Summary of Optimization Monte Carlo methods

Vasileios Gkolemis

June 2020

1 Symbols for functions/probabilities

- $p(x|y)$: any valid pdf function $g(x, y) : \int_x g(x, y) = 1$.
- $p(x|y = y_0)$ is a function of x (i.e. $g(x) : \int_x g(x) = 1$)
- $p(x = x_0|y)$ is a function of y (i.e. $g(y)$)
- $p(x = x_0|y = y_0)$ is a real number $\kappa \in [0, 1]$

2 Symbols for random variables or parameters

- x, y are dummy variables, used as placeholders
- D the available data (observed variables)

3 Likelihood-Free methods Introduction

In probabilistic models, the random variables are split into 2 categories; the observed ones (visibles) and the unobserved. The unobserved variables can be furtherly split into 3 categories:

- the parameters of the model θ, over which we try to infer a posterior distribution
- the hidden/latent variables h, which we cannot observe but are part of the data generation process
- the nuisance variables u, which can be though as dummy random variables that model the randomness when sampling from a distribution. Hence, when the value of the nuisance variables is known, the sampling procedure $x \sim p(x|\theta)$ transforms to a deterministic function $x = f(\theta, u)$

The particularity of likelihood-free methods is that we cannot evaluate the likelihood function $p(y|\theta)$ but we are able to sample from it. Intuitively, we can think of holding a black box simulator (stochastic function) that given a set of parameters, can generate samples $y \sim M_{rand}(\theta)$. As explained above, the random generator can be transformed to a deterministic, by modelling the nuisance variables; hence $y = M(\theta, u)$.

Likelihood-free methods target the following tasks:

- to estimate the pdf of the posterior distribution $p(\theta|y = D)$
- to get some samples from the posterior distribution $\theta \sim p(\theta|y = D)$
- to estimate $E_{P(\theta|y=D)}[h(\theta)]$

4 Background of Likelihood-Free methods

The fundamental idea of all Likelihood-Free methods is that we try to generate (artificial) data, using our simulator $M_{rand}(\theta)$ that is close to the observed (real) data D. If we achieve so, the parameters θthat produced those data are "kept" as candidates of the posterior.

In the contiguous case it is impossible to exactly reproduce the data, hence a relaxation is introduced:

$$p(\theta|y = D) \approx p(\theta|y \in B_\epsilon(D)) = \frac{p(\theta)p(y \in B_\epsilon(D)|\theta)}{p(y \in B_\epsilon(D))} \tag{1}$$

$B_\epsilon(D)$ represents an area around D governed by the hyper-parameter ϵ. In the limit where $\epsilon \to 0$, the approximation becomes equality.

When the data belongs to a high-dimensional space (high in our methods can be even 10 dimensions), it becomes difficult to reproduce the observed data $M_{rand}(\theta) \in B_\epsilon(D)$ without increasing ϵ exponentially (curse of dimensionality). Hence, we normally use a way to project our data to a lower dimension; we use a function $T : \mathbf{R}^{D_1} \to \mathbf{R}^{D_2}$ where $D_1 > D_2$. Then, the equation 1 becomes:

$$p(\theta|y = D) \approx p(\theta|T(y) \in B_\epsilon(D)) = \frac{p(\theta)p(T(y) \in B_\epsilon(D)|\theta)}{p(y = D)} \tag{2}$$

For simplicity we won't use the function T anymore, but all the expressed ideas can be easily applied in the presence of T.

5 Background of Optimization Monte Carlo (OMC) methods

The fundamental idea behind OMC is the creation of a dual problem. We denote as $p_\epsilon(x_1|x_2)$ a kernel distribution; in the simplest case the boxcar kernel.

