100	10	1	Gaussian	LASSO
100	10	5	TGR Setting 1	TGR Setting 1
100	10	9	OLS	Arctan
100	25	3	Gaussian	Gaussian
100	25	13	Ridge	TGR Setting 1
100	25	23	TGR Setting 1	TGR Setting 1
100	50	5	Gaussian	Arctan
100	50	25	TGR Setting 1	TGR Setting 1
100	50	45	Ridge	Arctan
250	10	1	Gaussian	LASSO
250	10	5	TGR Setting 2	OLS
250	10	9	TGR Setting 1	TGR Setting 2
250	25	3	Gaussian	Arctan
250	25	13	TGR Setting 1	TGR Setting 2
250	25	23	TGR Setting 1	TGR Setting 1
250	50	5	TGR Setting 2	Gaussian
250	50	25	TGR Setting 3	OLS
250	50	45	TGR Setting 3	Ridge

Table 5: Table showing the best performing methods according to smallest Mean and Median Loss accordingly

perform better than the existing methods in this specific scenarios. However, not a single setting seems to outperform the other settings over a broad range of scenarios.

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 on a local system.⁶ 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 arctangent 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 and time-intensive. In figure 11 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 $s = 0.1$ and an increasing number of features p .

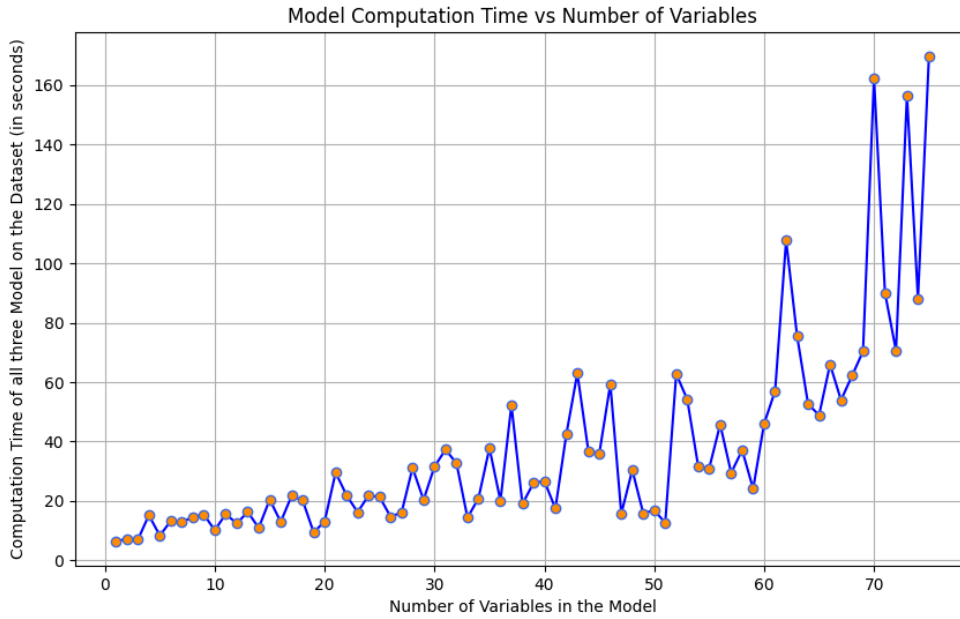


Figure 11: Time necessary to compute all three scenarios of the Triple-Gamma-Regularization with $N = 100$ and $s = 0.1$ dependent on the number of Features p

As can be seen from figure 11 above, the time to estimates these models increases drastically with an increased number of variables. Setting aside that it might even grow

⁶The technical specifications of this system can be found in the Appendix.

exponentially, it can be seen that the computation time also increasingly becomes more unstable. Several approaches might help in resolving these issues, like optimizing the underlying approximation of the confluent hypergeometric function 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.

It is also worth noting that the computational efficiency of the *Triple-Gamma-Penalty* still relies on the specification of the penalty itself. *Setting 2* for example with its very *smooth-at-origin* behaviour takes significantly longer to compute compared to its *singular-at-origin* specifications in *setting 1* and *setting 3*. A plot showcasing these differences can be found in figure 12 below. For an specific scenario, in this case with 100 samples, 10 features and one non-zero feature, 25 runs have been computed and timed. It can clearly be seen that settings 1 and 2 usually take about 2.5 seconds to estimate, whereas the *smooth-at-origin* specification takes about 6 seconds on average. This disparity increases with an increasing number of features. All in all, this is an open issues which needs to be studied further to increase practicability in real-world applications.

