ETH
Swiss Federal Institute of Technology Zurich

Seminar for Statistics

Department of Mathematics

Master Thesis Spring 2024

Yiting Nian

Optimization methods for Gaussian process hyper-parameter estimation

Submission Date: June 14, 2024

Advisor: Dr. Fabio Sigrist

Abstract

Gaussian process (GP) is a flexible statistical and machine learning model that is widely used in applications involving time series or spatial data. To make predictions based on this model, several hyper-parameters in the covariance function have to be estimated. In this thesis, we perform a systematic comparison of several commonly used optimization methods, which based on gradients, secondary derivatives, or heuristic search strategy, for the estimation of Gaussian process hyper-parameters from the perspective of training speed, convergence performance, pros and cons. Approximation methods are used for the large spatial data to reduce the computation burden. Simulated data samples of different settings and distributions will be used to test the performance of the optimization methods, then theoretical and experimental comparison will be summarized.

Contents

List of Figures

List of Tables

Notation

GP - Gaussian process

GD - Gradient Descent

BFGS - Broyden–Fletcher–Goldfarb–Shanno

LBFGS - Limited-memory BFGS

FITC - Fully Independent Training Conditional

Chapter 1

Introduction

Gaussian process (GP) regression, or kriging (Eidsvik (2011)), is a popular non-parametric function model that combines the characteristics of easy-implementing, flexibility, and modelling accuracy. Gaussian Processes (GPs) have applications in various areas, such as non-parametric regression, time series or spatial data, and emulation of large computer models (Kennedy and O'Hagan (2001)). The defining features of a GP are its mean function and covariance function (kernel), which are parameterised by different variables; depending on different classes of covariance function. Hyper-parameters in GPs include parameters of the covariance function, the noise variance and, in some cases, parameters of the mean function. The covariance function, which defines the similarity between data points, is particularly sensitive and crucial; as it directly affects the smoothness and overall shape of the predicted function. In this thesis, we focus on the Matérn class of covariance functions (Williams and Rasmussen (2006)).

The practical utility of GPs largely depends on the accurate estimation of hyper-parameters, a process that can be challenging due to the non-convex, often multimodal nature of the likelihood function and the computational infeasibility for many large datasets. The optimization of GP hyper-parameters is typically achieved by maximizing the marginal likelihood (Williams and Rasmussen (2006)), which is non-trivial due to the aforementioned characteristics. Consequently, various optimization methods can be employed to tackle this problem, each with its own advantages and limitations. Gradient-based methods, such as gradient descent (GD), which is one of the oldest and best known numerical optimization methods, and its variants, such as Nesterov Accelerated Gradient Descent (Nesterov (2004)), leverage the gradient information to navigate the iteration steps. Moreover, Newton's method uses the second-derivative information to approach the local optima, and quasi-Newton methods; such as Fisher Scoring, BFGS and limited-memory BFGS (L-BFGS), are also commonly considered in unconstrained optimization (Nocedal and Wright (2006)). On the other hand, derivative-free methods offer an alternative by exploring the parameter space without relying on gradient information.

This thesis aims to provide a comprehensive comparison of these different optimization methods in the context of Gaussian process hyper-parameter estimation, by systematically evaluating their performance, computational efficiency, and robustness across different scenarios and datasets. Such an analysis will provide practical insights for selecting the most appropriate approach for specific applications.

Chapter 2 describes the concepts of Gaussian process, hyper-parameter estimation, and

techniques of six optimization methods. Chapter 3 presents the experimental results of (i) simulated GPs with different settings, (ii) GPs with model mis-specification, and (iii) deterministic functions.

Chapter 2

Background

2.1 Gaussian Process

Gaussian process (GP) is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. A Gaussian process model can be fully specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, denoted as $Y \sim \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ in $\mathbb{R}^d (d \geq 1)$. The covariance function defines the similarity between data points, and it can reflect the prior information of the distribution from which the data are drawn. In this thesis, we focus on the Matérn class of covariance functions (Williams and Rasmussen (2006)), which is defined as

$$k_{\mathrm{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu r}}{\ell} \right), \ \nu, \ell > 0 \tag{1}$$

where K_ν is a modified Bessel function, ν and ℓ are the smoothness and range parameter of the kernel, respectively. The covariance function is assumed to be the addition of the signal term and an independent noise term:

$$\sigma_f^2 \cdot k_{\mathrm{Matern}}(r) + \sigma_n^2 \cdot I_{r=0} \tag{2}$$

where σ_f^2 and σ_n^2 are the signal and noise variance, respectively.

According to different smoothness ν, the Matérn class of covariance function can actually be recognized as some more familiar covariance function. For example, the covariance function in Eq.(1) converges to the squared exponential covariance (Gaussian covariance) as $\nu \to \infty$, or exponential covariance when $\nu = 0.5$. Moreover, $\nu = 1.5$ and $\nu = 2.5$ are also two interesting covariance functions for many machine learning problems.

We use four types of covariance functions with smoothness $\nu \in (0.5, 1.5, 2.5, \mathrm{inf})$ in our experiments and assume a zero mean function in most of the experiments (except in Section 3.1.2 and 3.1.4, where we investigate the influence of an additional linear regression term).

2.2 Hyperparameter estimation

Training a GP model is actually finding the optimal estimate of the parameters in the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$, also known as *model selection*.

In this thesis, we will only consider the GP models with the mean function $m(\mathbf{x})$ as zero or linear regression terms. In addition to the potential linear regression coefficients $\boldsymbol{\beta}$, the hyper-parameters $\ell, \sigma_f^2, \sigma_n^2$ of GP models are estimated using *maximum likelihood estimation* (MLE) method. The log marginal likelihood of the observed data is given in Eq.(3) (Williams and Rasmussen (2006)).

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log(2\pi), \tag{3}$$

where K_y is the covariance function given in Eq.(2), and $\mathbf{y}$ is replaced by $\mathbf{y} - Z\boldsymbol{\beta}$ in the presence of the linear regression term, represented by $Z\boldsymbol{\beta}$. The derivative of the log marginal likelihood with respect to the hyper-parameters $\boldsymbol{\theta}$ is:

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - K_y^{-1})\frac{\partial K_y}{\partial\theta_j}\right) \text{ where } \boldsymbol{\alpha} = K_y^{-1}\mathbf{y} \tag{4}$$

The gradient-based optimization methods use this derivative, with the computational complexity of $\mathcal{O}(n^3)$, to obtain the hyper-parameters that maximize the log marginal likelihood.

2.3 Optimization methods

Several optimization methods for hyper-parameter estimation are compared: (i) Gradient Descent (GD), (ii) GD with Nesterov acceleration, (iii) Newton's, (iv) Limited-memory BFGS (LBFGS), (v) Fisher Scoring, and (vi) Nelder-Mead.

The methods except Nelder-Mead are gradient-based optimization methods. Gradient descent finds the local optimum of the objective function by taking steps proportional to the negative gradient, and Nesterov Accelerated Gradient Descent (Nesterov (2004)), also known as Nesterov Momentum, is a faster modification based on ordinary gradient descent.

Newton's method calculates the exact second derivatives of log marginal likelihood in Eq.(3). BFGS approximates the iterative direction of Newton's method and avoids calculating the second derivatives, and based on this, LBFGS uses a limited amount of computer memory (Nocedal and Wright (2006)). Fisher Scoring method also takes the form of Newton's, but uses an expected information matrix called Fisher information to replace the second derivative matrix in Newton's. Nelder-Mead uses a heuristic search strategy that requires a number of evaluations of function values per iteration.

Chapter 3

Experiments

The experiments are performed using the `GPBoost` library, which includes the GP model fitting implementation and provides all the optimization methods mentioned in Section 2.3. We will compare the performance of different optimization methods in the following situations: (i) data simulated from GP models and fitted with a correctly specified GP model (with the same smoothness), (ii) data simulated from GP models and fitted with a mis-specified GP model (different smoothness), and (iii) data generated from deterministic functions (Branin, Borehole, Piston). The simulation is conducted on a 10-core M2 pro processor with 16GB RAM.

In the optimization stage, the hyper-parameters, range, signal variance and noise variance $(\rho, \sigma_f^2, \sigma_n^2)$ are estimated. Covariance functions with different smoothness values have different effective range distances, which represent the distance between data points that are correlated. By calculating the range value when the correlation takes the value 0.05, different initial values of the range ρ will be given. The default initialization calculation of the hyper-parameters under each smoothness value is shown in Table 3.1.