Primal View

$$\begin{aligned} L(\theta) = p(y \in B_\epsilon(D)|\theta) = p(M_{rand}(\theta) \in B_\epsilon(D)|\theta) && (3)\\ = \int_x p_\epsilon(y = D|x)p(x|\theta)dx && (4)\\ \approx \frac{1}{N}\sum_i^N p_\epsilon(y = D|x = x^{(i)}), x^{(i)} \sim p(x|\theta) && (5)\end{aligned}$$

Inuitively: the probability the parameters θ to have generated the data D is **the summing over all possible data x the simulator can produce and checking for each one if it is close to D.**

Dual View

$$\begin{aligned} L(\theta) = p(y \in B_\epsilon(D)|\theta) = p(M_{rand}(\theta) \in B_\epsilon(D)|\theta) && (6)\\ = \int_x\int_u p_\epsilon(y = D|x)p(x|\theta, u)p(u)dxdu && (7)\\ = \int_u p_\epsilon(y = D|x = M(\theta, u))p(u)du && (8)\\ \approx \frac{1}{N}\sum_i^N p_\epsilon(y = D|x = M(\theta, u^{(i)}), u^{(i)} \sim p(u) && (9)\end{aligned}$$

Intuitively: the probability the parameters θ to have generated the data D is **the summing over all the possible nuisance variables and checking for each one if it (deterministically) produces data close to D.**

Setting $p_\epsilon(y|x)$ to be the boxcar kernel

If $p_\epsilon(y = D|x = M(\theta, u^{(i)}))$ is a boxcar kernel:

$$p_\epsilon(y = D|x = M(\theta, u^{(i)})) = \begin{cases} c & \text{if } d(D, M(\theta, u^{(i)})) \leq \epsilon \\ 0 & \text{else} \end{cases} \tag{10}$$

$$\tag{11}$$

then there is a set $S^{(i)} = \{\theta : d(D, M(\theta, u^{(i)})) < \epsilon\}$ of θ values that generate data "close enough" to D. Using this kernel we do not express any preference among those θ; we consider them equally probable to have been the parameter setting that generated the data, with probability c. The size of the set $S^{(i)}$ is related to ϵ in a positive way; increasing ϵ increases $S^{(i)}$ but in a non-linear way. We have to investigate more this relationship.

The value of c is the volume of the accepatnce area and is related to (a) the dimensionality of the problem d (b) the radius ϵ and (c) the type of the distance

$d(\cdot,\cdot)$. Hence, it is independendent of the size of the set $S^{(i)}$ and is constant for all nuisance variables u_i (i.e. independent of (i)). If the distance is defined as the euclidean distance $d(\cdot,\cdot) = \|\cdot,\cdot\|_2$, then $c = \frac{1}{V}$ where V is the volume of the $n - ball$:

$$V = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}\epsilon^d \tag{12}$$

$$\Gamma(d) = (d-1)! \tag{13}$$

$$\Gamma(d+\frac{1}{2}) = (d-\frac{1}{2})(d-\frac{3}{2})\cdot\cdot\frac{1}{2}\pi^{\frac{1}{2}} \tag{14}$$

We observe that $c \propto \frac{1}{\epsilon^d}$. Hence, in high-dimensionality (i.e. $d \geq 10$), if $\epsilon > 1$, c becomes very small. On the other hand, if $\epsilon < 1$, c becomes very large.

Intuition: if c is large, if a specific θ is inside the boxcar kernel it has large probability to be the θ that generated the data. If c is small, θ inside the boxcar kernel has small probability to have generated the data.

Posterior when $p_\epsilon(y|x)$ is the boxcar kernel

$$L(\theta) = \frac{1}{N}\sum_i^N g_i(\theta) \tag{15}$$

$$g_i(\theta) = p_\epsilon(y = D|x = M(\theta, u^{(i)})) = \begin{cases} c & \text{if } d(D, M(\theta, u^{(i)})) \leq \epsilon \\ 0 & \text{else} \end{cases} \tag{16}$$

$$= c\mathbb{1}_{C_\epsilon^{(i)}}(\theta) \tag{17}$$

With these assumptions the posterior distribution can be computed as:

$$p(\theta|y = D) = \frac{p(\theta)p(y = D|\theta)}{\int_\theta p(\theta)p(y = D|\theta)} \tag{18}$$

$$= \frac{1}{Z}p(\theta)\sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta) \tag{19}$$

$$Z = \int_\theta p(\theta)\sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta)d\theta \tag{20}$$

We observe that c disappears. There are 2 difficulties with using this posterior:

- Computing the partition function is difficult since $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ can represent any abstract shape
- Evaluating a specific $\theta = \theta_0$ requires N forward-passes of the simulator (one for each nuisance variable) which can be expensive.