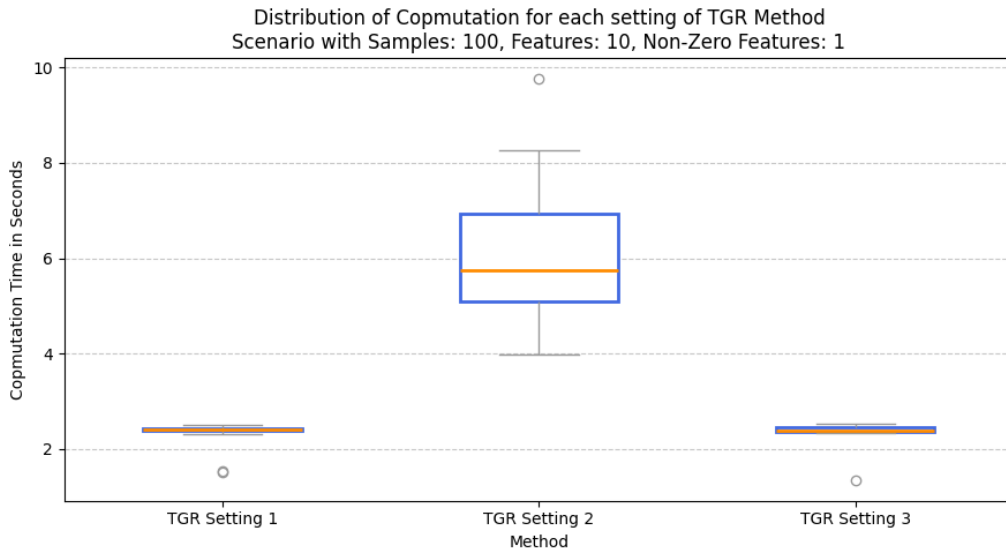


Figure 12: Distribution of computation time for different settings of the TGR (Samples: 100, Features: 10, Non-zero Features: 1, 500 runs)

Table 6: Mean and Standard Deviations of MSE on the Validation Set for different Scenarios (15 runs per scenario)

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	Arctan	Gaussian	TGR Setting 1	TGR Setting 2	TGR Setting 3
10	10	1	12.746 (14.993)	7.163 (5.372)	8.589 (5.595)	9.551 (13.523)	6.751 (6.217)	7.722 (4.92)	5.348 (3.151)	6.663 (4.127)
		5	83.12 (59.984)	84.273 (62.912)	81.288 (53.656)	85.04 (61.694)	84.585 (66.75)	80.645 (51.107)	79.03 (50.777)	89.188 (57.828)
			220.161 (360.491)	224.464 (365.323)	221.269 (354.385)	219.74 (356.702)	219.399 (354.232)	228.631 (356.692)	219.837 (355.328)	239.05 (335.249)
25	25	3	50.664 (49.508)	44.771 (34.833)	44.945 (40.545)	48.249 (40.413)	53.057 (40.557)	37.565 (31.623)	37.675 (29.551)	38.136 (32.025)
		13	235.981 (256.182)	234.756 (260.716)	238.259 (261.011)	235.193 (259.992)	237.285 (267.622)	233.065 (251.616)	231.373 (249.771)	261.546 (311.695)
			657.683 (927.439)	663.27 (948.327)	682.629 (969.617)	663.035 (950.083)	665.583 (950.939)	684.95 (986.148)	661.124 (931.974)	707.196 (1055.199)
50	50	5	85.213 (65.938)	86.522 (63.037)	87.626 (64.062)	82.807 (61.584)	92.967 (67.136)	72.882 (52.811)	86.194 (62.525)	73.482 (55.999)
		25	380.704 (348.464)	393.886 (369.79)	383.191 (358.369)	387.278 (362.043)	385.417 (362.873)	385.333 (347.936)	400.293 (356.349)	407.237 (380.271)
			586.161 (509.473)	595.584 (502.119)	587.042 (503.356)	589.93 (516.23)	595.248 (517.682)	628.681 (532.073)	599.853 (516.79)	625.393 (567.819)