ν	0.5	1.5	2.5	inf
ρ_{init}	$\bar{d}/3$	$\sqrt{3} \times \bar{d}/4.7$	$\sqrt{5} \times \bar{d}/5.9$	$\bar{d}/\sqrt{3}$
$\sigma^2_{f\,init}$	$\text{var}(y)/2$	$\text{var}(y)/2$	$\text{var}(y)/2$	$\text{var}(y)/2$
$\sigma^2_{n\,init}$	$\text{var}(y)/2$	$\text{var}(y)/2$	$\text{var}(y)/2$	$\text{var}(y)/2$

Table 3.1: Default initialization calculation of hyper-parameters

The computational time and space complexity of the derivative in Eq.(4) are respectively $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$, which becomes burdensome for large data. To reduce the computational cost, we enable the FITC (Fully Independent Training Conditional) (Quinonero-Candela and Rasmussen (2005)) and Vecchia (Nesterov (2004)) approximations for large data ($n > 2000$). The approximations will provide an alternative and approximate way to compute the likelihood and its gradient in different ways, also implemented in `GPBoost` (Sigrist (2022)).

We run 10 repeated rounds using each optimization method discussed in Section 2.3 for each setting of experiments. To check whether an optimization process successfully converges, we use the median of final log-likelihood values of the six optimization methods as each repetition's criterion and the relative tolerance is set to 0.001.

In the results, we will report the average, minimum and maximum times of 10 repetitions for each experiment, which are represented by, respectively, the center point, lower bar and upper bar in the time figures, and similarly in the number of iterations figures.

3.1 GP models with Matérn covariance

3.1.1 Zero-mean GP models

In this section, we uniformly generate samples of n independent points from the spatial domain $\mathcal{D} = [(0, 1)]^2$, and simulate observations Y from a GP model with zero mean and Matérn class covariance function with smoothness $\nu \in (0.5, 1.5, 2.5, \text{Inf})$.

The experiment settings are designed such that, in each setting of experiments, one of the variables $(\rho, \sigma_f^2/\sigma_n^2, n, \nu)$ is varied based on the control setting, which is $(0.1, 2, 1.5, 500)$. Note that the signal variance σ_f^2 is always fixed at 1, which means changing σ_n^2 controls the signal-to-noise ratio. The variables vary in the set $\rho \in (0, 0.01, 0.05, 0.1, 0.5, \sqrt{2})$, $\sigma_f^2/\sigma_n^2 \in (0.1, 0.5, 1, 2, 5, 10)$, $n \in (200, 500, 1000, 2000, 5000, 20000)$, and $\nu \in (0.5, 1.5, 2.5, \text{Inf})$.

In the model fitting stage, we use a GP model with a Matérn class covariance function with smoothness ν set to the true value. This is the case when the GP covariance function used for fitting belongs to the same parametric class as the model for simulation. The results of zero-mean experiments are displayed in Figure 3.1. Moreover, Figure 3.2 shows the number of iterations required to converge in the same experiments.

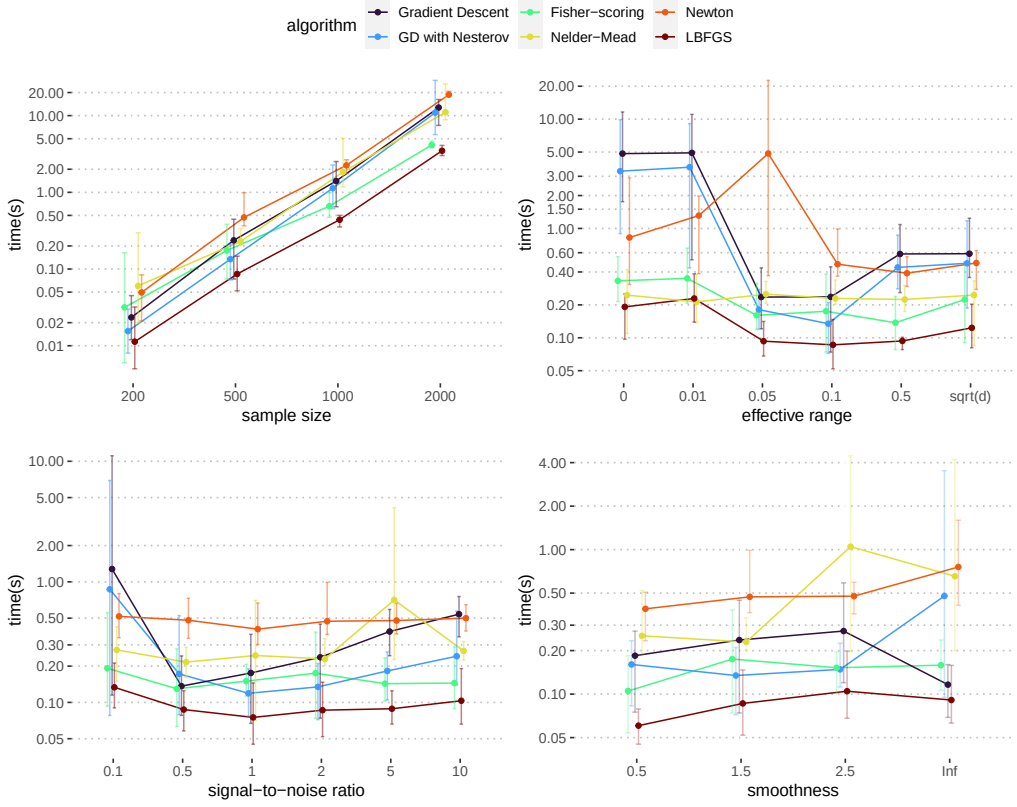

Figure 3.1: Comparison of optimization methods with different sample sizes and hyper-parameters $(n, \rho, \sigma_f^2/\sigma_n^2, \nu)$, all y-axes in $\log_{10}$ scale.

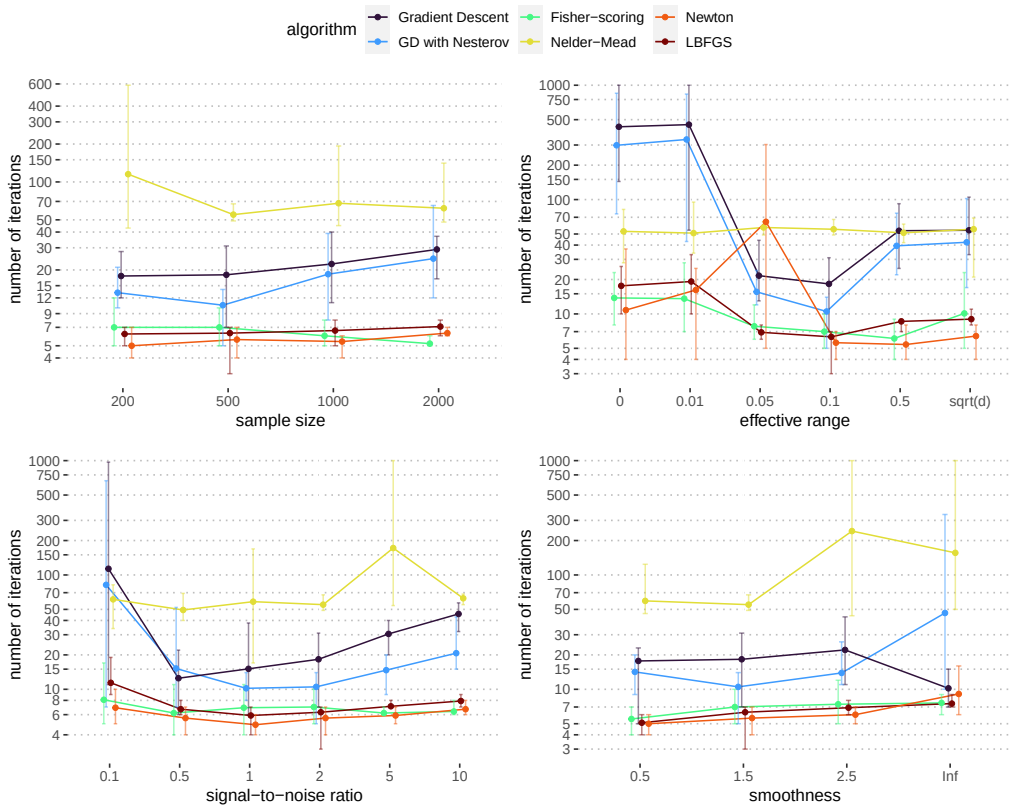

Figure 3.2: Comparison of number of iterations of different methods, all y-axes are in $\log_{10}$ scale.

Figure 3.1 illustrates that LBFGS is the fastest method in most settings, and Fisher-scoring also has good performance. In most settings, Newton is the slowest method due to the need to compute the exact second derivative.

