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

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. Practically, a simulator can be understood as a black-box machine M_r that, given parameter θ, produces samples y in a stochastic manner, i.e. $M_r(\theta) \rightarrow y$.

Simulator-Based models are particularly captivating due to the low-level of restrictions they demand in the modeling; 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 or logic-based decisions. On the other hand, this level of freedom comes at a cost; performing inference is particularly demanding from both computational and mathematical perspective. Unfortuenately, the algorithms deployed so far, allow the performance of inference only at low-dimensionality parametric spaces, i.e. $\theta \in \mathbb{R}^D$ where D is small.

Example

For underlying the importance of Simulator-Based models, lets use the tuberculosis disease spread example 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) by the transmission rate α (b) the mutation rate τ and (c) the exclusion rate δ, creating a $3D$-parametric space $\theta = (\alpha, \tau, \delta)$. 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 as a computer program is simple and computationally efficient, hence using a Simulator-Based Model is a perfect fit.

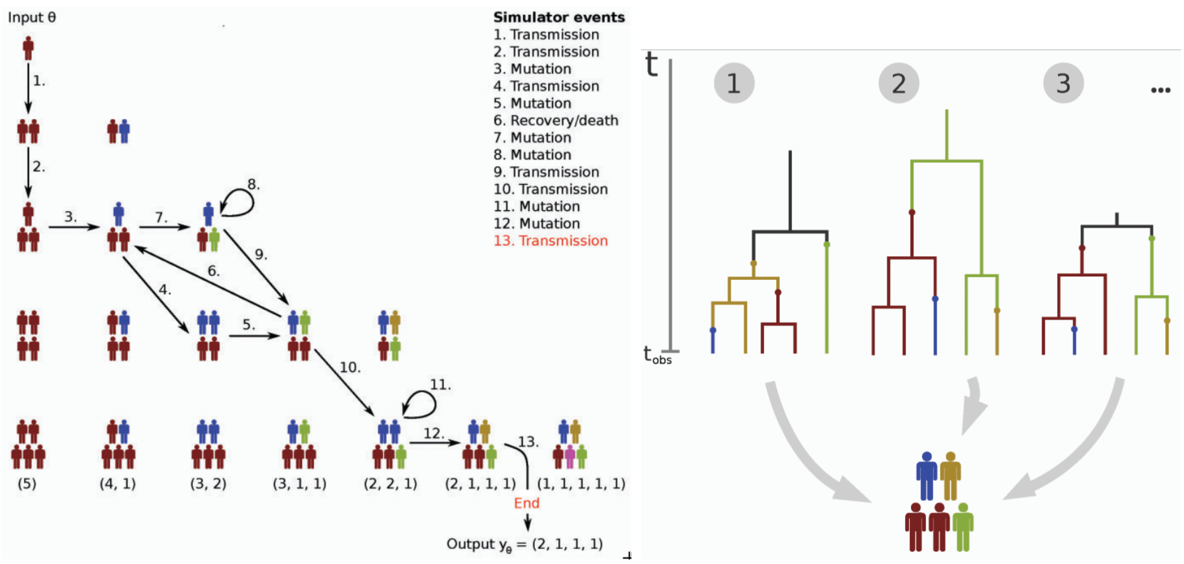

Figure 1: Image taken from [5]

Goal of Simulation-Based Models

As in all Machine Learning (ML) conceptes, the fundamental goal is the derivation of the parameter configuration(s) θ^* that *describe* well the data i.e. generate samples $M_r(\theta^*)$ 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 and obtain some samples from this distribution $\theta \sim p(\theta|y_0)$. Doing so, we have uncovered the mechanism that produces the output, based on passed captured realisations of the phenomenon, and so we are able to achieve a wide range of tasks, such as predicting future outcomes or understanding the internals of the method.

