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
Senior Lecturer 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 $M(\theta, V) \to y$

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.

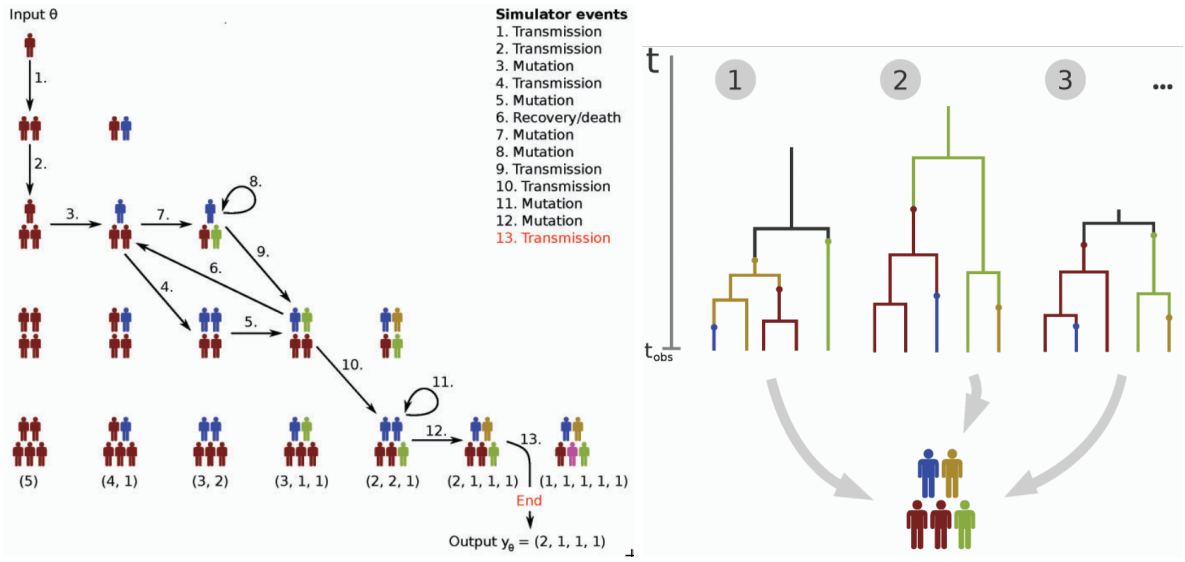

Figure 1: Image taken from [5]

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.

2 Mathematical Modelling

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 $M(\theta, V)$ that can be used to generate data. If we denote as Y_θ the random variable that describes the simulator, then

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

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

$$L(\theta) = \lim_{\epsilon \to 0} c_\epsilon Pr(Y_\theta \in B_\epsilon(y_o)) \tag{2.2}$$

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

Rejection Sampling is well-known method for sampling from a posterior distribution; 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 posterior distribution becomes $p(\theta|y_o) \propto Pr(Y_\theta = y_o)p(\theta)$. 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 to $d(y_i, y_0) < \epsilon$ where $d(\cdot, \cdot)$ can represent any valid distance. With this modification, the maintained samples follow an approximate posterior $p_{d,\epsilon}(\theta|y_0) \propto Pr(Y_\theta \in B_\epsilon(y_0))p(\theta)$, where B_ϵ is defined by d, ϵ. This procedure is called Rejection ABC algorithm and form the basis of Likelihood-Free methods.

2.1.2 Summary Statistics

2.1.3 Optimization Monte Carlo (OMC)

2.2 Robust Optimistation Monte Carlo (ROMC) approach

2.2.1 Define deterministic optimisation problems

2.2.2 Gradient-Based Approach

2.2.3 Gaussian Process Approach

2.2.4 Weighted Sampling

3 Implementation

Now it's getting very technical ... I will cite. I will also show my incredible α, β and γ mathematics and do some other fancy stuff.

3.1 Engine for Likelihood-Free Inference (ELFI) Package

For example look at this

$$\min \sum_{s\in\mathcal{S}} Pr_s \left[\sum_{t=1}^{T} \left(\sum_{g\in\mathcal{G}} \left(\alpha_{gts} C_g^0 + p_{gts} C_g^1 + (p_{gts})^2 C_g^2 \right) + \sum_{g\in\mathcal{C}} \gamma_{gts} C_g^s \right) \right], \tag{3.1}$$

and you will see that it has a little number on the side so that I can refer to it as equation (3.1). Now if I do this

$$\begin{aligned} \sum_{i=1}^{n} k_i &= 20 \\ \sum_{j=20}^{m} \delta_i &\geq \eta \end{aligned} \tag{3.2}$$

I can align two formulae and control which one has a number on the side. It is (3.2). I can also do something like this

$$Y_l = \begin{bmatrix} \left(y_s + i\frac{b_c}{2}\right)\frac{1}{\tau^2} & -y_s \frac{1}{\tau e^{-i\theta^s}} \\ -y_s \frac{1}{\tau e^{i\theta^s}} & y_s + i\frac{b_c}{2} \end{bmatrix},$$

and it won't have a number on the side. Now if I have to do some huge mathematics I'd better structure it a little and include linebreaks etc. so that it fits on one page.

$$\begin{aligned} p_l^f &= G_{l11}\left(2v_{F(l)}\bar{v}_{F(l)} - \bar{v}_{F(l)}^2\right) \\ &+ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ &+ \begin{bmatrix} \bar{v}_{T(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\left[B_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[B_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) - G_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \\ \bar{v}_{F(l)}\bar{v}_{T(l)}\left[-B_{l12}\cos\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right) + G_{l12}\sin\left(\bar{\delta}_{F(l)} - \bar{\delta}_{T(l)}\right)\right] \end{bmatrix} \cdot \begin{bmatrix} v_{F(l)} - \bar{v}_{F(l)} \\ v_{T(l)} - \bar{v}_{T(l)} \\ \delta_{F(l)} - \bar{\delta}_{F(l)} \\ \delta_{T(l)} - \bar{\delta}_{T(l)} \end{bmatrix}, \end{aligned} \tag{3.3}$$

This is a lot of fun!

3.2 Implementation of the ROMC algorithm

Finally we should have a nice picture like this one. However, I won't forget that figures and table are environments which float around in my document. So LaTeX will place them wherever it thinks they fit well with the surrounding text. I can try to change that with a float specifier, e.g. [!ht]. Now I want to use one of my own environments. I want to define something.

Definition 3.1 *I define*

$$\Gamma_\eta := \sum_{i=1}^{n} \sum_{j=i}^{n} \xi(i,j)$$

I definitely need some good tables, so I do this. I should really refer to Table 1.

Case	Generators	Therm. Units	Lines	Peak load: [MW]	[MVar]
6 bus	3 at 3 buses	2	11	210	210
9 bus	3 at 3 buses	3	9	315	115
24 bus	33 at 11 buses	26	38	2850	580
30 bus	6 at 6 buses	5	41	189.2	107.2
39 bus	10 at 10 buses	7	46	6254.2	1387.1
57 bus	7 at 7 buses	7	80	1250.8	336.4

Table 1: Something that doesn't make sense.

3.2.1 Training Part

3.2.2 Inference Part

3.2.3 Inspection Tools

3.2.4 Evaluation and Visualisation

3.3 Computational Complexity

4 Experiments

Add experiments ...

4.1 Higher-Dimension Example

4.2 Computational Complexity

5 Conclusions

5.1 Outcomes

5.2 Future Research Directions

I have no idea how to conclude, so I don't write much. But the stuff that follows is important. lala

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.