Continued on next page

Table 6 – continued from previous page

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	Arctan	Gaussian	TGR Setting 1	TGR Setting 2	TGR Setting 3
100	10	1	9.424 (3.128)	9.053 (3.351)	9.395 (3.177)	8.864 (3.296)	8.679 (2.888)	9.503 (3.212)	8.691 (3.022)	9.393 (3.125)
			57.358 (18.573)	56.949 (18.757)	57.351 (18.811)	56.565 (18.628)	57.245 (18.709)	55.015 (18.942)	57.423 (19.208)	56.856 (18.704)
			116.117 (44.899)	117.717 (46.085)	116.924 (45.616)	116.152 (45.468)	116.713 (47.097)	118.196 (46.824)	117.065 (45.264)	117.579 (45.929)
		5								
		9								
	25	3	32.254 (13.184)	30.553 (11.05)	33.468 (13.544)	30.841 (12.644)	29.089 (12.183)	32.741 (12.647)	29.98 (12.085)	33.19 (13.086)
			174.105 (74.371)	174.086 (71.812)	172.684 (72.5)	174.514 (75.149)	173.975 (72.858)	173.319 (71.67)	175.257 (72.028)	174.474 (73.942)
			372.536 (119.118)	375.219 (122.728)	372.548 (119.272)	374.19 (121.703)	374.707 (121.828)	370.122 (124.796)	375.738 (122.027)	371.922 (119.823)
250	50	5	74.46 (21.745)	71.822 (21.782)	73.924 (20.749)	72.343 (21.084)	68.12 (22.565)	73.804 (20.536)	69.25 (23.649)	75.756 (20.395)
			419.029 (150.3)	420.748 (149.616)	420.585 (151.05)	418.464 (150.909)	422.38 (149.169)	414.026 (162.605)	419.259 (150.257)	430.39 (174.063)
			660.425 (219.371)	658.744 (221.85)	657.582 (220.248)	659.611 (220.094)	661.732 (220.315)	661.103 (225.032)	662.506 (222.481)	661.674 (217.426)
		45								
	10	1	8.797 (1.74)	8.692 (1.599)	8.825 (1.708)	8.673 (1.656)	8.661 (1.674)	8.77 (1.741)	8.669 (1.667)	8.773 (1.726)

Continued on next page

Table 6 – continued from previous page

Sample Size	Features	Non-zero Features	OLS	LASSO	Ridge	Arctan	Gaussian	TGR Setting 1	TGR Setting 2	TGR Setting 3
		5	48.357 (7.664)	48.502 (7.806)	49.367 (7.531)	48.327 (7.523)	47.985 (7.308)	47.944 (7.69)	47.939 (7.505)	48.215 (7.747)
			95.877 (23.697)	96.414 (24.141)	95.621 (23.726)	96.268 (23.569)	97.287 (23.637)	95.458 (23.909)	96.026 (23.563)	96.794 (22.755)
			30.641 (7.932)	29.895 (6.621)	31.892 (7.391)	29.45 (7.185)	28.34 (5.67)	29.621 (7.539)	28.742 (6.161)	30.599 (7.725)
25		3	157.836 (45.342)	158.001 (44.129)	157.34 (44.617)	156.739 (44.256)	157.788 (44.235)	154.68 (46.043)	157.53 (45.347)	157.915 (44.678)
			298.676 (60.928)	299.437 (60.018)	299.121 (60.275)	299.933 (60.418)	298.27 (60.256)	291.426 (54.161)	299.305 (61.564)	298.262 (60.842)
			58.478 (8.895)	56.326 (8.352)	60.041 (8.732)	55.643 (8.258)	52.592 (8.083)	58.55 (9.343)	52.532 (8.209)	59.195 (8.924)
50		5	352.75 (70.694)	353.179 (72.004)	352.526 (71.89)	353.512 (72.101)	353.35 (71.954)	352.349 (72.133)	352.161 (68.27)	352.072 (71.796)
			664.144 (133.001)	663.053 (131.957)	661.547 (129.346)	663.788 (128.434)	666.384 (130.161)	663.345 (131.016)	665.715 (130.906)	660.788 (128.093)