Robust Optimisation Monte Carlo (ROMC) method

The ROMC method [3] is very a recent Likelihood-Free approach; its fundamental idea is the transformation of the stochastic data generation process $M_r(\theta)$ to a deterministic mapping $g(\theta)$, by pre-sampling the variables that produce the randomness $v_i \sim p(V)$. Formally, in every stochastic process the randomness is influenced by a vector of random variables v, whose state is unknown before the execution of the simulation; pre-sampling this state makes the procedure deterministic, namely $g_i(\theta) = M_d(\theta, V = v_i)$. This approach initially introduced by Meeds et. al [6] with the title Optimisation Monte Carlo (OMC). The ROMC extended this approach by improving a fundamental failure-mode of OMC. The ROMC describes a methodology for approximating the posterior through a series of steps, without explicitly enforcing which algorithms must be used for each step[1]; 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 present an overview of the symbols and the quantities used in the document. At this point, the quantities are described quite informally. Most of them will be defined formally in the next chapters. We try to keep the notation as consistent as possible throughout the document. The symbol $\mathbb{R}^N$, when used, describes that a variable belongs to a multi-dimensional space in $\mathbb{R}$; N doesn't represent a specific number.

[1] The implementation chooses a specific algorithm for each step, but the choice has just demonstrative value; any other appropriate algorithm can be used instead.

Random Generator

- $M_r(\boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}$: The black-box data simulator.

Parameters/Random Variables/Symbols

- $D \in \mathbb{R}$, the dimensionality of the parameter-space
- $\boldsymbol{\theta} \in \mathbb{R}^D$, the parameters of interest
- $\mathbf{y_0} n \mathbb{R}^N$, the observations
- $\epsilon \in R$, the threshold for defining the region around $\mathbf{y_0}$
- $\mathbf{v} \in \mathbb{R}^N$, random variable that represents the stochasticity of the generator.
- $\mathbf{v}_i \sim \mathbf{v}$, a specific sample drawn from $\mathbf{v}$
- $\mathbf{Y}_{\boldsymbol{\theta}}$, random variable descriping the simulator $M_r(\boldsymbol{\theta})$. The pdf of $\mathbf{Y}_{\boldsymbol{\theta}}$ is either not unknown or intractable
- $\mathbf{y}_i \sim \mathbf{Y}_\theta$, a sample drawn from $\mathbf{Y}_\theta$. The sample is obtained by executing the random simulator $\mathbf{y}_i \sim M_r(\boldsymbol{\theta})$

Sets

- $B_{d,\epsilon}(\mathbf{y_0})$, the set of points $\mathbf{y}$ around the observations $\mathbf{y_0}$, i.e. $\mathbf{y} := \{\mathbf{y} : d(\mathbf{y}, \mathbf{y_0}) < \epsilon\}$
- $B_{d,\epsilon}^i = B_{d,\epsilon}(\mathbf{y}_i)$, the set of points $\mathbf{y}$ around $\mathbf{y}_i$, i.e. $\mathbf{y} := \{\mathbf{y} : d(\mathbf{y}, \mathbf{y}_i) < \epsilon\}$

Generic Functions

- $p(\cdot)$, any valid pdf
- $p(\cdot|\cdot)$, any valid conditional distribution.
- $p(\boldsymbol{\theta})$, the prior distribution
- $p(\boldsymbol{\theta}|\mathbf{y_0})$, the posterior distribution
- $p_{d,\epsilon}(\boldsymbol{\theta}|\mathbf{y_0})$, the approximate posterior distribution
- $d(\mathbf{x}, \mathbf{y}) : \mathbb{R}^{2N} \rightarrow \mathbb{R}$: any valid distance e.g L2 norm: $||\mathbf{x} - \mathbf{y}||_2^2$

Functions (Mappings)

- $M_d(\boldsymbol{\theta}, \mathbf{v}) : \mathbb{R}^D \rightarrow \mathbb{R}$, the deterministic generator; representing all stochastic variables that are part of the data generation process with the parameter $\mathbf{v}$, transform the stochastic generator to a deterministic.
- $f_i(\boldsymbol{\theta}) = M_d(\boldsymbol{\theta}, \mathbf{v}_i)$, deterministic generator associated with parameter $\mathbf{v}_i$
- $g_i(\boldsymbol{\theta}) = d(f_i(\boldsymbol{\theta}), \mathbf{y_0})$, distance of the generated data from the observations
- $T(\mathbf{x}) : \mathbb{R}^{D_1} \rightarrow \mathbb{R}^{D_2}$ where $D_1 > D_2$, the summary statistic mapping.
- $\mathbb{1}_{B_{d,\epsilon}(\mathbf{y_0})}(\mathbf{y})$, the indicator function; returns 1 iff $d(\mathbf{y}, \mathbf{y_0}) < \epsilon$
- $L(\boldsymbol{\theta})$, the likelihood
- $L_{d,\epsilon}(\boldsymbol{\theta})$, the approximate likelihood

2 Background

2.1 Simulator-Based (Implicit) Models

As already stated, in Simulator-Based models we cannot evaluate the posterior $p(\boldsymbol{\theta}|\mathbf{y_0}) \propto L(\boldsymbol{\theta})p(\boldsymbol{\theta})$, since it is impossible to evaluate the likelihood $L(\boldsymbol{\theta}) = p(\mathbf{y_0}|\boldsymbol{\theta})$. So, we have to use the following property in order to take advantage of the simulator $M_r(\boldsymbol{\theta})$.

$$L(\boldsymbol{\theta}) = \lim_{\epsilon \to 0} c_\epsilon \int_{\mathbf{y} \in B_\epsilon(\mathbf{y_0})} p(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y} = \lim_{\epsilon \to 0} c_\epsilon Pr(M_r(\boldsymbol{\theta}) \in B_\epsilon(\mathbf{y_0})) \tag{2.1}$$

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 typical Rejection Sampling, a sample is obtained from the prior $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ and it is maintained with probability $L(\boldsymbol{\theta})/\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$. Although we cannot use this approach out of the box (evaluating $L(\boldsymbol{\theta})$ is impossible in our case), we can take advantage of the simulator.

In the discrete case scenario, where $\mathbf{Y}_{\boldsymbol{\theta}}$ can take a finite set of numbers, the likelihood becomes $L(\boldsymbol{\theta}) = Pr(\mathbf{Y}_{\boldsymbol{\theta}} = \mathbf{y_0})$ and the posterior $p(\boldsymbol{\theta}|\mathbf{y_0}) \propto Pr(\mathbf{Y}_{\boldsymbol{\theta}} = \mathbf{y_0})p(\boldsymbol{\theta})$. We can sample from the prior $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$, run the simulator $\mathbf{y}_i = M_r(\boldsymbol{\theta}_i)$ and maintain $\boldsymbol{\theta}_i$ only if $\mathbf{y}_i = \mathbf{y_0}$.

The method above becomes less usefull as the finite set of $\mathbf{Y}_{\boldsymbol{\theta}}$ values grows larger, since the probability of maintaining 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 $\mathbf{Y}_{\boldsymbol{\theta}}$ lie in a larger set of points i.e. $\mathbf{Y}_\theta \in B_{d,\epsilon}(\mathbf{y_0}), \epsilon > 0$. The region can be defined as $B_{d,\epsilon}(\mathbf{y_0}) := \{\mathbf{y} : d(\mathbf{y}, \mathbf{y_0}) < \epsilon\}$ where $d(\cdot, \cdot)$ can represent any valid distance. With this modification, the maintained samples follow the approximate posterior,

$$p_{d,\epsilon}(\boldsymbol{\theta}|\mathbf{y_0}) \propto Pr(\mathbf{Y}_{\boldsymbol{\theta}} \in B_{d,\epsilon}(\mathbf{y_0})p(\boldsymbol{\theta}) \tag{2.2}$$

This method is called *Rejection ABC* and forms the basis of Likelihood-Free methods.

2.1.2 Summary Statistics

When the dimensionality of $\mathbf{Y}_{\boldsymbol{\theta}} \in \mathbb{R}^D$ is high, generating samples inside $B_{d,\epsilon}(\mathbf{y_0})$ becomes rare even with large ϵ; this is the curse of dimensionality. As a representative example if (a) d is set to be the euclidean distance the $B_{d,\epsilon}(\mathbf{y_0}) := \{\mathbf{y} : ||\mathbf{y} - \mathbf{y_0}||_2^2 < \epsilon^2\}$ is a hyper-sphere with radius ϵ and (b) the prior distribution $p(\boldsymbol{\theta})$ is a uniform distribution in a hyper-cube with side of length 2ϵ, then the probability of drawing a sample inside the hyper-sphere becomes:

$$Pr(\mathbf{Y}_{\boldsymbol{\theta}} \in B_{d,\epsilon}(\mathbf{y_0})) = Pr(\boldsymbol{\theta} \in B_{d,\epsilon}(\mathbf{y_0})) = \frac{V_{hypersphere}}{V_{hypercube}} = \frac{\pi^{D/2}}{D 2^{D-1} \Gamma(D/2)} \to 0, \quad \text{as} \quad D \to \infty \tag{2.3}$$

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 $B_{d,\epsilon}(\mathbf{y_0}) := \{\mathbf{y} : d(T(\mathbf{y}), T(\mathbf{y_0})) < \epsilon\}$. This dimensioanlity-reduction is called *summary statistic.*

2.1.3 Approximations Introduced

Approximating the posterior as $p_{d,\epsilon}(\boldsymbol{\theta}|\mathbf{y_0}) \propto Pr(\mathbf{Y}_{\boldsymbol{\theta}} \in B_{d,\epsilon}(\mathbf{y_0}))p(\boldsymbol{\theta})$ where $B_{d,\epsilon}(\mathbf{y_0}) := \{\mathbf{y} : d(T(\mathbf{y}), T(\mathbf{y_0})) < \epsilon\}$ introduces two different types approximation error:

- ϵ is chosen to be large enough, so that enough samples are accepted
- T introduces loss of information, making possible a $\mathbf{y}$ far away from the $\mathbf{y_0}$, namely $\mathbf{y} : d(\mathbf{y}, data) > \epsilon$, to enter the acceptance region at after the dimensionality reduction $d(T(\mathbf{y}), T(\mathbf{y_0}) < \epsilon)$

In the following sections we will not use the summary statistics in our expressions, for the notation not to clutter. Though, all the following propositions are valid with the use of summary statistics.

2.1.4 Optimization Monte Carlo (OMC)

Based on $B_{d,\epsilon}$, we can define two useful entities; an indicator function and a conditional distribution.

Indicator Function

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

$$\mathbb{1}_{B_{d,\epsilon}(\mathbf{y})}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in B_{d,\epsilon}(\mathbf{y}) \\ 0 & \text{else} \end{cases} \tag{2.4}$$

Boxcar Kernel

The boxcar kernel is the conditional distribution:

$$p_{d,\epsilon}(\mathbf{y}|\mathbf{x}) = \begin{cases} c & \text{if } d(\mathbf{y},\mathbf{x}) \leq \epsilon \\ 0 & \text{else} \end{cases} \quad \text{where} c = \frac{1}{\int_{\{\mathbf{y}:d(\mathbf{y},\mathbf{x})<\epsilon\}} d\mathbf{y}} \tag{2.5}$$

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

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

$$p_{d,\epsilon}(\mathbf{y}|\mathbf{x}) = c\mathbb{1}_{B_{d,\epsilon}(\mathbf{y})}(\mathbf{x}) = c\mathbb{1}_{B_{d,\epsilon}(\mathbf{x})}(\mathbf{y}) \tag{2.6}$$

Initial View

Based on 2.2, we can approximate the likelihood as:

$$L_{d,\epsilon}(\boldsymbol{\theta}) = \int_{B_\epsilon(\mathbf{y_0})} p(\mathbf{y}|\boldsymbol{\theta})dy = \int p_{d,\epsilon}(\mathbf{y_0}|\mathbf{y})p(\mathbf{y}|\boldsymbol{\theta})d\mathbf{y} \tag{2.7}$$

$$\approx \frac{1}{N}\sum_{i}^{N} p_{d,\epsilon}(\mathbf{y_0}|\mathbf{y}_i) \tag{2.8}$$

$$\approx \frac{c}{N}\sum_{i}^{N} \mathbb{1}_{B_{d,\epsilon}(\mathbf{y}_i)}(\mathbf{y_0}), \mathbf{y}_i \sim M_r(\boldsymbol{\theta}) \tag{2.9}$$

This approach is quite intuitive; approximating the likelihood of a specific $\boldsymbol{\theta}$ requires sampling from the data generator and count the fraction of samples that lie close to the observations. Nevertheless, for every distinct evaluation of $L_{d,\epsilon}(\boldsymbol{\theta})$ N new samples are needed. This make this approach quite inconvenient.

Alternative View

For overcoming the disadvantage introduced above, OMC attempts an alternative approximation. It samples all the nuisance variables from a distribution $v_i \sim p(v)$ and it converts the random simulator to a deterministic mapping $M_d(\theta, v_i)$,

$$
\begin{aligned}
L_{d,\epsilon}(\theta) &= \int_{B_\epsilon(y_0)} p(y|\theta)dy = \int p_{d,\epsilon}(y_0|y)p(y|\theta)dy && (2.10)\\
&= \int_y \int_v p_{d,\epsilon}(y_0|y)p(y|\theta, v)p(v)dxdv && (2.11)\\
&= \int_v p_{d,\epsilon}(y_0|y = M_d(\theta, v))p(v)dv && (2.12)\\
&\approx \frac{1}{N}\sum_i^N p_{d,\epsilon}(y_0|y = M_d(\theta, v_i)) && (2.13)\\
&\approx \frac{c}{N}\sum_i^N \mathbb{1}_{B_{d,\epsilon}(M_d(\theta,v_i))}(y_0), v_i \sim p(v) && (2.14)
\end{aligned}
$$

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

$$
p_{d,\epsilon}(\theta|y_0) \propto p(\theta)\sum_i^N \mathbb{1}_{B_{d,\epsilon}(M_d(\theta,v_i))}(y_0) \tag{2.15}
$$

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,

$$
w_{ij} = \frac{L_{d,\epsilon}(\theta_{ij})p(\theta_{ij})}{q(\theta_{ij})}, \theta_{ij} \sim q_i(\theta) \tag{2.16}
$$

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,

$$
E_{p(\theta|y_0)}[h(\theta)] \approx \frac{\sum_{ij} w_{ij}h(\theta_{ij})}{\sum_{ij} w_{ij}} \tag{2.17}
$$

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:

$$\min_{\theta} \quad g_i(\theta) = d(y_0, f_i(\theta)) \tag{2.18a}$$

$$\text{subject to} \quad g_i(\theta) < \epsilon \tag{2.18b}$$

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 2.18a 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 2.18a. 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 2.15, 2.16, 2.17 it could replace g_i in the evaluation of the indicator function 2.4, 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 4 describes analytically the method.

2.3 Algorithmic Description of ROMC

2.3.1 ROMC as a Meta-Algorithm

As stated in the Introduction, ROMC can be understood as step-by-step alorithmic approach for perfroming the inference in Simulator-Based Models. The particular methods used for solving the sub-tasks are left as a free choice to the user. As presented in Algorithm 1, the methods involved in solving the optimistation problem (step 4) and constructing the bounding box (step 5) are not restricted. The practiosioner may choose any convenient algorithm, judging the trade-offs between accuracy, robustness, efficiency and complexity. In particular for the optimisation step, the choice of the appropriate optimiser should also consider the nature of the simulator that defines the properties of $g_i(\boldsymbol{\theta})$; for example, is the function differentiable, do we know the gradients $\nabla_{\boldsymbol{\theta}}[g_i]$ in closed-form or they should be approximated by finite differences. As described in sections 2.2.2 and 2.2.3, ROMC proposes two alternative optimisation schemes (gradient-based and gaussian-process approach) depending on whether the gradients are available or not.

2.3.2 Training and Inference Algorithms

In this section, we will provide the algorithmic description of the ROMC method; (a) the solution of the optimisation problems using either the gradient based approach or the Gaussian Process alternative and (b) the construction of the Bounding Box. Afterwards, we will discuss the advantages and the disadvantages of each choice both in terms of accuracy and efficiency.

Algorithm 1 ROMC as a Meta-Algorithm. Requires $M_r(\theta), y_0$. Hyperparameters n_1, n_2.

1: **for** $i \leftarrow 1$ to n_1 **do**
2:     Sample a random state $\mathbf{v}_i \sim p(\mathbf{v})$
3:     Define the deterministic mapping $f_i(\boldsymbol{\theta}) = M_d(\boldsymbol{\theta}, \mathbf{v})$ and therefore $g_i(\boldsymbol{\theta}) = d(f_i(\theta), y_0)$.
4:     Obtain $d_i^* = \min_{\boldsymbol{\theta}} [g_i(\boldsymbol{\theta})]$ and $\boldsymbol{\theta}_i^* = \text{argmin}_{\boldsymbol{\theta}} [g_i(\boldsymbol{\theta})]$ using any convenient optimiser.
5:     Approximate the local area $\{\boldsymbol{\theta} : g_i(\boldsymbol{\theta}) < \epsilon \text{ and } d(\boldsymbol{\theta}, \boldsymbol{\theta}_i^*) < M\}$ with a Bounding Box, using any convenient method.
6:     Define a uniform distribution $q_i(\boldsymbol{\theta})$ over the Bounding Box.
7:     **for** $j \leftarrow 1$ to n_2 **do**
8:         $\boldsymbol{\theta}_{ij} \sim q_i(\boldsymbol{\theta})$
9:         Accept $\boldsymbol{\theta}_{ij}$ as posterior sample with weight $w_{ij} = \frac{p(\boldsymbol{\theta}_{ij})}{q_i(\boldsymbol{\theta}_{ij})} \mathbb{1}_{B^i_{d,\epsilon}}(\boldsymbol{\theta}_{ij})$
    return (List with samples $\boldsymbol{\theta}_{ij}$ and weights w_{ij})

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 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 5 and 6.

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 7 and 8.

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

1: **for** $i \leftarrow 1$ to n **do**
2:     Obtain θ_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 4 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 4 to obtain q_i
    return $q_i, p(\theta), \hat{d}_i(\theta)$

Algorithm 4 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 $\mathbf{H}_i$ ($J_i^T J_i$ or GP Hessian)

1: Compute eigenvectors $\mathbf{v}_d$ of H_i $(d = 1, \ldots, ||\theta||)$
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 η.
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

Algorithm 5 Evaluate unnormalised posterior - Gradient approach. Requires $g_i(\theta), p(\theta)$

$k \leftarrow 0$
for $i \leftarrow 1$ to n_1 **do**
    if $g_i(\theta) > \epsilon$ **then**
        $k \leftarrow k + 1$
    return $kp(\theta)$

Algorithm 6 Evaluate unnormalised posterior - GP approach. Requires $\hat{d}_i(\theta), p(\theta)$

$k \leftarrow 0$
for $i \leftarrow 1$ to n_1 **do**
    if $d_i(\theta) > \epsilon$ **then**
        $k \leftarrow k + 1$
    return $kp(\theta)$

Algorithm 7 Sampling - Gradient Based approach. Requires $g_i(\theta), p(\theta), q_i$

for $i \leftarrow 1$ to n_1 **do**
    for $j \leftarrow 1$ to n_2 **do**
        $\theta_{ij} \sim q_i$
        if $g_i(\theta_{ij}) > \epsilon$ **then**
            Reject θ_{ij}
        else
            $w_{ij} = \frac{p(\theta_{ij})}{q(\theta_{ij})}$
            Accept θ_{ij}, with weight w_{ij}

Algorithm 8 Sampling - GP approach. Requires $\hat{d}_i(\theta), p(\theta), q_i$

for $i \leftarrow 1$ to n_1 **do**
    for $j \leftarrow 1$ to n_2 **do**
        $\theta_{ij} \sim q_i$
        if $\hat{d}_i(\theta_{ij}) > \epsilon$ **then**
            Reject θ_{ij}
        else
            $w_{ij} = \frac{p(\theta_{ij})}{q(\theta_{ij})}$
            Accept θ_{ij}, with weight w_{ij}

2.4 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	$\mathcal{O}(n_1 n_2 DS)$	$\mathcal{O}(n_1 n_2 D)$

2.5 Engine for Likelihood-Free Inference (ELFI) Package

llllxb ll'll

3 Implementation

3.1 General Design

3.2 Training

3.3 Performing the Infernce

3.4 Utilities

3.5 Computational Complexity

4 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.