The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Robust Optimisation Monte Carlo for Likelihood-Free Inference

by

Vasileios Gkolemis

Dissertation Presented for the Degree of
MSc in Operational Research with Data Science

August 2020

Supervised by
Dr. Michael Gutmann

Abstract

Here comes your abstract ...

Acknowledgments

Here come your acknowledgments ...

Own Work Declaration

Here comes your own work declaration

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Explanation of Simulation-Based Models

A Simulator-Based model is a parameterized stochastic data generating mechanism [2]. The key characteristic is that although we are able to sample (simulate) data points, we cannot evaluate the likelihood of a specific set of observations y_0. Formally, a simulator-based model is described as a parameterized family of probability density functions $\{p_{y|\theta}(y)\}_\theta$, whose closed-form is either unknown or intractable to evaluate. Although, evaluating $p_{y|\theta}(y)$ is intractable, sampling is feasible and frequently without huge computational cost. Practically, if we set as V the vector containing the (unobserved) random state of the process, then as a mapping $M(\theta, V) \rightarrow y$

The level of modelling freedom make implicit models particularly captivating; any physical process that can be conceptualized as a computer program of finite (determinstic or stochastic) steps, can be modelled as a Simulator-Based model without any mathematical compromise. This includes any amount of hidden (unobserved) internal variables. On the other hand, this level of freedom comes at a cost; performing inference is particularly demanding from a compuational and mathematical perspective. This constraints the dimensionality of $\theta \in \mathbb{R}^D$ to quite low levels (i.e. $D < 20$).

Example

For underlying the importance of Simulator-Based models, lets use as example the tuberculosis disease spread model as described in [7]. At each stage we can observe the following events; (a) the transmission of a specific haplotype to a new host (b) the mutation to a different haplotype (c) the exclusion of an infectius host (recovers/dies). The random process, which stops when m infectius hosts are reached, can be parameterized; (a) the transmission rate α (b) the mutation rate τ and (c) the exclusion rate δ. The outcome of the process is a variable-sized tuple y_θ, containg the size of all different infection groups, as described in figure 1. Computing $p(y = y_0|\theta)$ requires tracking all tree-paths that generate the specific tuple along with their probabilities and summing over them. Computing this probability becomes intractable when m grows larger as in real-case scenarios. On the other hand, modeling the data-generation process at a computer program is simple and computationally cheap.