Expectation on posterior, when $p_\epsilon(y|x)$ is the boxcar kernel

$$E_{p(\theta|y=D)}[h(\theta)] = \int_\theta h(\theta)p(\theta|y=D)d\theta \tag{21}$$

$$= \frac{1}{Z}\int_\theta h(\theta)p(\theta)\sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta)d\theta \tag{22}$$

$$Z = \int_\theta p(\theta)\sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta)d\theta \tag{23}$$

Changing the order:

$$E_{p(\theta|y=D)}[h(\theta)] = \frac{1}{Z}\sum_i \int_\theta h(\theta)p(\theta)\mathbb{1}_{C_\epsilon^{(i)}}(\theta)d\theta \tag{24}$$

$$Z = \sum_i \int_\theta p(\theta)\mathbb{1}_{C_\epsilon^{(i)}}(\theta)d\theta \tag{25}$$

Both integrals can be approximated by inventing a proposal distribution $q^{(i)}$ for each nuisance variable:

$$E_{p(\theta|y=D)}[h(\theta)] \approx \frac{1}{Z}\sum_i\sum_j h(\theta^{(ij)})\frac{p(\theta^{(ij)})}{q(\theta^{(ij)})}\mathbb{1}_{C_\epsilon^{(i)}}(\theta^{(ij)}) \tag{26}$$

$$Z = \sum_i\sum_j \frac{p(\theta^{(ij)})}{q(\theta^{(ij)})}\mathbb{1}_{C_\epsilon^{(i)}}(\theta^{(ij)})d\theta \tag{27}$$

$$\text{where } \theta^{(ij)} \sim q^{(i)} \tag{28}$$

Here, the difficult part is to define proposal regions $q^{(i)}$ with high acceptance rate i.e. $r = \frac{\sum_i\sum_j \mathbb{1}_{C_\epsilon^{(i)}}(\theta^{(ij)})}{NM}$ to be as high as possible. As a second goal, the fractions $\frac{p(\theta^{(ij)})}{q(\theta^{(ij)})}$ should be as high as possible.

6 OMC algorithm

OMC approximates each set $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ with a weighted delta $w_i\delta(\theta-\theta_*^{(i)})$. Hence the posterior **??** becomes:

$$p(\theta|y=D) = \frac{\sum_i^N w_i p(\theta_*^{(i)})\delta(\theta-\theta_*^{(i)})}{\sum_i^N w_i p(\theta_*^{(i)})} \tag{29}$$

which is very easy to evaluate, sample or compute an expectation on it. Estimating an expected value can be done through:

$$E_{p(\theta|y=D)}[h(\theta)] = \frac{\sum_i^N w_i p(\theta_*^{(i)}) h(\theta_*^{(i)})}{\sum_i^N w_i p(\theta_*^{(i)})} \tag{30}$$

The only question is how to compute $\theta_*^{(i)}, w_i$; OMC initially approximates $g_i(\theta)$ as an ellipsoid around $\theta_*^{(i)}$ with volume $V^{(i)}$. This ellipsoid is then shrinked into a delta placed at its center with weight being its volume. Intuitively, since all values inside the ellipsoid have equal probability of generating the data, their mass is being cummulated in the center point.