Figure 3.2 shows that an iteration of Newton is more expensive, whereas an iteration of Nelder-Mead algorithm takes the least time. In each iteration, Newton calculates the exact second order derivative, and Nelder-Mead only evaluates the function values. Also, LBFGS, Newton, and Fisher-scoring methods mostly converge within 10 steps of iteration, and tend to require more steps in the case of small range parameter ρ.

We check whether these optimization methods converge by checking the final log-likelihood values, and there are the following convergence problems:

- When $\rho = 0$ (the simulated data is actually i.i.d Gaussian distributed), Nelder-Mead fails to converge in 2 (of 10) repetitions.
- When $\rho = 0.01$, Nelder-Mead fails in 3 (of 10) repetitions, Gradient Descent and GD with Nesterov accelerator both fail to converge in 2 (of 10) repetitions, Newton and LBFGS both fail in 1 (of 10) repetition.
- When $\sigma_f^2/\sigma_n^2 = 0.1$ ($\sigma_n^2 = 10$), Nelder-Mead fails to converge in 1 (of 10) repetition.
- When $\sigma_f^2/\sigma_n^2 = 5$ ($\sigma_n^2 = 0.2$), Nelder-Mead fails to converge in 1 (of 10) repetition.
- When $\nu = 2.5$, Nelder-Mead fails to converge in 2 (of 10) repetitions.

- When $\nu = \text{inf}$, Nelder-Mead fails to converge in 1 (of 10) repetition.

We perform experiments with larger sample size with FITC and Vecchia approximations separately, resulting in four sets of experiment results in Figure 3.3. All repetitions of experiments result in successful convergence.

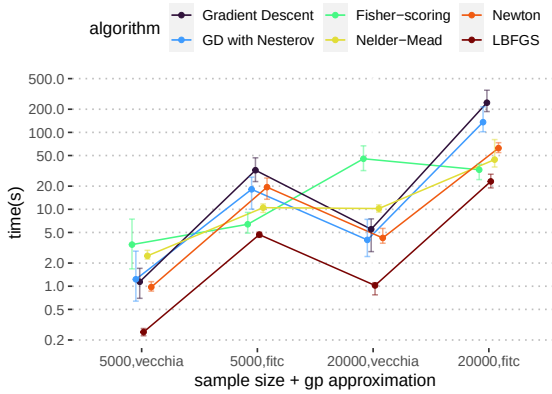

Figure 3.3: Comparison of times of different methods with approximation enabled, y-axis in $\log_{10}$ scale.

Figure 3.3 shows that LBFGS still outperforms the other methods with approximation enabled, and using the Vecchia approximation for GP models seems to be more efficient on average than FITC. LBFGS has the most stable performance, which can be deduced from the shorter vertical distances between bars. However, the approximate calculation of Fisher information with Vecchia in the implementation of `GPBoost` makes it slower than the other optimization methods, and in this sense, it is an unfair comparison between Fisher Scoring and other optimization methods in the presence of Vecchia approximation.

Different initialization

To investigate the influence of the starting points of the hyper-parameters on the optimization, we repeat the experiments in Figure 3.1, but with the starting values as true values and with inappropriately large values, which corresponds to the results in Figure 3.4 and Figure 3.5 respectively.

Comparing Figures 3.4 and 3.5 with Figure 3.1, we can conclude that LBFGS is the fastest method in most experiments with any initialization strategy. Newton is sensitive to the choice of starting point, reflected by the phenomenon that starting from points far from the optima makes it difficult to converge within the iteration limit. However, if a suitable starting point is chosen, Fisher-scoring or Nelder-Mead can be as fast as LBFGS.

When starting the iteration from true values, all optimization methods successfully converge to the optima and consume less time compared to the other two initialization strategies. It is intuitive that the time of any optimizer will be longer if it starts iterating from an unreasonably distant point instead of a reasonable point related to the sample. Also, the convergence problem with unreasonable starting points in Figure 3.5 is more severe than with the default initialization:

- In the default setting ($n = 500, \rho = 0.1, \sigma_f^2/\sigma_n^2 = 2$, and $\nu = 1.5$), Newton fails to converge in 2 (of 10) repetitions.

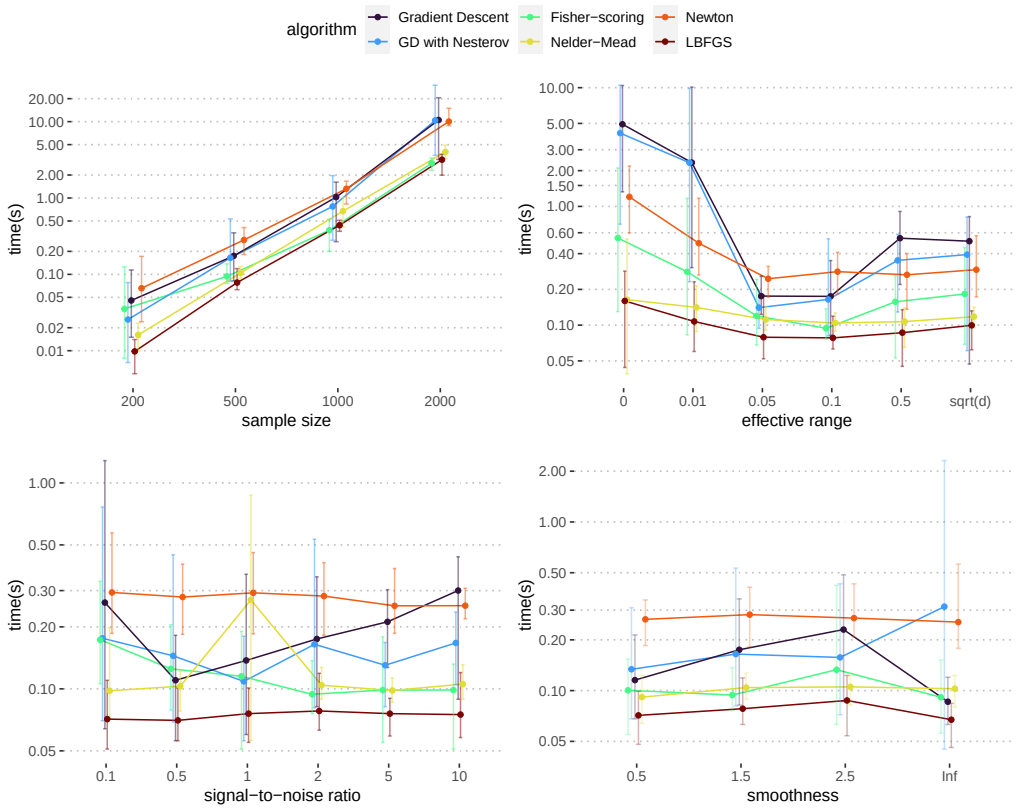

Figure 3.4: Comparison of different optimization methods with different sample sizes and hyper-parameters, starting at true values, all y-axes in $\log_{10}$ scale.

- When $n = 1000$, Newton fails to converge in 4 (of 10) repetitions.
- When $n = 2000$, Newton fails to converge in 3 (of 10) repetitions.
- When $\rho = 0$, Nelder-Mead and LBFGS fails both in 1 (of 10) repetition.
- When $\rho = 0.01$, GD, GD with Nesterov accelerator and LBFGS all fails in 1 (of 10) repetition.
- When $\rho = 0.05$, Newton fails to converge in 7 (of 10) repetitions.
- When $\rho = 0.5$, Newton fails to converge in 1 (of 10) repetition.
- When $\sigma_f^2/\sigma_n^2 = 0.1$, Nelder-Mead and Newton both fail in 2 (of 10) repetitions.
- When $\sigma_f^2/\sigma_n^2 = 1$, Newton fails to converge in 2 (of 10) repetitions.
- When $\sigma_f^2/\sigma_n^2 = 5$, Newton fails to converge in 1 (of 10) repetition.
- When $\sigma_f^2/\sigma_n^2 = 5$, Newton fails to converge in 3 (of 10) repetitions.
- When $\nu = 0.5$, Newton fails in 5 (of 10) repetitions.
- When $\nu = 2.5$, Newton fails in 5 (of 10) repetitions.