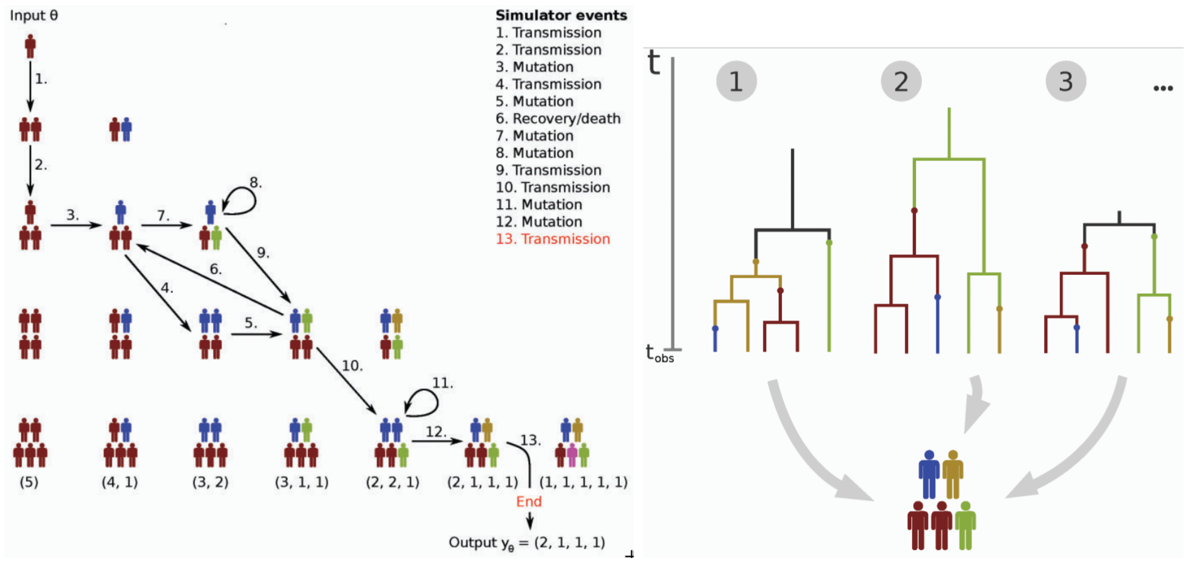

Figure 1: Image taken from [5]

Goal of Simulation-Based Models

As in all Machine Learning (ML) set-ups, the fundamental goal is the derivation of the parameter configuration θ^* that *describes* well the data (i.e. generates samples $M(\theta^*, V)$ that are as close as possible to the observed data y_0). Since Simulation-Based models belong to the broad category of Bayesian Machine Learning, the ultimate goal is to *infer* a posterior distribution $p(\theta|y_0)$ over of all possible configuration set-ups or/and get some samples $\theta \sim p(\theta|y_0)$.

Robust Optimisation Monte Carlo (ROMC) method

The ROMC method [3] is very recent Likelihood-Free approach; its fundamental idea is the transformation of the stochastic data generation process $M(\theta, V)$ to a deterministic function $g(\theta)$, by sampling the variables that the randomness $V_i \sim p(V)$. Hence, we can state that $g(\theta) = M(\theta, V = V_i)$. The ROMC method continues the research line introduced by Meeds et. al [6], by improving some fundamental failure-mode of OMC. ROMC describes a methodology for approximating the posterior through defining and solving deterministic optimisation problems, without enforcing the underlying algorithms for each step; in this sense it can be thought as a meta-algorithm.

Implementation

The most important contribution of this work is the implementation of the ROMC method in the Python package Engine for Likelihood-Free Inference (ELFI) [4]. Since it is very recently published work the ROMC method was not implemented by now in any ML software. This works attempts to provide to the research community a tested and robust implementation for further experimentation.

1.2 Outline of Thesis

The remainder of the dissertation is organized as follows. In Chapter 2 we establish the mathematical formulation; more specifically we initially describe the Simulator-Based models and we provide some fundamental algorithms that have been proposed for performing statistical inference. Afterwards, we provide the mathematical description of the ROMC approach [3]. In Chapter 3, we deal with the implementation part; we initially provide some information regarding the Python package Engine for Likelihood-Free Inference (ELFI) [4] and subsequently we analyze the implementation details of ROMC in this package. In Chapter 4, we present the functionalities of the ROMC implementation at some real-world examples; this chapter wishes to illustrate the success of the ROMC method and of our implementation at Likelihood-Free tasks. Finally, in chapter 5, we conclude with some thoughts on the work we have done and some future research ideas.

1.3 Notation

In this section, we keep an overview of the symbols used throughout the document along with their explanation. We try to keep the notation as consistent as possible, at least to the most central quantities.

- θ parameters of interest, that control the generator
- $M_r(\theta)$ the random generator.
- V the random variables we are not interested in and add stochasticity to the generator. $v_i \sim V$ a random sample drawn from the simulator
- Y_θ random variable descriping the simulator $M_r(\theta)$. Hence, executing the simulator $y_i \sim M_r(\theta)$ produces samples from Y_θ. The pdf of Y_θ is unknown in closed form or intractable to be evaluated.
- $M_d(\theta, v)$ the deterministic generator; if we pass the state v of all stochastic variables that are part of the data generation process, then producing an outcome becomes deterministic.
- y_0 the observations