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 <code>nn.Module</code> and initializes a linear model with 10 input features and 1 output feature without an intercept.
TripleGammaRegLoss	This function computes the loss for the Triple Gamma Regularization. It takes the predicted values (<code>Y_HAT</code>), the true values (<code>Y</code>), the coefficients of the model (<code>coefficients</code>), the regularization parameter (<code>lamda</code>), and the regularization parameters <code>a</code> , <code>c</code> and <code>kappa</code> as input. The loss is computed according to the theoretical formulation by using the script <code>log_hyperu.py</code> , which approximates the log of the confluent hyper-geometric function of the second kind.
TripleGammaModel	This function trains the <i>Triple-Gamma</i> Regularization model. It takes the feature matrix (<code>X</code>), the target values (<code>y</code>), the regularization parameter (<code>penalty</code>), regularization hyper-parameters <code>a</code> , <code>c</code> , <code>kappa</code> , the number of epochs (<code>num_epochs</code>), and the learning rate (<code>lr</code>) as input. It initializes the <code>PyTorch</code> 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

The *Triple-Gamma-Penalty* in its current form is very rudimentary and leaves plenty space for further scientific research and simulations. The three settings of the penalty given in figure 7 are examples of how the penalty could be used and are showcasing its flexibility by using different hyperparameters. Nonetheless, having highlighted some of the advantages of using the *Triple-Gamma-Regularization* approach in previous chapters, it is only reasonable to mention some of the downsides of it compared to methods in the existing body of literature as well. First, the introduced flexibility comes at the cost of mathematical simplicity. Existing methods like *LASSO* and *Ridge* still have the upper hand in practical applications due to its remarkable mathematical simplicity. Even more modern methods like *Gaussian* or *Arctan* regularization, which still only use rudimentary mathematical components, have troubles with establishing themselves as viable alternatives in practical application. It remains to be seen if the advantages of the *Triple-Gamma* penalty outweigh its obvious downsides in this regard. Second, as statistical models become increasingly more extensive in the age of AI, computational efficiency, even though computational power has increased as well in recent decades, still remains an issue of utmost importance. This surely is a problem which needs to be solved or at least handled in an appropriate manner to make the method more viable.

On the upside, these shortcomings still leave room for improvement and further research. Most importantly it will be important to explore the viability of the *Triple-Gamma* penalty in other simulation settings and in different contexts. For example,

more scenarios can be analysed with different data generating processes. In addition, these simulations can use more runs and more iterations in its estimation process to gain more robust results. As previously mentioned, an entire new estimation algorithm could be used as well to not only make the results more robust but possibly also solve the issue of computational efficiency. Another field of further research could explore more possibilities in the area of setting the hyperparameters. The three settings used in this thesis only represent a small subgroup of possible penalty structures that the *Triple-Gamma* penalty can take on. A more extensive analysis of these settings or even a possible hands-off approach to setting these values could open up an entire new field of applications. Lastly, the *Triple-Gamma* penalty has only been applied to a linear model setup, which is commonly used when dealing with cross-sectional data. As the variety of statistical models has increased in recent decades, it might also be interesting to see how this penalty applies to neural network scenarios or even time series problems.

8 Conclusion

This thesis introduced the *Triple-Gamma Regularization (TGR)*, a novel non-convex penalty inspired by the Triple-Gamma Prior developed by Cadonna et al. (2020). This new regularization method bridges existing convex approaches like LASSO with newer non-convex penalties such as Arctan and Gaussian, offering increased flexibility in addressing the challenges of underfitting and overfitting in high-dimensional regression settings. The theoretical foundation of TGR was derived by leveraging the mathematical connection between frequentist regularization techniques and Bayesian shrinkage priors. Through this connection, a penalty structure was derived which allows for a more flexible tuning of hyperparameters to achieve a balance between bias and variance. The flexibility of TGR lies in its three hyperparameters, which control the behaviour of the penalty structure at both the origin and the tails. This allows it to handle scenarios where existing penalties may fall not sufficiently solve problems.