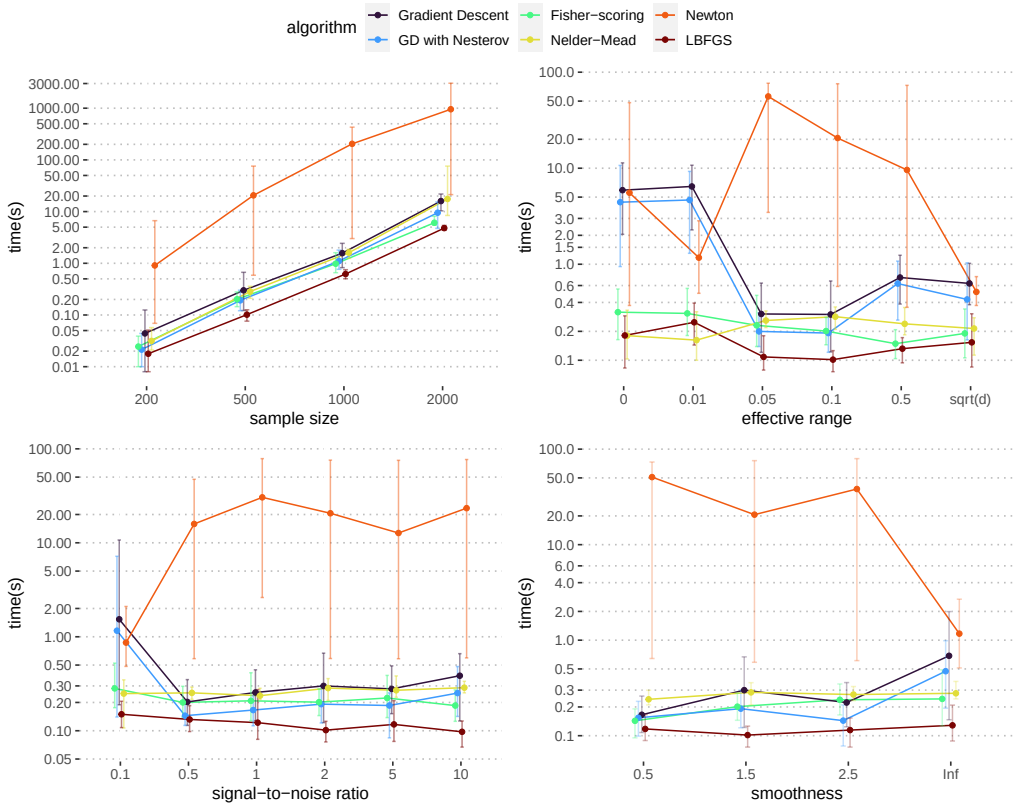

Figure 3.5: Comparison of different algorithms with different sample sizes and hyper-parameters, starting at large values, 10 repetitions. All y-axes are in log 10 scale.

BFGS-based methods

Considering the good performance of LBFGS, we further compare five variants of BFGS methods available in R. Two approximations are still enabled for large data. Besides "lbfgs" in `GPBoost`, which we have experimented with, we also introduce "lbfgs_not_profile_out_nugget" from `GPBoost`, and BFGS, L-BFGS-B and L-BFGS (L-BFGS-B with no bound on hyper-parameters) from `optim` function.

The results are displayed in Figures 3.6 and 3.7, which show that "lbfgs" is faster than "lbfgs_not_profile_out_nugget", and they both significantly beat the three optimization methods in `optim`. In addition, some methods have convergence problems:

- When $\rho = 0$, L-BFGS-B(optim) fails to converge in 1 (of 10) repetition.
- When $\rho = 0.01$, L-BFGS-B(optim) fails to converge in 3 (of 10) repetitions, "lbfgs_not_profile_out_nug
  fails to converge in 2 (of 10) repetitions.
- When $\rho = \sqrt{2}$, BFGS(optim) fails to converge in 1 (of 10) repetition.
- When $\sigma_f^2/\sigma_n^2 = 0.1$, L-BFGS-B(optim) fails to converge in 2 (of 10) repetitions.
- When $n = 20000$ and using FITC approximation, BFGS(optim) fails to converge in 2 (of 10) repetitions.

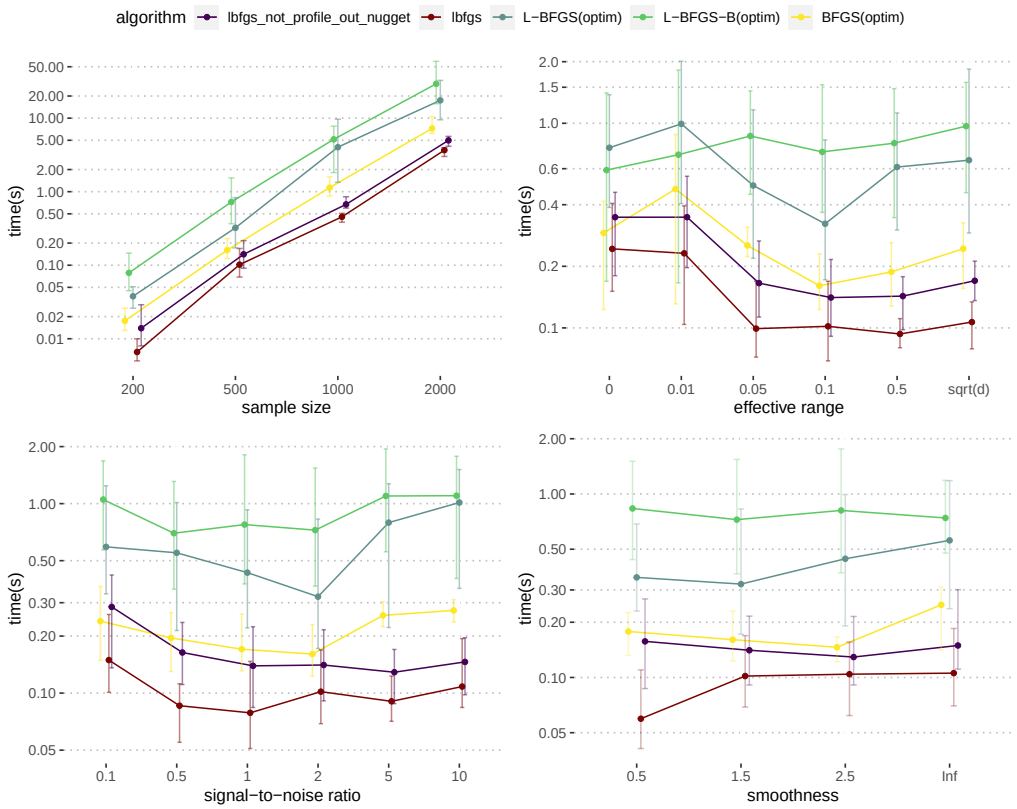

Figure 3.6: Comparison of BFGS-based algorithms with different sample sizes and hyper-parameters, starting at default values, all Y-axes are in $\log_{10}$ scale.

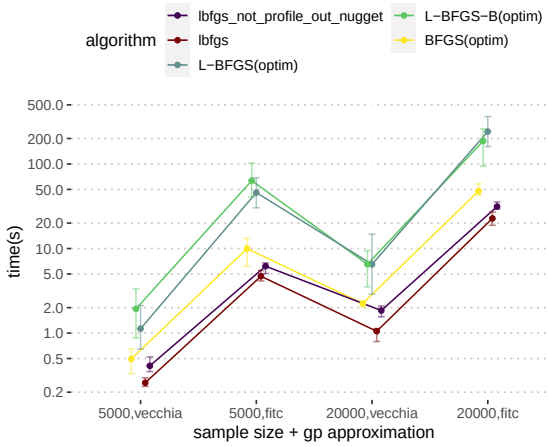

Figure 3.7: Comparison of times of BFGS methods with approximation on zero-mean GP models, y-axis in $\log_{10}$ scale.

3.1.2 GP with Non-zero mean

To further investigate the performance of the optimization methods on GP models, we add the linear regression term $\boldsymbol{Z\beta}$ as the mean function $m(\mathbf{x})$:

$$\boldsymbol{z} = [1, z_1, ..., z_9],\ \boldsymbol{\beta} = [0, 1, ..., 1], \text{ where } z_i \overset{\text{iid}}{\sim} N(0, 0.1). \tag{5}$$

Now we need to estimate the hyper-parameters $(\rho, \sigma_f^2, \sigma_n^2)$ and the linear coefficients $\boldsymbol{\beta}$. The experimental settings of hyper-parameters in section 3.1.1 are applied again. The experimental results are shown in Figure 3.8, and the large scale optimization results are shown in Figure 3.9. Note that Nelder-Mead method is excluded because it is unable to converge to the optima in any setting, and it takes a long time to search in each step. The heuristic search strategy of Nelder-Mead makes it difficult to solve the problem when the number of parameters to be estimated increases enormously.