- $T : \mathbb{R}^{D_1} \rightarrow \mathbb{R}^{D_2}$ where $D_1 > D_2$, the summary statistic mapping.
- $d(\cdot, \cdot)$ a distance between two vectors, e.g. euclidean distance
- $B_\epsilon(y_0)$ the local set of points around y_0, defined as $B_\epsilon(y_0) := \{y : d(y, y_0) < \epsilon\}$
- $\mathbb{1}_{B_{d,\epsilon}(y_0)}(y)$ the indicator function:

$$\mathbb{1}_{B_{d,\epsilon}(y_0)}(y) = \begin{cases} 1 & \text{if } d(y, y_0) \leq \epsilon \\ 0 & \text{else} \end{cases}$$

- $p(\theta)$ the prior distribution on the parameters
- $p(\theta|y_0)$ the posterior distribution
- $p_{d,\epsilon}(\theta|y_0)$ the approximate posterior
- $L(\theta)$ the likelihood function
- $L_{d,\epsilon}(\theta)$ the approximate likelihood function

2 Background

2.1 Simulator-Based (Implicit) Models

As already stated, in Simulator-Based models it is impossible to evaluate the quantity $p_{y|\theta}(y)$. The only tool we own is a black-box simulator $M_r(\theta)$ that can be used to generate data. If we denote as Y_θ the random variable that describes the simulator, then

$$Pr(Y_\theta \in B_\epsilon(y_o)) = Pr(M_r(\theta) \in B_\epsilon(y_o)) = \int_{y \in B_\epsilon(y_0)} p(y|\theta)dy \tag{2.1}$$

On the other hand, the likelihood function can be defined as:

$$L(\theta) = \lim_{\epsilon \to 0} c_\epsilon Pr(Y_\theta \in B_\epsilon(y_0)) = \lim_{\epsilon \to 0} c_\epsilon \int_{y \in B_\epsilon(y_0)} p(y|\theta)dy \tag{2.2}$$

and the posterior distribution as:

$$p(\theta|y_0) \propto L(\theta)p(\theta) \tag{2.3}$$

Since $p_{y|\theta}$ cannot be evaluated, so does $L(\theta)$ and subsequently $p(\theta|y_0)$.

2.1.1 Approximate Bayesian Computation (ABC) Rejection Sampling

ABC Rejection Sampling is a modified version of Rejection Sampling, for cases when likelihood evaluation is intractable. In the Rejection method a sample is obtained from the prior $\theta \sim p(\theta)$ and it is maintained with probability $L(\theta)/\max_\theta L(\theta)$. The samples θ_i obtained with this procedure follow the posterior distribution $p(\theta|y_0)$. Although we cannot use this approach out of the box (evaluating $L(\theta)$ is impossible in our case), we can adjust it with some slight modifications.

In the discrete case scenario where Y_θ can take a finite set of numbers, the likelihood becomes $L(\theta) = Pr(Y_\theta = y_0)$ and the posterior $p(\theta|y_o) \propto Pr(Y_\theta = y_o)p(\theta)$. Hence, we can sample from the prior $\theta_i \sim p(\theta)$, run the simulator $y_i = M(\theta_i, V)$ and maintain θ_i if only $y_i = y_0$.

The above method becomes less usefull as the finite set of possible Y_θ values grows large. As the set grows larger, the probability to maintain a sample becomes smaller. In the limit where the set becomes infinite (i.e. continuous case) the probability becomes zero. In order for the method to work in this set-up, a relaxation is introduced; we relax the acceptance criterion by letting Y_θ lie in a contiguous area around y_0, i.e. $Y_\theta \in B_\epsilon(y_0), \epsilon > 0$. The area can be defined as $B_\epsilon(y_0) := \{y : d(y, y_0) < \epsilon\}$ where $d(\cdot,\cdot)$ can represent any valid distance. With this modification, the maintained samples follow an approximate posterior

$$p_{d,\epsilon}(\theta|y_0) \propto Pr(Y_\theta \in B_\epsilon(y_0))p(\theta) \tag{2.4}$$

where B_ϵ is defined by d, ϵ. This procedure is called Rejection ABC algorithm and forms the basis of Likelihood-Free methods.