A key feature of *TGR* is its ability to perform both continuous shrinkage and variable selection, making it a versatile alternative in high-dimensional data analysis. Compared to existing regularization methods, such as LASSO and Ridge, *TGR* offers the advantage of a more refined approach to penalization. This was supported by the simulation study conducted, which demonstrated that TGR performs competitively across a variety of high-dimensional settings, sometimes even outperforming conventional methods in terms of mean-squared error reduction. Downsides include the computational challenges posed by non-convex optimization problems, which are inherent in methods like *TGR*. While the results presented here suggest that the *Triple-Gamma* penalty is a promising addition to the regularization literature, it does not come without limitations. The penalty's

performance heavily depends on the tuning of hyperparameters, and further research is needed to refine selection procedures for these parameters.

All in all, the Triple-Gamma Regularization offers a novel, flexible, and effective approach to penalized regression in high-dimensional settings. Its ability to unify and extend existing methods provides a valuable tool for both theoretical research and practical application in sparse modeling and high-dimensional data analysis.

9 Appendix

9.1 Technical Specification of System used for Simulation

Specification	Details
Operating System	Microsoft Windows 11 Pro, Version 10.0.22621
Processor	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 4 Cores, 8 Threads
RAM	16 GB DDR4
Graphics	Intel(R) UHD Graphics 620
Storage	1TB SSD (C:)
BIOS Version	LENOVO N2HET74W (1.57), 01.06.2023
Virtualization Support	Enabled (Hyper-V, SLAT, Virtualization in Firmware)
Available Virtual Memory	28.8 GB
Available Physical Memory	15.8 GB

Table 8: System specifications for simulation

List of Figures

1	1st and 8th order polynomial fit to data (Green & Blue lines fitted with red data point; Orange & Red lines fitted without).	4
2	A possible classification of regularization methods based on the mathematical properties of the penalty function	10
3	Triple-Gamma-Penalty using different values of a^ξ	19
4	Triple-Gamma-Penalty using different values of c^ξ with $0 < c^\xi \leq 0.1$. . .	20
5	Triple-Gamma-Penalty using different values of c^ξ , with $c^\xi \geq 0.1$	21
6	Triple-Gamma-Penalty using different values of κ_B	22
7	Different settings of the triple-gamma-penalty	24
8	Basic representation of the triple-gamma-penalty using hyperparameters $c^\xi = 0.1, \kappa_B = 2, a^\xi = 0.75$ compared to <i>LASSO</i> and <i>ridge</i> penalties . . .	25
9	Distribution of MSE on validation set in the scenario with 250 observations, 25 features and 23 non-zero coefficients	29
10	Distribution of MSE on validation set in the scenario with 100 observations, 50 features and 25 non-zero coefficients	30
11	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	32
12	Distribution of computation time for different settings of the TGR (Samples: 100, Features: 10, Non-zero Features: 1, 500 runs)	33

List of Tables

1	Classification of several Penalties	11
2	Summary of the effects of changes in the hyperparameters a^ξ , c^ξ and κ_B on the penalty structure	23
3	Three example settings of the <i>triple-gamma-penalty</i>	24
4	Scenarios Values	27
5	Table showing the best performing methods according to smallest Mean and Median Loss accordingly	31
6	Mean and Standard Deviations of MSE on the Validation Set for different Scenarios (15 runs per scenario)	34
8	System specifications for simulation	40

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.
- Carvalho, C. M., Polson, N. G., & Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2), 465–480.
- 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.
- Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol. 1). McGraw-hill New York.

- Paris, G., Robilliard, D., & Fonlupt, C. (2003). Exploring overfitting in genetic programming. *International Conference on Artificial Evolution (evolution Artificielle)*, 267–277.
- Piironen, J., & Vehtari, A. (2017). Sparsity information and regularization in the horse-shoe and other shrinkage priors.
- Qi, Z., Cui, Y., Liu, Y., & Pang, J.-S. (2022). Asymptotic properties of stationary solutions of coupled nonconvex nonsmooth empirical risk minimization. *Mathematics of Operations Research*, 47(3), 2034–2064.
- 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.
- Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168, 022022.
- 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.