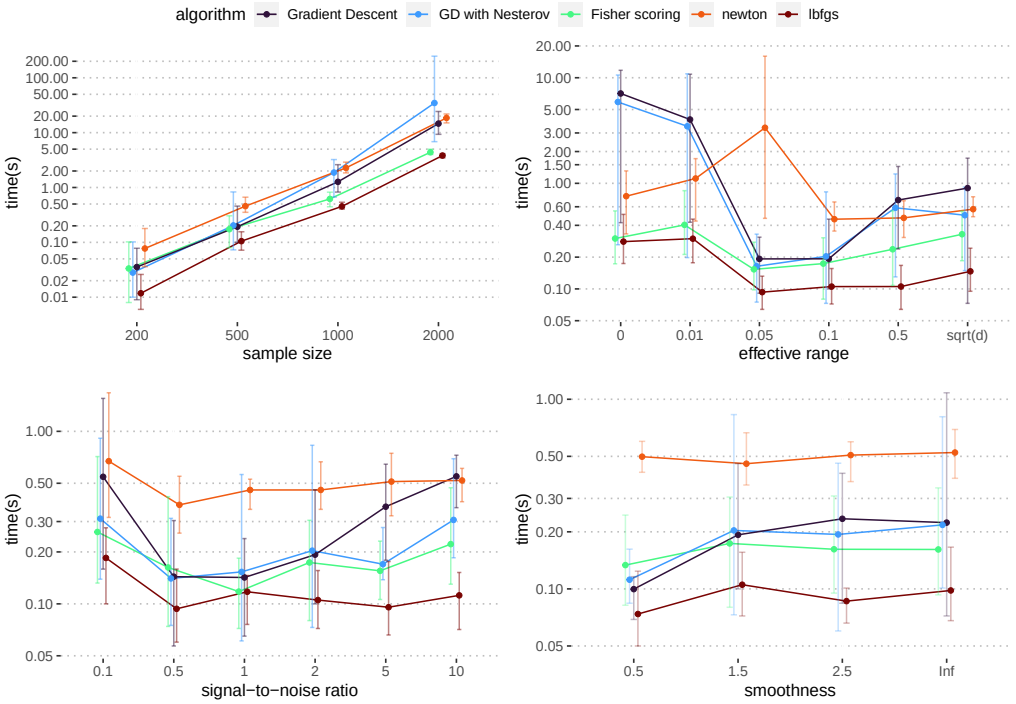

Figure 3.8: Comparison of different methods of GP models with added linear regression term, default starting values, all y-axes are in $\log_{10}$ scale.

Compared to the results of the zero-mean GP models in Figure 3.1, Figure 3.8 shows that the relative ranking among the optimization methods and the average time used by each method remain the same after introducing the linear regression term, and LBFGS still has outstanding performance. Results in Figure 3.9 show that LBFGS and Newton take the least time in large scale optimization.

The convergence check shows that:

- When $\rho = 0.01$, GD, GD with Nesterov accelerator and Newton all fail to converge in 1 (of 10) repetition.
- When $n = 5000$ and using Vecchia approximation, Newton's method fails to converge in 1 (of 10) repetition.

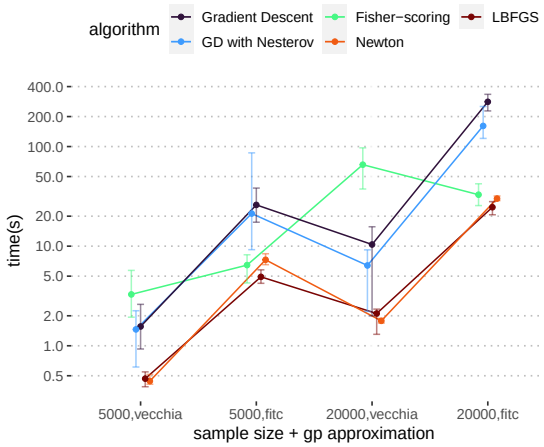

Figure 3.9: Comparison of times of different methods with approximation on model with linear terms, y-axis in $\log_{10}$ scale.

3.1.3 Binary data

The observation variables are categorical in some applications, where GP classification models are applied. Therefore, we test the performance of optimization methods on GP models with binary likelihood (Bernoulli distribution with *logit* link function). In this section, similar settings of hyper-parameters are used, whereas the noise term is no longer included in the data simulation. Instead of the signal-to-noise ratio, we only change the signal variance in the experiment. The results are displayed in Figurea 3.10 and 3.11, note that Fisher-scoring is excluded in the results, because the Fisher information is not computable for GP with binary likelihood. The convergence check shows that, when $n = 5000$ and using FITC approximation, Newton fails to converge in 1 (of 10) repetition.

Furthermore, we again add the linear regression terms as in Eq.(5) into the GP classification models. The performance comparison of four algorithms (GD, GD with Nesterov accelerator, Nelder Mead, and LBFGS) is displayed in Figure 3.12 and Figure 3.13. In these experiments, Newton's method faces some convergence problems:

- When $\rho = 0.01$, Newton fails to converge in 1 (of 10) repetition.
- When $\sqrt{d}$, Newton fails to converge in 1 (of 10) repetition.
- When n=20000 and using FITC approximation, Newton fails to converge in 2 (of 10) repetitions.

From Figures 3.10, 3.11, 3.12 and 3.13 in this section, similar conclusions can be drawn; LBFGS is still excellent, followed by Nelder-Mead method and two GD methods, and Newton's method is the slowest one in most cases and shows instability, which is reflected by the large gap between maximum and minimum times. Gradient Descent methods are limited by the long-ridge and flat likelihood, which is illustrated by the slower optimization in the models with small range.

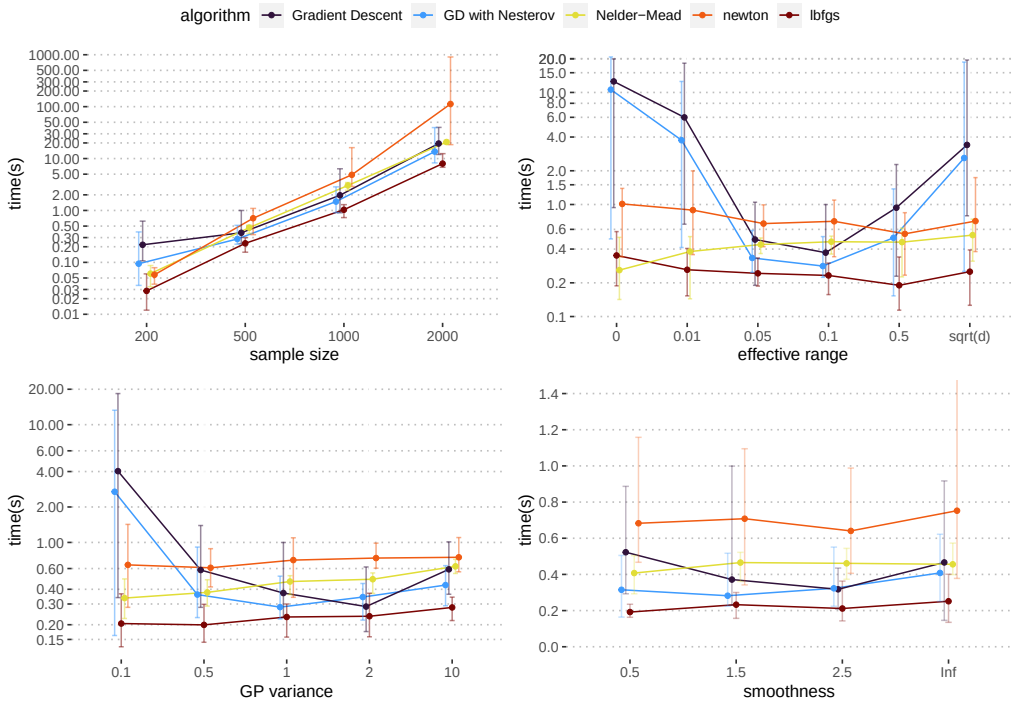

Figure 3.10: Comparison of methods on GP with binary likelihood of different sample sizes and hyper-parameters, all y-axes except lower-right sub-figure are in $\log_{10}$ scale.

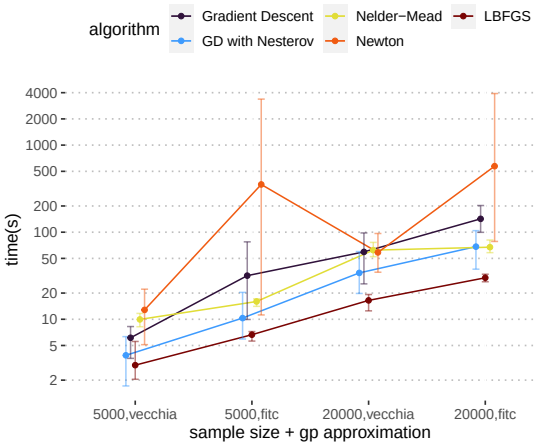

Figure 3.11: Comparison of times of different methods with approximation on model with binary likelihood, y-axis in $\log_{10}$ scale.