2.1.2 Summary Statistics

When the dimensionality of $Y_\theta \in \mathbb{R}^D$ is high, generating samples inside $B_\epsilon(y_0)$ becomes rare; this is the curse of dimensionality. As an representative example if $B_\epsilon(y_0) := \{y : ||y - y_0||_2^2 < \epsilon^2\}$ is a hyper-sphere with radius ϵ and the prior distribution $p(\theta)$ is a uniform distribution in a hyper-cube with side of length $2/epsilon$, the probability of drawing a sample inside the hyper-sphere becomes:

$$Pr(Y_\theta \in B_\epsilon(y_0)) = Pr(\theta \in B_\epsilon(y_0)) = \frac{V_{hypersphere}}{V_{hypercube}} = \frac{\pi^{D/2}}{D2^{D-1}\Gamma(D/2)} \to 0 \text{as} D \to \infty \tag{2.5}$$

We observe that the probality tends to 0, independently of ϵ; enlarging ϵ will not increase the acceptance rate. This produces the need for a mapping $T : \mathbb{R}^{D_1} \to \mathbb{R}^{D_2}$ where $D_1 > D_2$, redefining the area as $B_\epsilon(y_0) := \{y : d(T(y), T(y_0)) < \epsilon\}$. This process is called measuring the distance at the *summary statistics* level.

2.1.3 Approximation

Approximating the posterior as $p(\theta|y_0) \approx p_{d,\epsilon}(\theta|y_0) \propto Pr(Y_\theta \in B_\epsilon(y_0))p(\theta)$ where $B_\epsilon(y_0) := \{y : d(T(y), T(y_0)) < \epsilon\}$ introduces two approximation errors:

- ϵ is chosen to be large enough for enough samples to be accepted
- Summary Statistics (a) make the distance not a metric in a formal sense, i.e. $d = 0$, even if $y \neq y_0$ (b) make possible disjoint sets of y to lie inside $B_\epsilon y_0$

In the following sections we will not use the summary statistics in our expression, for the notation not to clutter. We can state that all the following statements are valid with incorporating summary statistics.

2.1.4 Optimization Monte Carlo (OMC)

We have already defined $B_{d,\epsilon}(y) := \{x : d(y, x) < \epsilon\}$ as the set of points that lie inside area defined by (d, ϵ, y). Based on that, we can define two useful entities; an indicator function and a conditional distribution.

Indicator Function

The indicator function $\mathbb{1}_{B_{d,\epsilon}(y)}(x)$ returns 1 if $x \in B_{d,\epsilon}(y)$ and 0 otherwise. If $d(\cdot, \cdot)$ is a formal distance, due to symmetry $\mathbb{1}_{B_{d,\epsilon}(y)}(x) = \mathbb{1}_{B_{d,\epsilon}(x)}(y)$.

$$\mathbb{1}_{B_{d,\epsilon}(y)}(x) = \begin{cases} 1 & \text{if } x \in B_{d,\epsilon}(y) \\ 0 & \text{else} \end{cases} \tag{2.6}$$

Boxcar Kernel

The boxcar kernel is the conditional distribution:

$$p_{d,\epsilon}(y|x) = \begin{cases} c & \text{if } d(y,x) \leq \epsilon \\ 0 & \text{else} \end{cases} \quad \text{where } c = \frac{1}{\int_{\{y:d(y,x)<\epsilon\}} dy} \tag{2.7}$$

If we understand the boxcar kernel as a data generation process we can make two important notices:

- given a specific x, all values $y : y \in B_{d,\epsilon}(x)$ have equal probability to be generated
- if a specific y value has been generated, all $x : x \in B_{d,\epsilon}(y)$ have equal probability to be the conditional value that lead to this generation