Alogorithm for obtaining $\theta_*^{(i)}$, $V^{(i)}$

Firstly we search for a point inside the boxcar kernel $\theta_0^{(i)} : \mathbb{1}_{C_\epsilon^{(i)}}(\theta_0^{(i)}) = 1$. For obtaining such a point, any gradient-based optimiser can be used for solving the following problem:

$$\min_\theta \qquad f(\theta) = d(D - M(\theta, u^{(i)})) \tag{31a}$$

$$\text{subject to} \qquad f(\theta)) < \epsilon \tag{31b}$$

$$\tag{31c}$$

In the simplest case, d can be the euclidean distance. Let's say $\theta_0^{(i)}$ is an approximate solution to the problem i.e. $\theta_0^{(i)} \approx \text{argmin}_\theta f(\theta)$ and $f(\theta_0^{(i)}) < \epsilon$, as returned by the optimizer. Then, the central point of the ellipsoid is:

$$\theta_*^{(i)} = \theta_0^{(i)} + J_0^{-1}(D - M(\theta_0^{(i)}, u^{(i)})) \tag{32}$$

and the volume covered is:

$$V^{(i)} = \frac{\gamma}{\sqrt{det(J_0^{(i)T} J_0^{(i)})}} \tag{33}$$

(Question: Is $M(\theta, u^{(i)})$ a smooth function with respect to θ? If not how is the problem solved?)

7 Robust Optimization Monte Carlo (ROBC)

OBC has some disadvantages:

- For each set of nuisance variables $u^{(i)}$ it relates only one $\theta_*^{(i)}$ i.e. single-mode function
- It shrinks all the area mass (however big this area is) to a single point
- The size of the area can be overestimated due to ill conditioned jacobian matrix J and hence dominate the posterior

7.1 Gradients available

ROBC attempts to solve the two latter problems, by approximating the sets $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ as:

$$\mathbb{1}_{C_\epsilon^{(i)}}(\theta) \approx \mathbb{1}_{C_{\text{BB}}^{(i)}}(\theta) \cap \mathbb{1}_{C_\epsilon^{(i)}}(\theta)$$

Intuitively: We design one bounding box around a **single** continuous set of $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$. If $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ includes multiple disjoint sets, we extract only one of them. The bounding box must be big enough to include all the specific set of $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ (for not deleting parts of the posterior) and also small enough since it will serve as a proposal region for drawing samples.

Hence the posterior becomes:

$$p(\theta|y=D) = \frac{p(\theta)p(y=D|\theta)}{\int_\theta p(\theta)p(y=D|\theta)} \tag{34}$$

$$= \frac{1}{Z} p(\theta) \sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta) \mathbb{1}_{C_{\text{BB}}^{(i)}}(\theta) \tag{35}$$

$$Z = \int_\theta p(\theta) \sum_i \mathbb{1}_{C_\epsilon^{(i)}}(\theta) \mathbb{1}_{C_{\text{BB}}^{(i)}}(\theta) d\theta \tag{36}$$

Computing an expectation $E_{p(\theta|y=D)}[h(\theta)]$ can be done from 26 with just setting $q^{(i)}$ to be a uniform distribution over the bounding box.

Algorithm for obtaining $\theta_*^{(i)}$, $V^{(i)}$

The task is computing the $BB^{(i)}$. The central point $\theta_*^{(i)}$ and the jacobian at this point $J^{(ij)T}J^{(ij)}$ can be computed exactly as in OMC. The boundaries are computed as the edge points along the eigenvectors of $J^{(ij)T}J^{(ij)}$.

Hence, ROMC is identical to OMC in the optimization part but differs in the way it represents and samples from the posterior.

7.2 Gradients not available

Questions

Some open questions to ask to Michael at the call

- Is $M(\theta, u)$ smooth w.r.t. θ? what is exactly the nuisance variables? If it is the seed it can take only integer values? Yes, we can suppose that $M(\theta, u)$ smooth w.r.t. θ. In general, the nuisance variables are the underlying random numbers that are generated under the hood when a random generator is called. At high level when calling a random routine, the nuisance variables could be the random seed.

- How ROMC solves the problem of ill-conditioned Jacobian matrix? In option 1, even though the proposal region is overestimated, the samples are filtered by $\mathbb{1}_{C_\epsilon^{(i)}}(\theta)$ so the samples are not going to be overestimated- Probably refers to the GP option.
- In OMC the central point of the ellipsoid is not the outcome of the optimizer (distinction between θ_0 and θ_*)
- Can we sample from the posterior of $ROMC$?
- Questions about the case that we do not have gradients. After we get the boundary box with the algorithm, why we fit a regression model? Why we don't keep the box?
- How to start coding at ELFI?

Some ideas

- The ROMC method has something that is not very intuitive to me. It defines a bounding box as the proposal area and then it accepts the samples if they are inside the acceptance region; it doesn't attach any weight to the generated samples. This means for example that if there is a very small set of acceptable θ and we draw a very small rectangle around it as the proposal region, almost all generated samples are accepted, this very small region will be overly-represented with too many samples in the posterior. Shouldn't we make a normalisation?

  Normalisation Idea

  if V^i is the volume of the approximate (loose) bounding box, N the number of samples generated from itand $N_1^{(i)}$ the number of accepted samples, then approximate volume of the acceptance area is $\hat{V}^{(i)} = ||\mathbb{1}_{C_\epsilon^{(i)}}(\theta)|| \approx \frac{N_1^{(i)}}{N}V^{(i)}$. Hence, in order to keep the proportionalities of the regions and not overly-represent some areas the following should hold. If $\bar{\theta}^{(i)} = [\theta_1^{(i)}, ..., \theta_{N_1^{(i)}}^{(i)}]$, $N_1^{(i)}$ the current number of elements of the set and the $\hat{N}_1^{(i)}$ the final number of elements after interpolating/discarding, then

  $$\frac{\hat{V}^{(i)}}{\hat{V}^{(j)}} \approx \frac{N_1^{(i)}}{N_1^{(j)}}\frac{V^{(i)}}{V^{(j)}} \approx \frac{\hat{N}_1^{(i)}}{\hat{N}_1^{(j)}}$$

  So a simple method of interpolating points must be implemented.

- If instead of boxcar kernel we use a Gaussian $p(y = D|x) = \mathbb{N}(y; x, \sigma^2)$ then the optimizer will return θ_0; then $g_i(\theta)$ becomes a Gaussian as well? If this is the case, then the posterior becomes a mixture of Gaussians which seems convenient.

8 Implementation

Symbols:

- dim: The dimensionality of the inference problem. On how many parameters I want to define the posterior.

8.1 Parameter Inference

8.2 Utils

8.2.1 Classes

- $NDimBoundingBox$
- $OptimisationProblem$
- $RomcPosterior$

8.2.2 Functions

Three functions for creating the Bounding Box region:

- gt_around_theta: (θ_0, func, lim, step, eps) $\rightarrow$ List[nDimBoundingBox]: It measures the bounding box by following the canonical directions and moving one step at a time, until $f(\theta_0 + ke_d) > eps$. So $k = i * step$. Works for all dimensionalities. Complexity: $\mathcal{O}(dim \times step \times lim)$
- romc_jacobian: (θ_0, func, lim, step, eps) $\rightarrow$ List[nDimBoundingBox]: It measures the bounding box by following the eigenvectors of the curvature at the point θ_0 and moving one step at a time, until $f(\theta_0 + ke_d) > eps$. So $k = i * step$. Works for all dimensionalities. Complexity: $\mathcal{O}(dim \times step \times lim)$
- gt_full_coverage:(θ_0, func, left_lim, right_lim, step, eps) $\rightarrow$ List[nDimBoundingBox]: It computes **all** bounding boxes inside the square region defined by left_lim, right_lim parameters. In all dimensions it follows the canonical directions. Works only for 1D case. Complexity: $\mathcal{O}(dim = 1 \times right\_lim \times left\_lim \times step)$

Create deterministic functions utility:

- create_deterministic_generator: wrapper functions that gets the ElfiModel and the nuisance variable and returnes a function: (theta_flat) $\rightarrow$ (dictionary with all output nodes)
- create_output_function: selcts only the output node from the deterministic generator

Utility functions:

- collect_solutions (OptimisationProblems) → (bounding_boxes, funcs, funcs_unique, nuisance)): Collects all the solutions already obtained (i.e. all the solved optimisation problems for which at least one bounding box region is already obtained). Used before defining the ROMC posterior.

test test test