3.2 Mis-specified models

In Section 3.1, we have performed several experiments based on the GP models with Matérn class covariance function, when we have used correctly specified models in the model fitting stage. In this section, performance of optimization methods in the situation of model mis-specification will be compared, which means that the GP model used for fitting belongs to a different distribution class than the model used for data simulation. In our experiments, the data sample is simulated from GP with Mat'ern covariance function

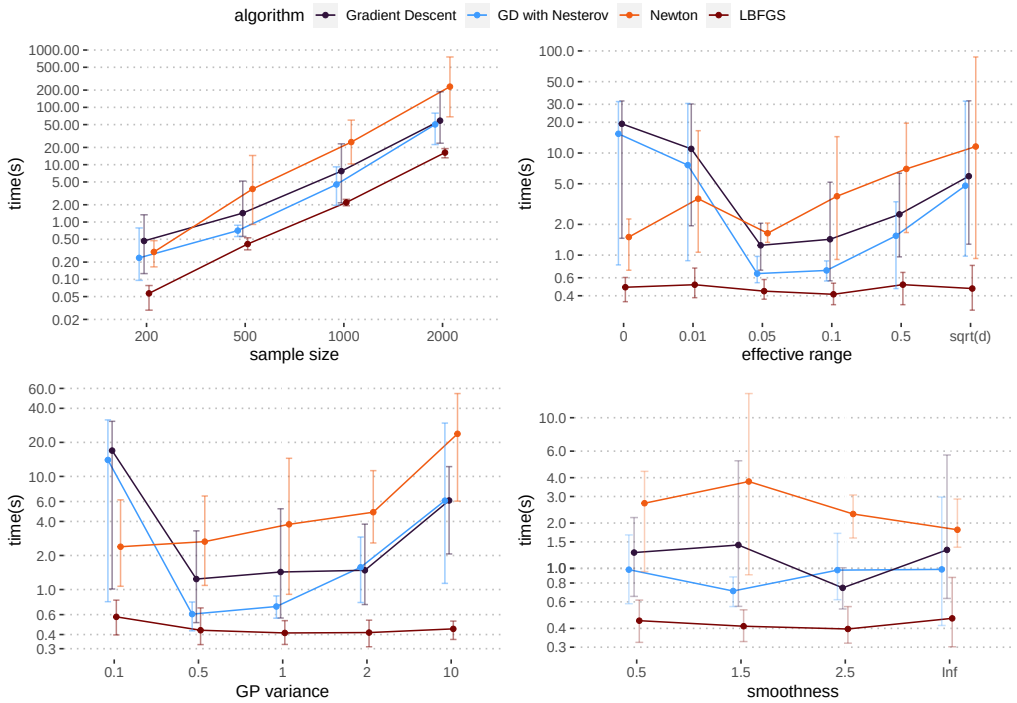

Figure 3.12: Comparison of different methods on GP with binary likelihood and added linear regression terms, all y-axes are in $\log_{10}$ scale.

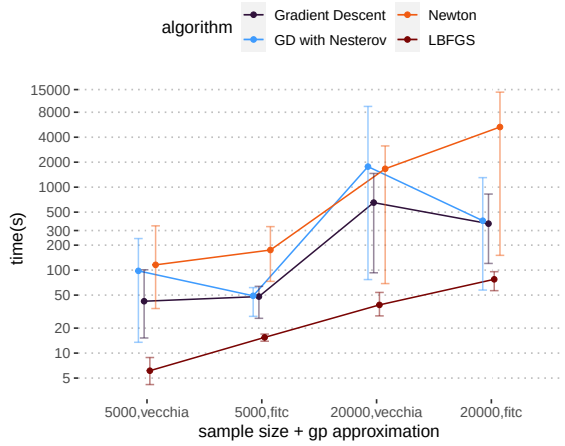

Figure 3.13: Comparison of times of different methods with approximation on model with binary likelihood and linear regression terms, y-axis in $\log_{10}$ scale.

with smoothness $\nu = 0.5/\inf$, and then optimized using the GP model with different smoothness value.

Again, we consider the application of a smaller sample and a large sample of size 5000 (using Vecchia and FITC) and fit a GP model with different smoothness values. The results of data simulated by GP models with smoothness $\nu = 0.5$ (exponential covariance) are shown in Figure 3.14. Figure 3.15 displays the results of smoothness $\nu \to \infty$ (squared exponential or Gaussian covariance). The figures are divided into two parts: left for the experiments with smaller sample size, right for the large sample with Vecchia and FITC

approximation.

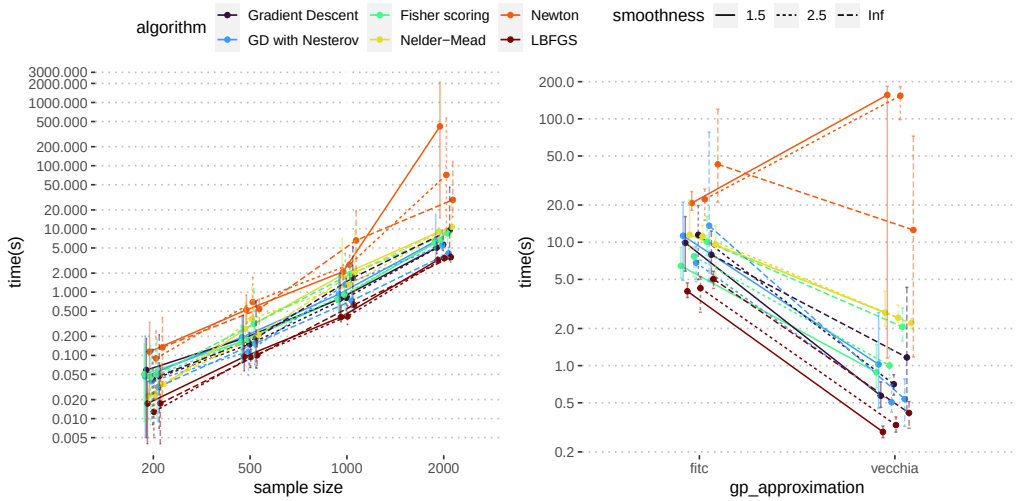

Figure 3.14: Comparison of different methods using mis-specified GP models on data from GP with 0.5 smoothness, default starting values, 10 repetitions. Y-axis is in log 10 scale. Vecchia approximation is utilized when n = 5000

Figure 3.14 shows that LBFGS has the best performance with any smoothness parameters, and Newton is the slowest. The convergence check of this figure shows that for large data of 5000 points:

- When $\nu = 1.5$ and using Vecchia approximation, Newton fails to converge in 7 (of 10) repetitions.
- When $\nu = 2.5$ and using Vecchia approximation, Newton fails to converge in 5 (of 10) repetitions.

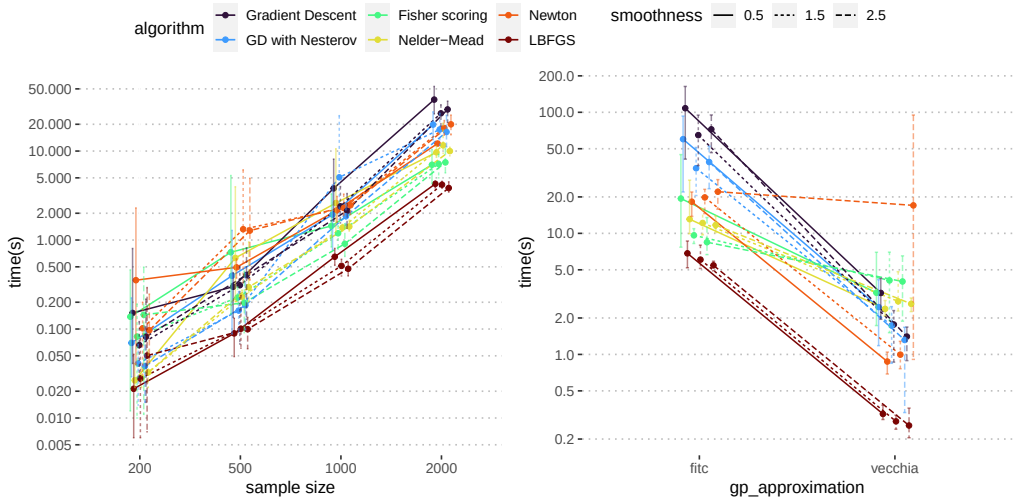

Figure 3.15: Comparison of different methods using mis-specified GP models on data from GP with infinite smoothness, default starting values, 10 repetitions. Y-axis is in log 10 scale. Vecchia approximation is utilized when n = 5000.

bFor experiments of data from GP model of Gaussian covariance (infinite smoothness

value), the convergence check shows that:

- When $\nu = 0.5, n = 200$, Newton fails to converge in 6 (of 10) repetitions.
- When $\nu = 0.5, n = 500$, Nelder-Mead fails to converge in 1 (of 10) repetition.