Finally, we can also observe that the kernel can be defined through the indicator function:

$$p_{d,\epsilon}(y|x) = c\mathbb{1}_{B_{d,\epsilon}(y)}(x) = c\mathbb{1}_{B_{d,\epsilon}(x)}(y) \tag{2.8}$$

Initial View

Based on the knowledge we have so far, we could define and approximate the approximate likelihood $L_{d,\epsilon}(\theta)$ and through 2.4 the approximate posterior $p_{d,\epsilon}(\theta|y_0)$. The approximation is given below:

$$L_{d,\epsilon}(\theta) = \int_{B_\epsilon(y_0)} p(y|\theta)dy = \int p_{d,\epsilon}(y_0|y)p(y|\theta)dy \tag{2.9}$$

$$\approx \frac{1}{N}\sum_i^N p_{d,\epsilon}(y_0|y_i) \tag{2.10}$$

$$\approx \frac{c}{N}\sum_i^N \mathbb{1}_{B_{d,\epsilon}(y_i)}(y_0), y_i \sim M_r(\theta) \tag{2.11}$$

This approach is quite intuitive; approximating likelihood of a specific θ requires sampling from the data generator and count the fraction of samples that lie inside the area around the observed data. On the other hand, it has a major disadvantage; evaluating $L_{d,\epsilon}(\theta)$ requires resampling N points and checking which ones are close to the observed data y_0.

Alternative View

OMC attempts an alternative view to the approximation of $L_{d,\epsilon}(\theta)$, which has some advantages. Instead of incorporating the random generator $M_r(\theta)$, it samples all the nuisance variables from a prior distribution $v_i \sim p(v)$ and then it uses the deterministic mapping $M_d(\theta, v_i)$. More formally the approach is the following:

$$L_{d,\epsilon}(\theta) = \int_{B_\epsilon(y_0)} p(y|\theta)dy = \int p_{d,\epsilon}(y_0|y)p(y|\theta)dy \tag{2.12}$$

$$= \int_y \int_v p_{d,\epsilon}(y_0|y)p(y|\theta, v)p(v)dxdv \tag{2.13}$$

$$= \int_v p_{d,\epsilon}(y_0|y = M_d(\theta, v))p(v)dv \tag{2.14}$$

$$\approx \frac{1}{N}\sum_i^N p_{d,\epsilon}(y_0|y = M_d(\theta, v_i)) \tag{2.15}$$

$$\approx \frac{c}{N}\sum_i^N \mathbb{1}_{B_{d,\epsilon}(M_d(\theta,v_i))}(y_0), v_i \sim p(v) \tag{2.16}$$

Based on this approach, the unnormalized approximate posterior can be defined as:

$$p_{d,\epsilon}(\theta|y_0) \propto p(\theta)\sum_i^N \mathbb{1}_{B_{d,\epsilon}(M_d(\theta,v_i))}(y_0) \tag{2.17}$$

Forming an analogy with the previous approach, we sample many nuisance variables in order to absorb the randomness of the generator and we count the fraction of times the deterministic generator produces mapps to outputs close to the observed data. Though it is conceptually close to the previous approach, this approach has a major advantage; we can sample the nuisance variables once (training part) and afterwards evaluate every θ based on a predefined expression (inference part).

2.2 Robust Optimistation Monte Carlo (ROMC) approach

Weighted Sampling

Apart from defining a tractable aproximation of the posterior, Likelihood-Free methods target on sampling from it accurately and efficiently. Sampling can be performed by importance sampling, using the prior as proposal distribution; hence $\theta_i \sim p(\theta)$ and the corresponding weight is $w_i = \frac{L_{d,\epsilon}(\theta_i)}{p(\theta_i)}$. This approach has the same drawbacks as ABC rejection sampling; when the prior is wide, drawing a sample with weight is rare, leading to either poor Effective Sample Size (ESS) or huge execution time.

