Master's Programme in Data Science

VMBC Report

Tuomas Vuontisjarvi

October 13, 2025

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Degree programme Master’s Programme in Data Science

Tekijä — Författare — Author

Tuomas Vuontisjarvi

Työn nimi — Arbetets titel — Title

VMBC Report

Työn laji — Arbetets art — Level Report	Aika — Datum — Month and year October 13, 2025	Sivumäärä — Sidantal — Number of pages 9

Tiivistelmä — Referat — Abstract

This report is about Variational Bayesian Monte Carlo (VBMC), a method for performing Bayesian inference with complex and computationally expensive black-box models. Key concepts related to the VMBC are explained to provide clear a understanding of the background and the algorithm. As the algorithm is also offered as a python package, usage examples are explored and used to explain VMBC even further.

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

According to Acerbi[3], a significant problem with probabilistic models that have expensive, black-box likelihoods is that the characteristics prevent the usage of standard techniques for Bayesian inference.

In order to address the problem of high computational cost, a novel sample efficient method has been introduced, called Variational Bayesian Monte Carlo (VMBC). Acerbi claims that the VMBC solves the previously costly problem by combining variational inference with Bayesian quadrature, solving model posteriors efficiently and with a relatively small amount of sampling[3].

This report aims to explain the VMBC by exploring the key concepts behind it. Chapter two focuses on explaining all the relevant concepts and providing examples and on the way. The goal is to build a clear picture of the mathematical notions involved with the algorithm. Chapter three will dive deeper into the workings of the VMBC by using the python package pyVBMC provided by the author.

2. Key concepts and VMBC explained

The VMBC is used for complex and computationally expensive black-box models. Acerbi notes a few examples of such models, such as computational neuroscience, biology and big data models[3][4]. The VMBC algorithm is a novel approximate inference method for investigating such black-box models.

A model is a computationally expensive black-box model when there is no access to its inner processes - it can be viewed completely in terms of its inputs and outputs - and the evaluation of the model is time consuming, one second or more per evaluation[1].

One way of learning about models is Bayesian inference, which is a method for computing posterior distribution over parameters and the model evidence. However, since Bayesian inference is generally analytically intractable, statistical approximate inference methods are often used.[3]

Inference methods include Markov Chain Monte Carlo algorithms and variational inference. As Acerbi notes, such methods generally require knowledge about the model processes or a significant number of evaluations[3]. Since neither of those are available for expensive black-box models, the existing methods are unfit for the inference. This is exactly the problem that VMBC aims to solve.

The VMBC produces a flexible approximate posterior distribution of the model parameters [1]. The posterior distribution is a joint probability distribution which describes how plausible each parameter is given the observed data.

The complete operation of the VMBC can be viewed in terms of the following formula:

$$p(x \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid x)p(x)}{p(\mathcal{D})}$$

The posterior distribution is expressed as $p(x \mid \mathcal{D})$, where $\mathcal{D}$ is the dataset or the evidence and vector $x \in \mathbb{R}^D$ describes the black-box model parameters.

The VMBC is an iterative algorithm. Each iteration of the algorithm consists roughly of the following steps.

1. Sample sequentially a batch of points, which maximize a given acquisition function and evaluates the log joint f for each of them

2. Train a GP-model

3. Update the posterior approximation, optimizing ELBO

The log joint f refers to the numerator of the VMBC formula: $p(\mathcal{D} \mid x)p(x)$, where $p(\mathcal{D} \mid x)$ is the expensive black-box model. From the user point-of-view, the black-box models is provided as a function, which computes the log-likelihood of the sample points.

For example, the log-likelihood function could simply be the distance from origon inverted in 2-D space. The further the given sample point is from the origon, the less likely it would be. The loglikelihood function g then would would be just:

$$g(x, y) = \log \frac{1}{x^2 + y^2}$$

The log joint f includes also the prior over the paramaters $p(x)$, which is given as log function for the pyVMBC. Since both likelihood and prior are given as log, the pyVMBC uses their sum, which the user needs to provide as the log-joint function as input parameter for the algorithm.

$$f_{joint}(x) = \log(p(\mathcal{D} \mid x)p(x)) = \log(p(\mathcal{D} \mid x)) + \log(p(x))$$

The algorithm also has to compute the marginal likelihood $p(\mathcal{D})$ in order to solve the posterior. Fong and Holmes define marginal likelihood as a measure of model fit[2] and Acerbi also refers to it as the model evidence[3]. The marginal likelihood is the integral over the parameter space:

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid x)p(x)dx$$

Marginal likelihood is therefore a function, which means that it is a mathematical description of the likelihoods over the whole paramater space.

Solving the marginal likelihood problem has two components. First is to somehow learn about the function itself. Since the model in question is a black-box model, there is no way to directly learn about the model fit. The second problem is the integral part. Even if there was complete knowledge about the underlying function, there is no guarantee that the integral is analytically tractable. Therefore, the VMBC combines Bayesian quadrature with Gaussian process to estimate the marginal likelihood.

According to Williams and Rasmussen, a Gaussian process is a generilaziation of the Gaussian probability distribution[6]. They use the example of classifying handwritten symbols and a training set. The goal is not to just classify the existing data but move on to new unknown data and correctly classifying those. The task then is to transition from finite data to a function which produces correct outputs for all possible inputs. The problem is that there has to be some assumptions made about the function. A broad a definition could lead to overfitting, where the function describes the existing dataset very well but fails to produce correct outputs for new data. Or the definition could be too strict and fail to match to the target function.

One possible solution is to give prior probability to every possible function. But as Williams and Rasmussen note, this leads to performing calculation on an infinite number of different possible functions. Instead, functions are considered as long vectors, where each vector value describes the output for the function $f(x)$ with the particular input. Even though these function vectors are infinetely long, the advantage of Gaussian process is that it gives the same answer for a finite number of points, whether you ignore all the infinite number of other points or not. And answers with different finite sets are consistent with each other.[6]

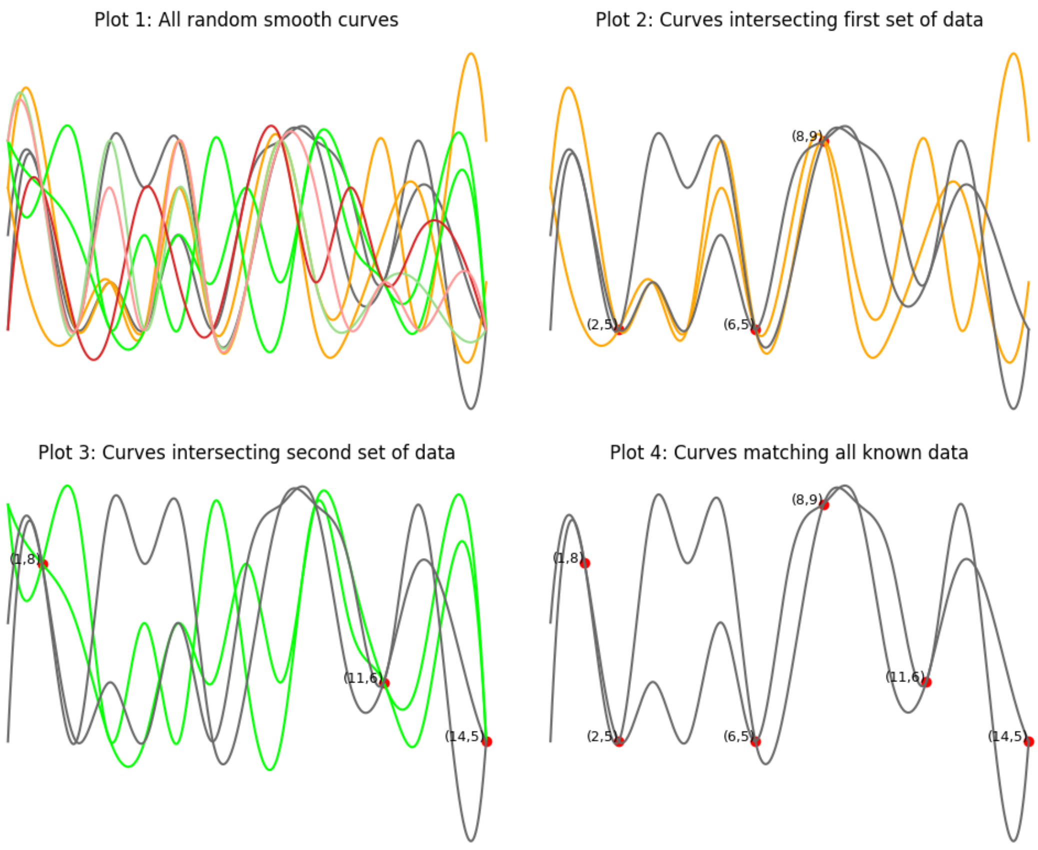

Figure 2.1: Smooth curves representing

The fig. 2.1 provides a (flawed) illustration to build intuition. Plot 1 describe a large set of random smooth curves. Plot 2 shows curves that match the first set of data, reducing the number of possible curves. Plot 3 shows curves matching a different set of known data. Finally plot 4 shows the curves that match all of data. Notice that the curves in plot 4 can also be found in plot 2 and 3. This is a visual representation of how Gaussian process gives consistent answers for different data - the set of possible curves gets restricted with data, but all considered sets of data include the curves consistent with other sets of data. In actuality the GP doesn't deal with any particular function but a probability distribution of functions.

The VMBC algorithm uses GP-based Bayesian quadrature to solve the marginal likelihood. The GP-model provides a prior for the possible functions and the Bayesian quadrature or Bayesian Monte Carlo is used for active sampling[3]. According to Rasmussen and Ghahramani, Bayesian Monte Carlo starts with a prior over some function and makes inferences about it from a set of samples [5].

The Bayesian quadrature used by the VMBC differs from standard simple methods by the sampling method. The sampling minimizes the KL divergence and variance of the final estimate of ELBO or the evidence lowerbound. The minimizing is done by the acquisition function, which aims to minimize the uncertainty for the current posterior and for potential locations of future posteriors.[3]

3. pyVMBC examples

The aim of this chapter is to provide a comprehensive yet simple usage example of the pyVMBC package. Using this

4. Conclusions

TODO

References

[1] Bobby Huggins,Chengkun Li, Marlon Tobaben, Mikko J. Aarnos and Luigi Acerbi. Pyvbmc: Efficient bayesian inference in python, 2023. https://arxiv.org/pdf/2303.09519.

[2] Fong, E and Holmes, C.C. On the marginal likelihood and cross-validation, 2020. https://academic.oup.com/biomet/article/107/2/489/5715611.

[3] Luigi Acerbi. Variational bayesian monte carlo, 2018. https://arxiv.org/pdf/1810.05558.

[4] Luigi Acerbi. Variational bayesian monte carlo with noisy likelihoods, 2020. https://arxiv.org/pdf/2006.08655.

[5] Rasmussen, C.E. and Ghahramani, Z. Baysian monte carlo, 2002. Advances in Neural Information Processing Systems 15, https://mlg.eng.cam.ac.uk/zoubin/papers/RasGha03.pdf.

[6] Rasmussen, C.E and Williams, K.I. Gaussian processes for machine learning, 2006. https://gaussianprocess.org/gpml/chapters/RW.pdf.