3.3 Deterministic functions

In this section, we introduce three deterministic functions for simulation experiments: Branin, Borehole and Piston (Surjanovic and Bingham (Surjanovic and Bingham)), and compare the performance of the optimization methods. We sample data from the functions and estimate the hyper-parameters of the Matérn covariance function. This is the case when not enough information is known about the underlying function and the fitting model does not belong to the same distribution family as the true model. Note that in all deterministic experiments, the data is centered first in the pre-processing. We will first experiment with data of size 1000, and in section 3.3.4 we will use the Vecchia approximation for experiments on large data of size 20000.

3.3.1 Branin function

Branin function is a 2-dimensional function that usually evaluated over the domain $[-5, 10] \times [0, 15]$. It is defined as:

$$f(\boldsymbol{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos x_1 + s \tag{6}$$

and we take the recommended value as $a = 1, b = 5.1/4\pi^2, c = 5/\pi, r = 6, s = 10$, and $t = 1/8\pi$ from Surjanovic and Bingham (Surjanovic and Bingham).

Experiment results in Figure 3.16 illustrate that, even with Nesterov accelerator, Gradient Descent algorithm needs significantly more time to converge, especially when smoothness is 1.5 or 2.5. LBFGS remains the fastest in all cases. The convergence check of Branin experiment shows that, when $\nu \to \infty$, Newton's method fails to converge in 3 (of 10) repetitions.

3.3.2 Borehole function

Borehole function models the flow of water through a borehole and it consists of eight input variables. Similar conclusions could be drawn from the results in Figure 3.17, except that Newton's method takes the longest average time with a large range when smoothness is 0.5.

The convergence check shows that, when $\nu = 0.5$, Newton algorithm fails to converge in 2 (of 10) repetitions. In these cases, Newton's method fails to find the optima within the prescribed of 1000 iterations limit.

3.3.3 Piston function

The 7-dimensional Piston function, which models the circular motion of a piston within a cylinder, is sampled with 1000 uniformly distributed points, and the results are displayed in Figure 3.18. The convergence check shows that when $\nu = 0.5$, Newton's method fails to converge in 2 (of 10) repetitions.

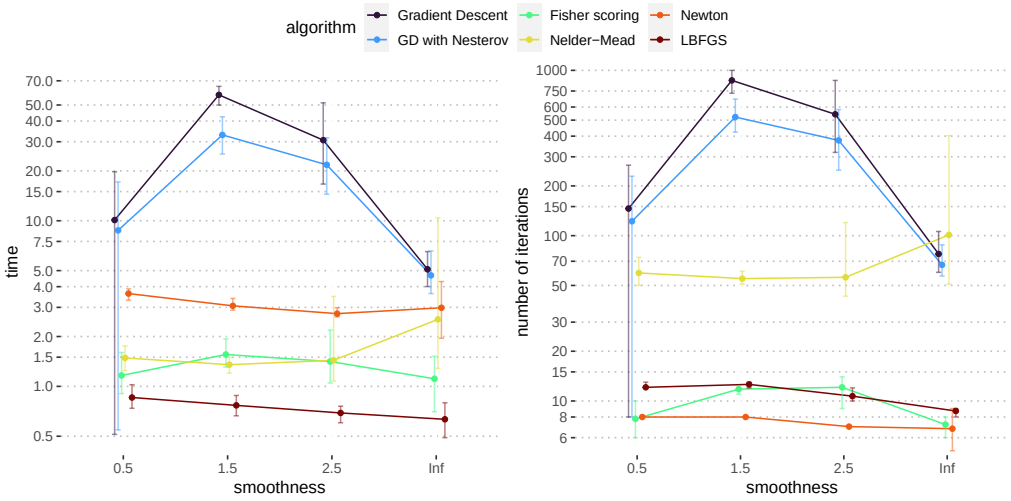

Figure 3.16: Comparison of different methods using mis-specified GP models on data from Branin function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

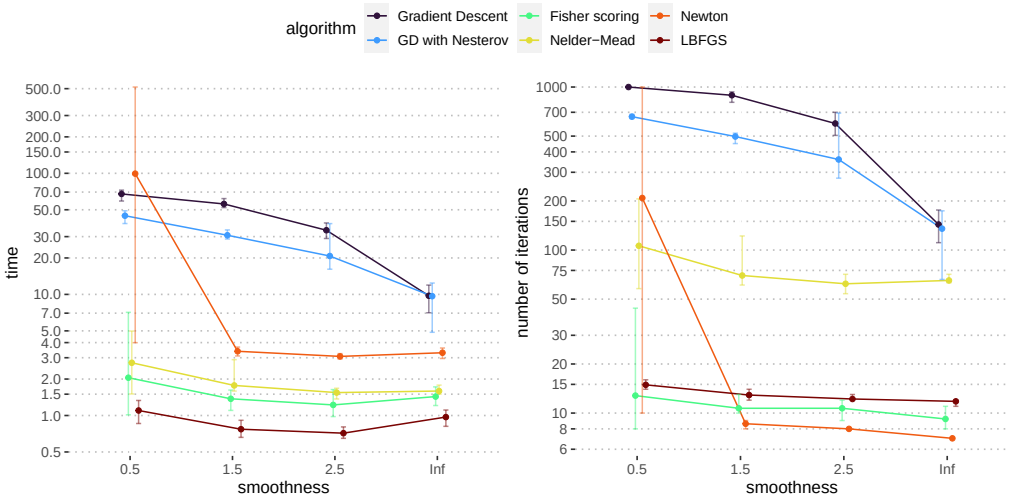

Figure 3.17: Comparison of different methods using mis-specified GP models on data from Borehole function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

3.3.4 Large scale optimization

We want to explore the performance of optimization methods on the large scale of deterministic functions. We generate random samples of 20000 points and fit GP-Matérn models with Vecchia approximation. Again, we perform the convergence check, but with stricter relative tolerance of 0.0001, since the original tolerance would not be precise enough due to the large log-likelihood values of this sample size.

The results of Branin, Borehole, and Piston functions are displayed in Figures 3.19, 3.20 and 3.21. Among the six optimization methods, LBFGS consistently shows excellent performance of optimization for all of the three deterministic functions.

The convergence problems of Branin large data experiment are:

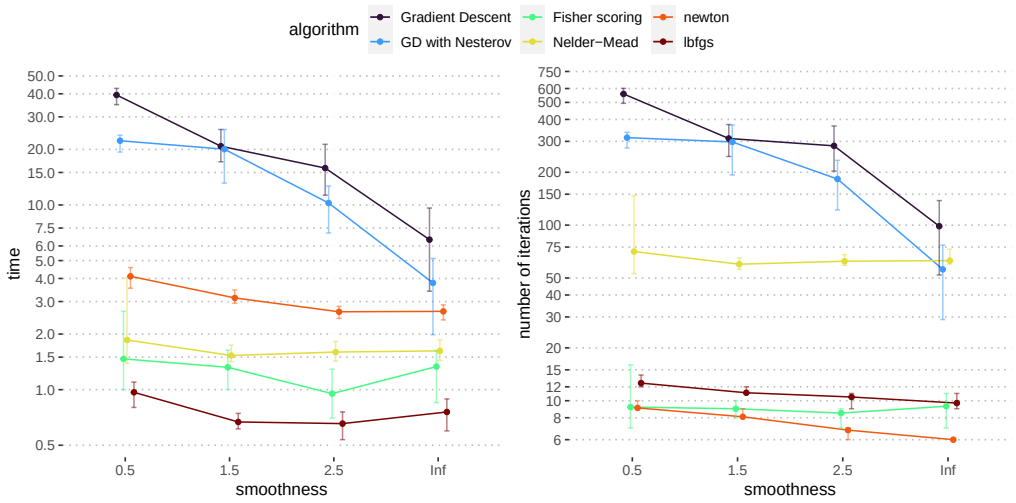

Figure 3.18: Comparison of different methods using mis-specified GP models on data from Piston function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

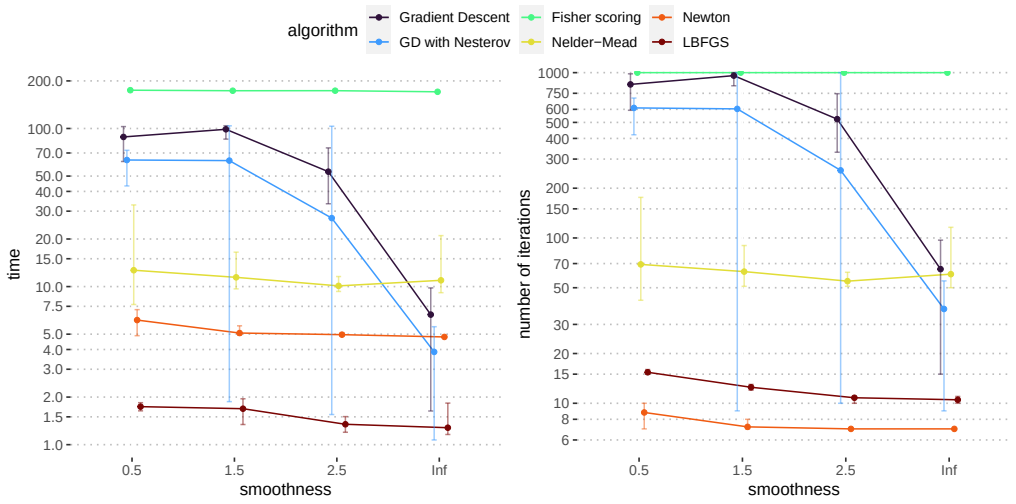

Figure 3.19: Comparison of different methods using GP models on 20000 points data from Branin function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

- With any smoothness, Fisher-scoring fails to converge within prescribed 1000 steps in any repetition.
- When $\nu = 1.5$, GD with Nesterov accelerator fails in 5 (of 10) repetitions.