The ROMC method proposes the construction of a better proposal distribution $q(\theta)$; specifically it proposes the construction of one proposal distribution q_i per sampled nuisance variable v_i. Therefore,

$$w_{ij} = \frac{L_{d,\epsilon}(\theta_{ij})p(\theta_{ij})}{q(\theta_{ij})}, \theta_{ij} \sim q_i(\theta) \tag{2.18}$$

Computing an expectation

Another goal of the method is approximating the quantity $E_{p(\theta|y_0)}[h(\theta)]$. Using the weighted samples from above this can be performed through,

$$E_{p(\theta|y_0)}[h(\theta)] \approx \frac{\sum_{ij} w_{ij} h(\theta_{ij})}{\sum_{ij} w_{ij}} \tag{2.19}$$

2.2.1 Define deterministic optimisation problems

For easier notation, we define as f_i the $i - th$ deterministic problem, namely $f_i(\theta) = M_d(\theta, v_i)$, where $v_i \sim p(v)$. For constructiong the proposal region, we search for a point $\theta_* : d(f_i(\theta_0), y_0) < \epsilon$; this point can be obtained by solving the the following optimisation problem:

$$\min_{\theta} \quad g_i(\theta) = d(y_0, f_i(\theta)) \tag{2.20a}$$

$$\text{subject to} \quad g_i(\theta) < \epsilon \tag{2.20b}$$

We maintain a list of the solutions θ_i^* of the optimisation problems. If for a specific set of nuisance variables v_i, there is no feasible solution we add nothing to the list. The Optimisation problem 2.20a can be treated as unconstrained, accepting the optimal point $\theta_i^* = \text{argmin}_\theta g_i(\theta)$ only if $g_i(\theta_i^*) < \epsilon$.

2.2.2 Gradient-Based Approach

The nature of the generative model $M_r(\theta)$, the properties of the objective function g_i. If g_i is continuous with smooth gradients $\nabla_\theta g_i$ any gradient-based iterative algorithm can be used for solving 2.20a. The gradients $\nabla_\theta g_i$ can be either provided in closed form or approximated by finite differences.

2.2.3 Gaussian Process Approach

In cases where gradients are not defined or they are not available, the Bayesian Optimisation scheme is an alternative choice. Such approach apart from providing an optimal θ_i^*, also fits a surrogate model $\hat{d}_i$ of the distance g_i which can be used for the forthcoming steps. Specifically, in the construction of the proposal region and in equations 2.17, 2.18, 2.19 it could replace g_i in the evaluation of the indicator function 2.6, providing a major speed-up.

2.2.4 Construction of the proposal area q_i

Independently of the approach chosen above, the constraction of the proposal region follows a common method. The search directions $\mathbf{v}_d$ are computed as the eigenvectors of the curvature at θ_i^* and a line-search method is used to obtain the limits. Algorithm 1 describes analytically the method.

2.2.5 Algorithmic Description of Training and Inference Phases

At a high-level, the ROMC method can be split into the training and the inference part.

At the training (fitting) part, the method samples the nuisance variables $v_i \sim p(v)$, defines the the optimisation problems $\min_\theta[g_i(\theta)]$, solves them to obtain θ_i^*, checks whether the optimal point the respects the constraint and finally builds the bounding box for obtaining the proposal region q_i. Using the Gaussian Process set up makes the training part slower due to the fitting of the surrogate model at step 2. On the other hand, computing the q_i becomes faster since the evaluating the distance doesn't

Algorithm 1 Proposal Region q_i construction; Needs, a model of distance d ($\hat{d}$ or g_i), optimal point θ_i^*, number of refinements K, step size η and curvature matrix H_i ($J_i^T J_i$ or GP Hessian)

```
1: Compute eigenvectors $\mathbf{v}_d$ of $H_i$ $(d = 1, \dots, ||\theta||)$
2: for $d \leftarrow 1$ to $||\theta||$ do
3:     $\tilde{\theta} \leftarrow \theta_i^*$
4:     $k \leftarrow 0$
5:     repeat
6:         repeat
7:             $\tilde{\theta} \leftarrow \tilde{\theta} + \eta\ \mathbf{v}_d$                ▷ Large step size $\eta$.
8:         until $d((\tilde{\theta}, i), ) \geq \epsilon$
9:         $\tilde{\theta} \leftarrow \tilde{\theta} - \eta\ \mathbf{v}_d$
10:        $\eta \leftarrow \eta/2$                        ▷ More accurate region boundary
11:        $k \leftarrow k + 1$
12:    until $k = K$
13:    Set final $\tilde{\theta}$ as region end point.
14:    Repeat steps 3 - 13 for $\mathbf{v}_d = -\mathbf{v}_d$
15: Fit a rectangular box around the region end points and define $q_i$ as uniform distribution
```

involve running the whole simulator $M_d^i(\theta)$ for each query point. The algorithms are presented in 2 and 3.

Performing the infernce part includes evaluating the unnormalised posterior and sampling from the posterior. Computing an expectation is not described as a distinct phase, since it can be done in straightforward manner using the weighted samples. For evaluating the unnormalized posterior in the Gradient-Based approach, only the deterministic functions g_i and the prior distribution $p(\theta)$ are required; there is no need for solving the optimisation problems and building the proposal regions. The evaluation requires iterating over all g_i and evaluating the distance from the observed data. In contrast, using the GP approach, there is need for the optimisation part to run for fitting the surrogate models $\hat{d}_i(\theta)$. Afterwards the distance is evaluated on them. The evaluation of the posterior is presented analytically in 6 and 7.

Weighted Sampling is performed by getting n_2 samples from each proposal region q_i, evaluating if the actually fall inside the acceptance region and if so, compute their weight. The procedure is identical in both cases, apart from step 3 where the acceptance check is done in the real model g_i in the Gradient-Based approach and in the surrogate model $\hat{d}_i$ otherwise. The sampling algorithms are presented step-by-step in 4 and 5.

Algorithm 2 Training Part - Gradient approach. Requires $g_i(\theta), p(\theta)$

```
1: for $i \leftarrow 1$ to $n$ do
2:     Obtain $\theta_i^*$ using a Gradient Optimiser
3:     if $g_i(\theta_i^*) > \epsilon$ then
4:         go to 1
5:     else
6:         Approximate $H_i \approx J_i^T J_i$
7:         Use algorihm 1 to obtain $q_i$
    return $q_i, p(\theta), g_i(\theta)$
```

Algorithm 3 Training Part - GP approach. Requires $g_i(\theta), p(\theta)$

```
1: for $i \leftarrow 1$ to $n$ do
2:     Obtain $\theta_i^*, \hat{d}_i(\theta)$ using a GP approach
3:     if $g_i(\theta_i^*) > \epsilon$ then
4:         go to 1
5:     else
6:         Approximate $H_i \approx J_i^T J_i$
7:         Use algorihm 1 to obtain $q_i$
    return $q_i, p(\theta), \hat{d}_i(\theta)$
```

Algorithm 4 Sampling - Gradient Based approach. Requires $g_i(\theta), p(\theta), q_i$

```
1: for $i \leftarrow 1$ to $n_1$ do
2:     for $j \leftarrow 1$ to $n_2$ do
3:         $\theta_{ij} \sim q_i$
4:         if $g_i(\theta_{ij}) > \epsilon$ then
5:             Reject $\theta_{ij}$
6:         else
7:             $w_{ij} = \frac{p(\theta_{ij})}{q(\theta_{ij})}$
8:             Accept $\theta_{ij}$, with weight $w_{ij}$
```

Algorithm 5 Sampling - GP approach. Requires $\hat{d}_i(\theta), p(\theta), q_i$

```
1: for $i \leftarrow 1$ to $n_1$ do
2:     for $j \leftarrow 1$ to $n_2$ do
3:         $\theta_{ij} \sim q_i$
4:         if $\hat{d}_i(\theta_{ij}) > \epsilon$ then
5:             Reject $\theta_{ij}$
6:         else
7:             $w_{ij} = \frac{p(\theta_{ij})}{q(\theta_{ij})}$
8:             Accept $\theta_{ij}$, with weight $w_{ij}$
```

Algorithm 6 Evaluate unnormalised posterior - Gradient approach. Requires $g_i(\theta), p(\theta)$

```
1: $k \leftarrow 0$
2: for $i \leftarrow 1$ to $n_1$ do
3:     if $g_i(\theta) > \epsilon$ then
4:         $k \leftarrow k + 1$
   return $kp(\theta)$
```

Algorithm 7 Evaluate unnormalised posterior - GP approach. Requires $\hat{d}_i(\theta), p(\theta)$

```
1: $k \leftarrow 0$
2: for $i \leftarrow 1$ to $n_1$ do
3:     if $d_i(\theta) > \epsilon$ then
4:         $k \leftarrow k + 1$
   return $kp(\theta)$
```

2.3 Computational Complexity

In the notation, we use S for describing the size of the Simulator. in table we observe that constructing the regions and sampling are the most demanding operations, though $n_2 >> K$. In the Gaussian-Process set-up sampling is not influenced by the size of the simulator.

Compuational Complexity		
	Gradient-Based	Gaussian Process
Optimize	Closed-Form Gradients: $\mathcal{O}(Sn_1)$ Approximate Gradients: $\mathcal{O}(DSn_1)$	
Construct Regions	$\mathcal{O}(KDSn_1)$	$\mathcal{O}(KDn_1)$
Evaluate Posterior	$\mathcal{O}(n_1 S)$	$\mathcal{O}(n_1)$
Sampling	$\mathcal{O}(n_1 n_2 DS)$	$\mathcal{O}(n_1 n_2 D)$

2.4 Engine for Likelihood-Free Inference (ELFI) Package

3 Implementation

3.1 General Design

3.2 Training

3.3 Performing the Infernce

3.4 Utilities

3.5 Computational Complexity

4 Experiments

Add experiments ...

4.1 Another Example

4.2 Execution Time Experiments

5 Conclusions

5.1 Outcomes

5.2 Future Research Directions

References

[1] Yanzhi Chen and Michael U Gutmann. “Adaptive Gaussian Copula ABC”. In: *Proceedings of Machine Learning Research*. Vol. 89. 2019, pp. 1584–1592. URL: `http://proceedings.mlr.press/v89/chen19d.html`.

[2] Michael U. Gutmann and Jukka Corander. *Bayesian optimization for likelihood-free inference of simulator-based statistical models*. 2016. arXiv: `1501.03291`.

[3] Borislav Ikonomov and Michael U. Gutmann. “Robust Optimisation Monte Carlo”. In: (2019). arXiv: `1904.00670`. URL: `http://arxiv.org/abs/1904.00670`.

[4] Jarno Lintusaari et al. *ELFI: Engine for Likelihood Free Inference*. 2018. eprint: `arXiv:1708.00707`.

[5] Jarno Lintusaari et al. “Fundamentals and recent developments in approximate Bayesian computation”. In: *Systematic Biology* 66.1 (2017), e66–e82. ISSN: 1076836X. DOI: `10.1093/sysbio/syw077`.

[6] Edward Meeds and Max Welling. “Optimization Monte Carlo: Efficient and embarrassingly parallel likelihood-free inference”. In: *Advances in Neural Information Processing Systems*. 2015. arXiv: `1506.03693`.

[7] Mark M. Tanaka et al. “Using approximate bayesian computation to estimate tuberculosis transmission parameters from genotype data”. In: *Genetics* (2006). ISSN: 00166731. DOI: `10.1534/genetics.106.055574`.

Appendices

A An Appendix

Some stuff.

B Another Appendix

Some other stuff.