Master’s Programme in Data Science

VMBC Report

Tuomas Vuontisjarvi

September 30, 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 September 30, 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

layout, summary, list of references

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

According to Acerbi (2018), 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 three 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 four 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 algorithm is a novel approximate inference method for learning about black-box models. A model is black-box model when there is no access to its inner processes. This means that the model can be viewed completely in terms of its inputs and outputs.

One way of learning about the model 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[3], statistical approximate inference methods are often used. These methods include Markov Chain Monte Carlo algorithms and variational inference.

As Acerbi notes, existing methods of approximate inference, such as above examples, generally require knowledge about the model in order to produce approximate inference[3]. When a method requires more knowledge about the model than just inputs and outputs, by definition it can't be applied to black-box models. Some methods can bypass this requirement when given a very large number of model evaluations.

A computationally expensive black-box model is a model where evaluating the model is time consuming, which means that there generally isn't access to large number of model evaluations. Therefore the existing methods for approximate Bayesian inference are unfit for computationally expensive black-box models. Expensive model is defined as one evaluation taking one second or more per evaluation[1]

The VMBC produces a flexible approximate posterior distribution of the model parameters [3]. The posterior distribution is a joint probability distribution which describes how plausible each parameter is given the observed data. The posterior is expressed as $p(x \mid \mathcal{D})$, where $\mathcal{D}$ is the dataset or the evidence and vector $x \in \mathbb{R}^D$ of the black-box model parameters.

The black-box model for the algorithm is expressed as $p(\mathcal{D} \mid x)$[3]. From the point of view of the pyVBMC, a python package for performing VMBC model and posterior inference, the black-box model is provided as Python function, which calculates target log likelihood of the black-box model[1]. This means that from user point of view, the black-box model takes in a parameter vector representing the estimation for the model parameters and calculates the likelihood of the data with the given parameters $\log p(\mathcal{D} \mid x)p(x)$.

A simple example of such log likelihood function could be a two value vector and a dataset of normally distributed values with some specific mean and variance. The function then would solve the log likelihood of seeing that particular dataset with the proposed parameters, which would stand for mean and variance.

The VMBC algorithm also computes the marginal likelihood, expressed as $p(\mathcal{D})$[3]. Fong and Holmes define marginal likelihood as a measure of model fit[2]. They also note that when the model is correctly specified, the marginal likelihood provides a method for computing the posterior probability of the model with the given data $\mathcal{D}$. Acerbi also calls marginal likelihood the model evidence[3].

By definition it is the integral over the paramater space:

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

Marginal likelihood is therefore a function, which means that is a mathematical description of the likelihoods over the whole paramater space. The VMBC algorithm uses marginal likelihood in the computation of the approximate posterior distribution[3]:

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

Thus, the computation of the algorithm has two components. Computing the marginal likelihood function and the approximate posterior distribution. $p(\mathcal{D} \mid x)$ is the likelihood of the model of interest and $p(x)$ is the prior over parameters.[3]

With the computational constraints, Acerbi suggests building a probabilistic model-based approximation of the function of interest and propose Gaussian process for the build process[3].

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]

As example Williams and Rasmussen give a set of smooth curve functions with a lot of variation, where smoothness of a function is considered a prior. As x, y points are given and a requirement for a function to intersect with those points, a posterior distribution of functions can be produced. The posterior functions still vary, but they all intersect in the given points. Intuitively this means that each data point restricts the number of functions, because all the functions that do not intersect given data points are concidered less likely. If different points are given, we have a different probability distribution for the functions, but the probable functions still include functions for all other data points.[6]

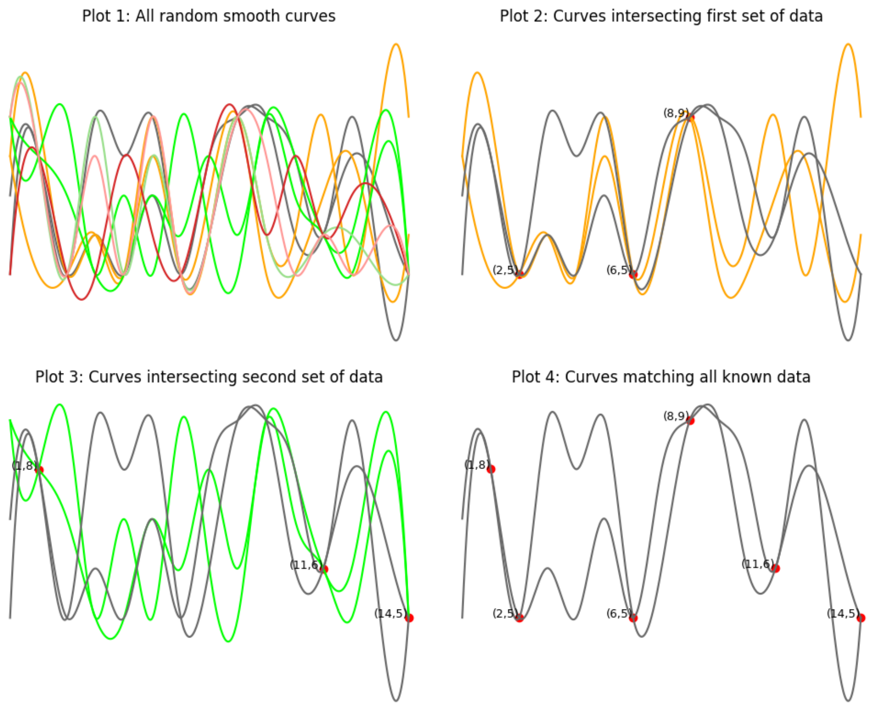

Figure 2.1: Smooth curves representing

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 uses Gaussian process based Bayesian quadrature to estimate the marginal likelihood[3]. According to Osborne *et al.*, Bayesian quadrature provides a method for numerical integration with sample efficiency. The method relies on a distribution of likelihood functions and a number of samples to provide inferences about the distribution[5]. The quadrature evaluates $f(x)$ at vector sample points and performs analytic Gaussian process to infer about the value of the integral. In the VMBC, the quadrature provides mean and variance of the integral[3]. The point of Bayesian quadrature then is to provide increasingly more accurate mean and variance of the marginal likelihood, which was the integral: $p(\mathcal{D}) = \int p(\mathcal{D} \mid x)p(x)dx$.

The VMBC algorithm is iterative and continues until a given budget of evaluations is depleted or when some termination criteria is reached. First step of the algorithm finds a set of samples that maximize given acquisition function, which are meant to guide optimal sampling. These sample points are then fed to the given log likelihood function provided by the user[3]. In other words, VMBC produces excellent guesses for the x in $p(\mathcal{D} \mid x)p(x)$ and then evaluates each guess for the black-box model.

Second step is to train a GP model based on the first step evaluations[3]. Here GP-based Bayesian quadrature is used to estimate the marginal likelihood: $p(\mathcal{D}) = \int p(\mathcal{D} \mid x)p(x)dx$. Third step is to update variational posterior approximation by optimizing the lower evidence bound[3]. In mathematical terms, this equates to solving the posterior formula: $p(x \mid \mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(\mathcal{D})}$, which is then used to guide the next iteration active sampling.

3. pyVMBC examples

TODO

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] Osborne, M.A. and Duvenaud, D. and Garnett, R. and Rasmussen, C.E. and Roberts, S.J. and Ghahramani, Z. Active learning of model evidence using bayesian quadrature, 2012. Advances in Neural Information Processing Systems 25, https://papers.nips.cc/paper_files/paper/2012/file/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Paper.pdf.

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