For experiment of large data from Borehole function, the convergence check shows that:

- With any smoothness, Fisher-scoring fails to converge within prescribed 1000 steps in any repetition of all smoothness settings.
- When $\nu = 1.5$, Newton fails in 2 (of 10) repetitions.
- When $\nu = 2.5$, Newton fails in 8 (of 10) repetitions.
- When $\nu = \text{inf}$, Newton fails in 1 (of 10) repetition.

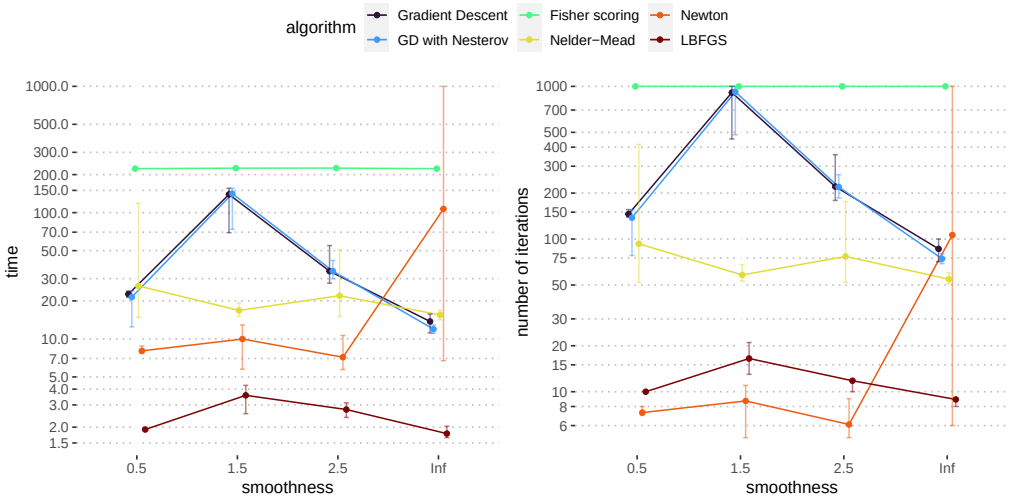

Figure 3.20: Comparison of different methods using GP models on 20000 points data from Borehole function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

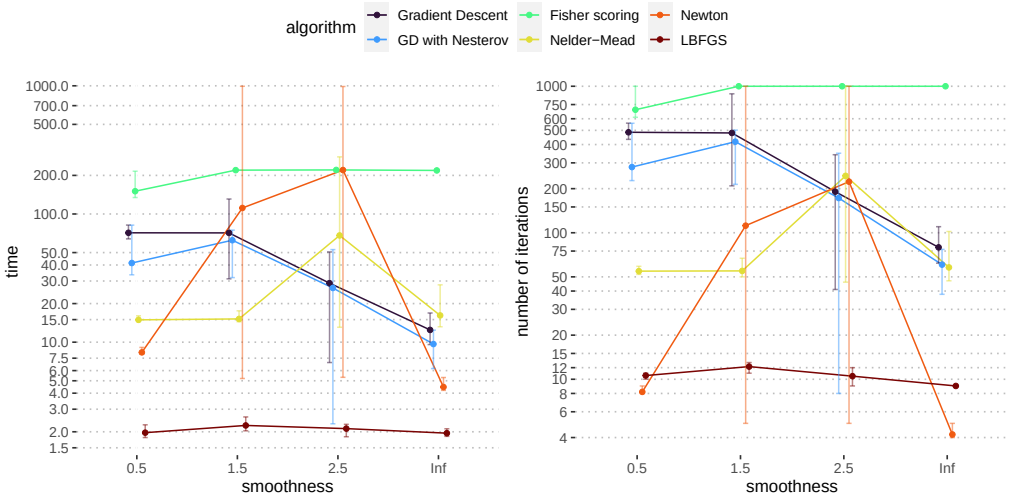

Figure 3.21: Comparison of different methods using GP models on 20000 points data from Piston function, default starting values, 10 repetitions. Y-axes are in log 10 scale.

For experiment of large data from Piston function, the convergence check shows that:

- When $\nu = 0.5$, Fisher-scoring fails in 5 (of 10) repetitions.
- When $\nu = 1.5$, 2.5, or ∞, Fisher-scoring fails to converge within prescribed 1000 iterations in all repetitions.
- When $\nu = 1.5$, Newton fails in 8 (of 10) repetitions.
- When $\nu = 2.5$, Newton fails in 10 (of 10) repetitions, Nelder-Mead fails in 2 (of 10) repetitions.

Chapter 4

Conclusions

In this thesis, we have explored the performance of different optimization methods for Gaussian processes hyper-parameter estimation based on `GPBoost` library, taking into account the training time, for both probabilistic and deterministic functions, and in both regression and classification models. We also include two state-of-the-art approximations (Vecchia, FITC) for GP models to reduce the computational time and space complexity as large as $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ in large scale optimization.

As can be seen from the results, LBFGS shows an excellent and more stable performance in terms of both training speed and convergence robustness compared to the other methods. Gradient Descent shows robustness in case of mis-specification, but still risks a slower training speed and is susceptible to the shape of the log marginal likelihood. GD, Newton and Fisher-scoring can get stuck in local optima and struggle with non-convex objective functions, making them excruciatingly slow on some difficult problems. Nelder-Mead, the derivative-free method, can perform in some circumstance but unable to cope with the increasing numbers of parameters and has a higher computational cost.

The experimental results also show that many factors, such as initialization strategy and reparameterization, affect the convergence time of the optimization methods. By choosing a reasonable starting point, the optimization methods, especially Newton's method, could perform significantly better. The *log* reparameterization of the hyper-parameters $(\rho, \sigma_f^2, \sigma_n^2)$ with positive constraints could facilitate the convergence, which is also suggested in Basak, Petit, Bect, and Vazquez (2021).

The choice of optimization method depends on the specific context and requirements of the application. Different optimization methods have their own advantages depending on the nature of data and the objective function. Without knowing much information about the underlying distribution, LBFGS can be a better and versatile choice for various optimization landscapes, but combining the characteristics of dataset and the computational environment is still necessary. Future research could focus on including more types of simulation data and investigating the real world dataset.

Bibliography

Basak, S., S. Petit, J. Bect, and E. Vazquez (2021). Numerical issues in maximum likelihood parameter estimation for gaussian process interpolation. In *International Conference on Machine Learning, Optimization, and Data Science*, pp. 116–131. Springer.

Eidsvik, J. (2011). Spatial statistics, addition to part i. parameter estimation and kriging for gaussian random fields.

Kennedy, M. C. and A. O'Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63*(3), 425–464.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, Volume 87. Springer Science & Business Media.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization* (2e ed.). New York, NY, USA: Springer.

Quinonero-Candela, J. and C. E. Rasmussen (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research 6*, 1939–1959.

Sigrist, F. (2022). Gaussian process boosting. *The Journal of Machine Learning Research 23*(1), 10565–10610.

Surjanovic, S. and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved June 10, 2024, from http://www.sfu.ca/~ssurjano.

Williams, C. K. and C. E. Rasmussen (2006). *Gaussian processes for machine learning*, Volume 2. MIT press Cambridge, MA.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

- (●) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].
- ( ) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].
- ( ) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Optimization methods for Gaussian process hyper-parameter estimation

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	First name(s):
Nian	Yiting

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	Signature(s)
Zurich 13.06.2024	Yiting Nian.

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard