Universidade Federal do Rio de Janeiro

Boosted Variational Inference via Bayesian Monte Carlo

Danilo de Freitas Naiff

Rio de Janeiro

Setembro de 2019

Universidade Federal do Rio de Janeiro

Boosted Variational Inference via Bayesian Monte Carlo

Danilo de Freitas Naiff

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática, da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Orientador: Prof. Fabio Antônio Tavares Ramos.

Rio de Janeiro
Setembro de 2019

Universidade Federal do Rio de Janeiro

Boosted Variational Inference via Bayesian Monte Carlo

Danilo de Freitas Naiff

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática, da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Aprovada por:

Prof. Fabio Antônio Tavares Ramos

Prof. Heudson Tosta Mirandola

Prof. Yuri Fahham Saporito

Prof. Amaro Gomes Barreto Junior

Prof. Hugo Tremonte de Carvalho

Prof. Adriano Mauricio de Almeida Cortes

Rio de Janeiro
Setembro de 2019

Naiff, Danilo de Freitas.
Boosted Variational Inference via Bayesian Monte Carlo/ Danilo de Freitas Naiff. - Rio de Janeiro: UFRJ/ IM, 2019.
xiii,135f.: il.; 29,7 cm.
Orientador: Fabio Antônio Tavares Ramos.
Dissertação (mestrado) — Universidade Federal do Rio de Janeiro, Instituto de Matemática, Programa de Pós-Graduação em Matemática, 2019.
Referências Bibliográficas: f. 101-114.

1. Bayesian Inference. 2. Variational Inference. 3. Gaussian Processes. 4. Bayesian Monte Carlo. I. Ramos, Fabio Antônio Tavares. II. Universidade Federal do Rio de Janeiro,Programa de Pós-Graduação em Matemática. III. Título.

Dedicado a Ronaldo Lemos Naiff.

Acknowledgments

The following acknowledgments are written in portuguese

Agradeço ao meu orientador Fábio, que me deu o apoio e conhecimento necessário para desenvolver esse trabalho. Agradeço ao pessoal do LABMA, pelo companheirismo e pelas ideias que me deram. Agradeço aos meus colegas e à equipe do FMTC, competição que me ensinou muitas coisas, algumas usadas neste trabalho. De forma mais abstrata, existe toda uma comunidade que permite que idéias possam ser compartilhadas, e implementadas, com uma facilidade muito maior do que se teria em décadas passadas, e eu agradeço a ela.

Agradeço a todos os amigos que me deram suporte por todos esses anos. Em particular, aqueles que me ouviram falando sobre meus estresses durante a escrita deste texto, sobre problemas em que seus pontos de vista eram absurdamente abstratos. Agradeço a minha mãe Rose, pelo suporte durante esse tempo, assim como ao meu avô Reinaldo. Finalmente, agradeço ao meu pai Ronaldo, que nunca imaginaria que eu acabaria por fazer um mestrado em Matemática, mas que acho que não ficaria triste com isso.

Finalmente, este é um trabalho financiado por agências públicas, o que no final é dinheiro de um povo sofrido, que em sua maioria ganha muito menos do que o que me foi oferecido. Agradeço a este, e espero que o ganho total que possa dar de volta justifique este investimento.

Danilo de Freitas Naiff
Setembro de 2019

Resumo

Boosted Variational Inference via Bayesian Monte Carlo

Danilo de Freitas Naiff

Resumo da dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática, da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Resumo: A maioria dos problemas importantes em aprendizado de máquina são caros computacionalmente, em particular aqueles que envolvem inferência Bayesiana. Muitas das técnicas Bayesianas atuais requerem um alto número de avaliações de verossimilhança, o que pode ser inviável em alguns casos. Nessa dissertação, propomos um algoritmo de inferência aproximada, baseado em trabalhos recentes sobre inferência variacional e processos Gaussianos. Este algorítmo, nomeado pelo autor *Boosted Variational Bayesian Monte Carlo*, é construido de forma que poucas avaliações de verossimilhança sejam necessárias. O algoritmo, seu pacote associado, e a teoria por trás dele são apresentados neste trabalho. Tambem ilustramos o algoritmo através de sua implementação em alguns *toy examples*, e em um problema de fonte de contaminação.

Palavras–chave. Inferência Bayesiana, Inferência variacional, Processos gaussianos, Bayesian Monte Carlo.

Rio de Janeiro
Setembro de 2019

Abstract

Boosted Variational Inference via Bayesian Monte Carlo

Danilo de Freitas Naiff

Abstract da dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática, da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Abstract: Most of the important problems in machine learning are computationally expensive, in particular the ones involving Bayesian inference. Many of the current Bayesian techniques requires a large number of likelihood evaluations, which may be infeasible for some cases. In this dissertation, we propose an approximate inference algorithm, based on recent works on variational inference and Gaussian processes. This algorithm, named by the author *Boosted Variational Bayesian Monte Carlo*, is designed so that few likelihood evaluations are needed. The algorithm, its associated package, and the theory behind it are presented in this work. We also illustrate the algorithm by implementing it in some toy examples, and in a contamination source problem.

Keywords. Bayesian inference, variational inference, Gaussian processes, Bayesian Monte Carlo.

Rio de Janeiro
September 2019

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 SUMMARY

The Bayesian framework for inference and learning is conceptually simple, while having good properties related to normative reasoning [40]. However, behind its simplicity lies the computational challenge of sampling and integration, which drives much of research on the field. Methods such as Markov Chain Monte Carlo and variational inference have undergone major advances in the recent decades, driving Bayesian methods back to popularity.

In general those methods assume that evaluation of a likelihood function $l(\theta) = p(\mathcal{D}|\theta)$ is computationally cheap enough to be done tens or hundred thousand of times, at least. However, in some cases this may not be true, as for example in cases where $l(\theta)$ must come from a computationally expensive simulation. This work presents a variational approximation method, using Gaussian process regression, that is adapted from [2, 45], in order to deal with cases where θ is a continuous random variable, and $l(\theta)$ can be evaluated only tens, hundreds, or thousands of times. The current algorithm is suited for low and medium dimensional problems (around 10 dimensions at most).

A Python package for deploying this method was developed, built mainly on top of PyTorch, and we test it in some cases. Currently, the package lacks complete documentation and unit testing, but an alpha version is already available in `https://github.com/DFNaiff/BVBMC`. Since this package may undergo changes in the future, all code used in the present work may be found in `https://github.com/DFNaiff/Dissertation`.

Next, we presented a brief summary of each chapter of this dissertation.

In Chapter 2, Bayesian theory is briefly reviewed, along with some approximate inference methods. Moreover, approaches to expensive or intractable likelihoods are discussed.

In Chapter 3, the Gaussian process regression method is reviewed, while in Chapter 4, Bayesian Monte Carlo, a Gaussian process based integration method, is discussed.

In Chapter 5, variational inference is reviewed, first in general, then focusing on the case where approximation is made by mixtures of Gaussian distributions, using the boosting approach found in [45]. Moreover, a recent method presented in [2], using variational inference via Bayesian Monte Carlo, called Variational Bayesian Monte Carlo, is presented.

In Chapter 6, we propose an adaptation of Variational Bayesian Monte Carlo, incorporating other ideas presented in chapter 2 and 3, particularly the boosting approach in [45]. This results in a new algorithm, which we call Boosted Variational Bayesian Monte Carlo. A small discussion on implementation follows, focused on backpropagation, which is the reason the developed package was built on PyTorch. In Chapter 7, the corresponding Python package is applied in a few toy examples, and in a contamination source problem.

The work is concluded in Chapter 8, where we discuss future directions for both scaling the presented method to higher dimension and to increase its accuracy. In the appendix, among other things, there is an extended discussion on Sparse Gaussian Process, a technique that the author tried to use in order to extend the present method to higher dimensions and greater number of evaluations, although with limited success.

2 BAYESIAN INFERENCE AND LEARNING

2.1 Learning described as Bayesian inference

The central problem of learning from data can be verbalized as: given that some agent have access to data $\mathcal{D}$, what knowledge can the learner extract from it? One way to approach this problem is by assuming that learning does not take place in a vacuum, but in a world that the learner has uncertain knowledge about, translated into beliefs. The fact that those beliefs are uncertain is important, given that if the learner knew exactly everything he should know about the world, access to data $\mathcal{D}$ could not teach him nothing more.

Given this general framework of "informed uncertainty", one natural way to describe it, mathematically, is by using probability theory for describing the problem. More specifically, it is used the *Bayesian* viewpoint of probability, where degrees of uncertainty about quantities are mapped into probabilities about those [66, 55].

In this interpretation, probability theory does not just deal with random events, but with anything that an agent is uncertain about. So, given some proposition A, and given that the learner knows I about the world, $P(A|I)$ represents what he knows about A. Thus $A|I$ becomes a random variable, *even if A is not a random event.* Cox's theorem [55],[25] says that, under some certain common sense assumptions about how the learner should ideally reason about beliefs, the rules of probability theory holds, as an extension to logic.

In a simplification of this setting, when learning from the data $\mathcal{D}$, and pre-

vious information I, the learner must have some set of hypothesis (assuming finite for now) $\mathcal{H} = \{H_1, \ldots, H_t\}$, such that the learner assumes one and only one of then is true, and with each of them associated with a probability $P(H_k)$ such that

$$\sum_{H_k \in \mathcal{H}} P(H_k|I) = 1.$$

Furthermore, each hypothesis H_k should say something about how likely it is for the data to be generated, given H_k is true, and this information is encoded in $P(\mathcal{D}|H_k, I)$. In this case, Bayes' theorem says that it is possible to obtain the updated probabilities (thus degree of beliefs) $P(H_k|\mathcal{D}, I)$ by Bayes' rule:

$$P(H_k|\mathcal{D}, I) = \frac{P(\mathcal{D}|H_k, I)}{P(\mathcal{D}|I)} P(H_k|I), \tag{2.1}$$

with $p(\mathcal{D}|I)$ being available by marginalization

$$P(\mathcal{D}|I) = \sum_{H_k \in \mathcal{H}} P(\mathcal{D}|H_k, I) P(H_k|I). \tag{2.2}$$

In practice, usually hypothesis does not comes in discrete chunks, but one assumes a *model* M. The model is usually endowed with free parameters $\theta \in \Theta \subset \mathbb{R}^D$, so that, assuming those to be continuous[1], the hypotheses are encoded by those parameter through a probability density function $p(\theta|M)$. Then, given the data $\mathcal{D}$, one seeks the posterior density function $p(\theta|\mathcal{D}, M)$. In this case, we also have a version of Bayes' theorem for the densities

$$p(\theta|\mathcal{D}, M) = \frac{p(\mathcal{D}|\theta, M)}{p(\mathcal{D}|M)} p(\theta|M), \tag{2.3}$$

with

$$p(\mathcal{D}|M) = \int_{\Theta} p(\mathcal{D}|\theta', M) p(\theta'|M) d\theta'. \tag{2.4}$$

In the Bayesian framework, the problem of learning is reduced to one of inference about θ, in a manner that if a specific parameter θ is sufficient to make a prediction $Q|\theta, M$, with density $p(Q|\theta, M)$, then the learner has access to $Q|\mathcal{D}, M$

[1]This is not necessary at all, but it simplifies the notation

by marginalization

$$p(Q|\mathcal{D}, M) = \int_{\Theta} p(Q|\theta, M)p(\theta|\mathcal{D}, M)d\theta. \tag{2.5}$$

Thus, there is no fundamental difference between learning and inference from a Bayesian point of view.

2.2 Decision theory

Following the Bayesian procedure, an agent can learn something about the world. However, ultimately what one wants to do with beliefs about the world is to convert those into actions. This can be formalized in *Bayesian decision theory* [91], where the components required for belief updating are combined with a *loss function* $L : \Theta \times \mathcal{A} \to \mathbb{R}^+$, with $L(\theta, a)$ being the cost of taking action a when the state of the world is θ [2]. Then, the action that minimizes the expected loss, given the posterior distribution $p(\theta|\mathcal{D}, M)$,

$$a^* = \arg\min_{a \in \mathcal{A}} \int_{\Theta} L(\theta, a)p(\theta|\mathcal{D}, M)d\theta, \tag{2.6}$$

is the Bayes-optimal decision for the agent to make [91].

From this point of view, parameter estimation is simply when the action taken is choosing a parameter, that is, $\mathcal{A} = \Theta$. In this setting, we have that, for some loss functions $L(\theta, \tilde{\theta})$, one can find the desired *Bayes estimator*,

$$\hat{\theta} = \arg\min_{\tilde{\theta}} \int L(\theta, \tilde{\theta})p(\theta|\mathcal{D}, M)d\theta, \tag{2.7}$$

by calculating the minimum analytically:

- The l_2 (quadratic) loss $L(\theta, \tilde{\theta}) = ||\theta - \tilde{\theta}||_2^2$, for which $\hat{\theta} = \mathbb{E}[\theta|\mathcal{D}, M]$

[2] This language refers back to the continuously parameterized model setting described above, and this will be assumed through the text. However, one can refer also to a more general setting, *mutatis mutandis*

- The l_1 (absolute) loss $L(\theta, \tilde{\theta}) = ||\theta - \tilde{\theta}||_1$, for which, at each coordinate i, $\hat{\theta}_i = \text{median}(\theta_i|\mathcal{D}, M)$.

However, loss functions can be much more general, allowing, for example, to encode asymmetry in the gravity of mistakes. For instance, if θ is the maximum load of some structural component, underestimating it may result in a bigger waste of resources, while overestimating it may result in collapse.

One way of choosing $\hat{\theta}$ that does not exactly enter the framework above, at least for continuous parameters, are either by the *maximum a posteriori* MAP of the probability density function

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|\mathcal{D}, M) = \arg\max_{\theta} p(\mathcal{D}|\theta, M)p(\theta|M). \tag{2.8}$$

The MAP estimation can be regarded however as a limit of minimizers of loss functions of the form

$$L_c(\theta, \tilde{\theta}) = \begin{cases} 0, & \text{if } ||\theta - \tilde{\theta}|| < c \\ 1, & \text{otherwise,} \end{cases} \tag{2.9}$$

so that [3]

$$\hat{\theta}_{\text{MAP}} = \lim_{c \to 0} \arg\min_{\tilde{\theta}} \int L_c(\theta, \tilde{\theta}) p(\theta|\mathcal{D}, M) d\theta \tag{2.10}$$

Related to the MAP estimator is the *maximum likelihood estimate* (MLE), which can be seem as a modification of the MAP that doesn't take in account prior belief

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} p(\mathcal{D}|\theta, M). \tag{2.11}$$

The MAP (and MLE) estimation suffers from some drawbacks for continuous distributions:

- The MAP estimation does not take in account any true loss function, just limits

[3] Provided some conditions. See [8] for a counterexample of the general result.

of loss functions. Specially in applications that has an intrinsic asymmetric loss, this may result in grave mistakes.

- The MAP estimation is in general an untypical point for the distribution, in the sense that the probability of the parameter to be near the MAP is low. In particular, in high dimensions that MAP will be very untypical (see [9]).

- The MAP estimation is not invariant under reparameterizations. To illustrate, assume that θ is a one-dimensional parameter representing some phenomena F, and let $\phi = g^{-1}(\theta)$, where g is diffeomorphic [4]. Clearly, ϕ is also a valid parameterization of F. Assuming $g'(\phi) > 0$ for simplicity, let $f_\theta(\theta) := p(\theta|\mathcal{D}, M)$. Then, we have, letting $f_\phi(\phi) := p(\phi|\mathcal{D}, M)$, that $f_\phi(\phi) = f_\theta(g(\phi))g'(\phi)$. This implies that:

$$f'(\phi) = g''(\phi)f_\theta(g(\phi)) + (g'(\phi))^2 f'_\theta(g(\phi)). \tag{2.12}$$

  Now, being $\hat{\theta}_{\text{MAP}}$ the MAP estimator for θ, we have $f'_\theta(\hat{\theta}_{\text{MAP}}) = 0$. However, letting $\hat{\phi} := g^{-1}(\hat{\theta}_{\text{MAP}})$, we have that $f'(\hat{\phi}) = g''(\hat{\phi})f_\theta(g(\hat{\phi}))$, which does not equal to 0 unless $g''(\hat{\phi}) = 0$. Hence, $\hat{\phi}$ cannot be the MAP estimator for ϕ. But, since θ and ϕ are both valid parameterizations for phenomena F, this lack of invariance implies that the MAP estimation has no meaning in estimating F.

Still, the MAP estimator is relatively straightforward to calculate, since it requires the optimization of $p(\mathcal{D}|\theta)p(\theta)$, which is in general a simpler problem than integration. Thus, it is widely used.

[4] In order to exclude some special conditions, assume both θ and ϕ are supported in $\mathbb{R}$

2.3 Model selection

In the previous discussion, the model M was assumed to be fixed, with only its parameters being unknown. In practice, we have a set of models $\mathcal{M}$ from which we choose M. This raises the question on how to make this choice of M.

The standard Bayesian solution for the problem would be placing a prior distribution $P(M)$ for the models, and then computing the posterior distribution for them, given the data, by Bayes' rule

$$P(M|\mathcal{D}) = \frac{P(\mathcal{D}|M)P(M)}{\sum_{M' \in \mathcal{M}} P(\mathcal{D}|M')P(M')}, \tag{2.13}$$

with the model likelihood, in this setting being called *marginal likelihood* or *evidence*, given by

$$p(\mathcal{D}|M) = \int_{\Theta_M} p(\mathcal{D}|\theta_M, M)p(\theta_M|M)d\theta_M, \tag{2.14}$$

emphasizing that the parameter space Θ_M depends on the model. Then, one can choose M by MAP estimation

$$\hat{M} = \arg\max_{M \in \mathcal{M}} p(\mathcal{D}|M)p(M), \tag{2.15}$$

or, in prediction settings, carrying the full posterior model distribution for doing model averaging. Still, choosing a prior for models may not be a trivial task, as discussed in [91]. To circumvent this, one can instead forget about the prior (or assume an uniform prior), and choose the model with maximum likelihood

$$\hat{M} = \arg\max_{M \in \mathcal{M}} p(\mathcal{D}|M) = \arg\max_{M \in \mathcal{M}} \int p(\mathcal{D}|\theta_M, M)p(\theta_M|M)d\theta_M. \tag{2.16}$$

The choice of models by maximization of evidence results in the *Bayesian Occam's razor* [66, 67, 88], named after the Occam's razor principle that says, given a choice between models, we should select the simplest models that still explains the data. We say that some model is simpler or more complex than another if it can

explain few or more data. To see how Occam's razor works in Bayesian setting, it suffices to realize that, between all possible datasets, probabilities must sum to one. For illustration, assume that $\mathcal{D}$ comes from a finite set of possible datasets. Then, we need

$$\sum_{\mathcal{D}'} p(\mathcal{D}'|M) = 1. \tag{2.17}$$

Now, compare three models, M_1, M_2 and M_3. M_1 can explain only very few datasets well, so few that it cannot explain $\mathcal{D}$. M_2 can explain more datasets, including $\mathcal{D}$, but not so much as M_3 explains, which is a vast number of datasets. We have then that $p(\mathcal{D}|M_1)$ must be very low, given that M_1 does not explain $\mathcal{D}$. The two other models, that explains the data, have higher values of $p(\mathcal{D}|M_2)$ and $p(\mathcal{D}|M_3)$. But, since $p(\mathcal{D}|M_3)$ "shares" probability mass with more datasets than $p(\mathcal{D}|M_2)$, by conservation of probability mass, we find that $p(\mathcal{D}|M_2)$ is higher. Hence we have the order

$$p(\mathcal{D}|M_2) > p(\mathcal{D}|M_3) > p(\mathcal{D}|M_1). \tag{2.18}$$

Hence, we find that M_2 is simple enough to be desirable, but not so simple as to not be able to explain $\mathcal{D}$, thus obeying the Occam's razor principle.

The model set $\mathcal{M}$ itself does not need to be discrete or enumerable. If $\mathcal{M}$ can be parametrized by a set Λ, then one can change $\mathcal{M}$ for Λ, and find the maximum of evidence by:

$$\lambda_{ML-II} = \arg\max_{\lambda \in \Lambda} p(\mathcal{D}|M(\lambda)). \tag{2.19}$$

This estimator is called *type II maximum likelihood* estimator. By setting an prior over Λ, we would have instead a *type II maximum a posteriori* estimator

$$\lambda_{MAP-II} = \arg\max_{\lambda \in \Lambda} p(\mathcal{D}|M(\lambda))p(M(\lambda)). \tag{2.20}$$

2.4 Approximate inference

Computationally, Bayesian inference suffers from two major issues:

- Because in the posterior density (2.3), the normalizing term $p(\mathcal{D})$ [5] is to be determined by the integral (2.4), a closed-form solution of the posterior density is often unavailable, even though the unnormalized density $Zp(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)p(\theta) = p(\theta, \mathcal{D})$ usually is.

- A more grave problem is that, even with the normalized posterior density at hand, for an arbitrary function $f(\theta)$, the expectation $\int f(\theta)p(\theta|\mathcal{D})d\theta$ is not trivial to calculate. And, as seen in Section 2.2, what one wants in the end with posterior distribution is to calculate expectations. Thus, computational methods for dealing with those problems are needed.

In this section, Monte Carlo methods are quickly reviewed, along with Laplace's approximation. Discussion on variational inference, another important approximate method, is postponed to Chapter 3, since it is a main subject of this work.

2.4.1 Monte Carlo integration methods

Consider back the expectation

$$\mu := \mathbb{E}_{\theta\sim p(\theta|\mathcal{D})}[f(\theta)] = \int f(\theta)p(\theta|\mathcal{D})d\theta. \tag{2.21}$$

[5] From now on we omit the dependence on the model M.

Assuming this expectation exists, if one can sample $\theta_1, \ldots, \theta_N$ from $\theta|\mathcal{D}$, independently, then the estimator

$$\hat{\mu} := \frac{1}{N}\sum_{i=1}^{N} f(\theta_i), \tag{2.22}$$

is such that, as $N \to \infty$, $\hat{\mu} \to \mu$ almost surely, by the law of large numbers. Moreover, if the variance of $f(\theta)$ is finite, then the convergence rate is $\mathcal{O}\left(\sqrt{\frac{\mathrm{Var}(f(\theta))}{N}}\right)$, by the central limit theorem. Hence, the challenge of Monte Carlo methods is how to get, from an unnormalized posterior distribution $p(\theta, \mathcal{D})$, independent or "independent enough" samples from this distribution.

2.4.1.1 Importance sampling

The importance sampling algorithm [92] is a relatively simple algorithm for sampling from unnormalized posteriors. Let $q(\theta)$ be some proposal distribution, such that one can sample easily from $q(\theta)$, having samples $\theta_1, \ldots, \theta_N \sim q(\theta)$. Finally, assume an unnormalized density $\bar{q}(\theta) = Z_q q(\theta)$ is known. Then, rewrite (2.21) as

$$\hat{\mu} = \int f(\theta)p(\theta|\mathcal{D})d\theta = \frac{Z_q}{Z}\int f(\theta)\frac{p(\theta,\mathcal{D})}{\bar{q}(\theta)}q(\theta)d\theta, \tag{2.23}$$

which can be estimated as

$$\frac{Z_q}{Z}\int f(\theta)\frac{p(\theta,\mathcal{D})}{\bar{q}(\theta)}q(\theta)d\theta \approx \frac{1}{Z/Z_q}\frac{1}{N}\sum_{i=1}^{N}\tilde{w}_i f(\theta_i), \quad \tilde{w}_i := \frac{p(\theta_i,\mathcal{D})}{\bar{q}(\theta_i)}. \tag{2.24}$$

The ratio Z/Z_q can himself be estimated, using the same samples, as

$$\frac{Z}{Z_q} = \frac{1}{Z_q}\int p(\theta,\mathcal{D})d\theta = \int \frac{p(\theta,\mathcal{D})}{\bar{q}(\theta)}q(\theta)d\theta \approx \frac{1}{N}\sum_{i=1}^{N}\tilde{w}_i. \tag{2.25}$$

Then, joining (2.24) and (2.25), we have an estimate for (2.21)

$$\hat{\mu} = \int f(\theta)p(\theta|\mathcal{D})d\theta \approx \sum_{i=1}^{N} w_i f(\theta_i), \quad w_i = \frac{\tilde{w}_i}{\sum_{j=1}^{N}\tilde{w}_j}, \forall i. \tag{2.26}$$

2.4.1.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods uses Markov chains to sample from the desired distribution [92, 20]. MCMC is arguably the most popular method in Bayesian statistics, due to their ability to sample efficiently from relatively high dimensional distributions, with only unnormalized density available. As such, the number of MCMC methods is enormous. Here a basic method, Metropolis-Hastings, is reviewed in passing.

Markov chains are sequences of random variables $X_0, X_1, \ldots$, with the property that the conditional distribution $X_i|X_0, \ldots, X_{i-1}$ is the same as $X_i|X_{i-1}$. Thus, a Markov chain is completely defined by the distribution of the initial random variable X_0, and the *transition probability distribution*, $p(x_{i+1}|x_i)$. If $p(x_{i+1}|x_i)$ is independent of i, is called a *stationary transition*, and those kinds of chains are of most interest in MCMC methods.

Markov chains becomes interesting when their transitions have some unique distribution $\pi(x)$, called a *stationary distribution*, such that

$$\pi(x') = \int p(x'|x)\pi(x)dx, \tag{2.27}$$

that is, when the initial random variable X_0 of a Markov chain is distributed according to $\pi(x)$, under the transition $p(x'|x)$, every X_i is distributed according to $pi(x)$. One of the conditions that suffices (although is not necessary)for $\pi(x)$ being a stationary distribution is that it satisfies the *detailed balance condition*

$$p(x'|x)\pi(x) = p(x|x')\pi(x'). \tag{2.28}$$

With the stationary distribution, under some technical conditions (see [92]), we have for a Markov chain with transition probability $p(x'|x)$, such that $X_0 = x_0$ and x_i is

sampled from $p(x_i|x_{i-1})$, for $i \geq 1$, that, as $N \to \infty$,

$$\frac{1}{N}\sum_{i=1}^{N} f(x) \to \mathbb{E}_{X\sim\pi}[f(X)]. \tag{2.29}$$

Moreover, the convergence follows a version of the central limit theorem. Assume first that $X_0 \sim \pi$. Then, we have that the central limit theorem holds, with (**??**) being substituted for

$$\mathcal{O}\left(\sqrt{\frac{\mathrm{Var}(f(X_1)) + 2\sum_{k=1}^{\infty}\mathrm{Cov}(f(X_1), f(X_{1+k}))}{N}}\right), \tag{2.30}$$

when i large enough [38]. More generally (and realistically), any Markov chain (modulo a technical conditions) for which the transition $p(x'|x)$ has stationary distribution $\pi(x)$ follows the central limit theorem, with asymptotic rate of convergence being the same as when $X_0 \sim \pi$. [6]

These results gives rise to the following procedure: construct a randomized algorithm such that, starting with some $\theta_i \in \Theta$, $\theta_{i+1} \in \Theta$ is generated such that $p(\theta_{i+1}|\theta_i)$ has stationary distribution $p(\theta|\mathcal{D})$.

The Metropolis-Hastings algorithm is one manner of doing this for a general distribution. Given a (possibly unnormalized) conditional distribution $g(\theta'|\theta)$ (a *proposal distribution*), and θ_t, $t \geq 0$, one samples a proposal $\tilde{\theta}_{t+1}$ from $g(\theta'|\theta_t)$, and then, letting

$$\alpha(\tilde{\theta}_{t+1}, \theta_t) = \min\left(1, \frac{p(\tilde{\theta}_{t+1}|\mathcal{D})g(\theta_t|\tilde{\theta}_{t+1})}{p(\theta_t|\mathcal{D})g(\tilde{\theta}_{t+1}|\theta_t)}\right), \tag{2.31}$$

be the *acceptance probability*, then let $\theta_{t+1} = \tilde{\theta}_{t+1}$ with probability $\alpha(\tilde{\theta}_{t+1}, \theta_t)$, else let $\theta_{t+1} = \theta_t$. One key observation is that the ratio $p(\tilde{\theta}_{t+1}|\mathcal{D})/p(\theta_t|\mathcal{D})$ is independent of the normalization constant, thus it can be substituted for $p(\mathcal{D}|\tilde{\theta}_{t+1})p(\tilde{\theta}_{t+1})/p(\mathcal{D}|\theta_t)p(\theta_t)$,

[6] This results in the important notion of *effective sample size* (ESS), where (2.30) is substituted for $\mathcal{O}\left(\sqrt{\frac{\mathrm{Var}(f(X))}{N_{\mathrm{eff}}}}\right)$, with $N_{\mathrm{eff}} = N/\left(1 + 2\sum_{k=1}^{\infty}\mathrm{corr}(f(X_i), f(X_{i+k})\right)$, when i is large enough.

avoiding the need for the normalized posterior.

For continuous distributions, one standard proposal distribution is $g(\theta'|\theta) = \mathcal{N}(\theta'|\theta, \epsilon^2 I)$, with ϵ being the step size, resulting in the *Random Walk Metropolis* algorithm. In particular, with this proposal distribution $g(\theta'|\theta) = g(\theta'|\theta)$, simplifying the acceptance probability to

$$\alpha(\tilde{\theta}_{t+1}, \theta_t) = \min\left(1, \frac{p(\tilde{\theta}_{t+1}|\mathcal{D})}{p(\theta_t|\mathcal{D})}\right), \tag{2.32}$$

2.4.2 Laplace's approximation

Laplace's approximation [12] is arguably the simplest technique from a class of methods that tries to approximate the density $p(\theta|\mathcal{D})$ by some other density $q(\theta)$, using $p(\theta, \mathcal{D})$, and work with the approximation. Another technique that belongs to this class of methods is variational inference, the subject of Chapter 5.

Consider $\Theta = \mathbb{R}^D$, $p(\theta|\mathcal{D})$ to be smooth, and $\theta^* = \hat{\theta}_{\text{MAP}}$ be the MAP of $p(\theta|\mathcal{D}) = p(\theta, \mathcal{D})/Z$. Then, doing a second order Taylor approximation on $l(\theta) = \log p(\theta, \mathcal{D}) = \log p(\theta|\mathcal{D}) + \log p(\mathcal{D})$, around θ^*, and noticing $\nabla_\theta l(\theta^*) = 0$,

$$l(\theta) \approx l(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H_{\theta^*}(l)(\theta - \theta^*), \tag{2.33}$$

where $H_{\theta^*}(l)$ is the Hessian matrix of $l(\theta)$ on θ^* [7]. Then, letting $\Sigma = -H_{\theta^*}^{-1}(l)$ and $\mu = \theta^*$, taking the exponential on both sides

$$p(\theta, \mathcal{D}) \approx \exp(l(\theta^*)) \exp\left(-\frac{1}{2}(\theta - \mu)^T \Sigma^{-1}(\theta - \mu)\right). \tag{2.34}$$

The second side is just the unnormalized density of $\mathcal{N}(\theta|\mu, \Sigma)$. Hence, normalizing back, we arrive at the Laplace's approximation for $p(\theta|\mathcal{D})$

$$p(\theta|\mathcal{D}) \approx \mathcal{N}\big(\theta; \theta^*, -H_{\theta^*}^{-1}(l)\big). \tag{2.35}$$

[7] The negative of the Hessian matrix $-H_\theta(l)$ is the same as Fisher information matrix $I(\theta)$.

The Laplace' approximation needs optimization of $l(\theta)$ and access to second derivatives, which for many cases may be cheaply available. However, since the approximation is only local, it may diverge sharply from the actual posterior. Moreover, in higher dimensions, calculating and inverting the Hessian matrix may be too costly.

2.4.2.1 Remark on approximation

It is important to consider a possible advantage of using an approximate density $q(\theta)$ over sampling from $p(\theta|\mathcal{D})$. Assume $q(\theta)$ simple enough so that there is no need for advanced sampling techniques for Markov Chain Monte Carlo. Then, consider the general loss minimization problem (2.6), and substitute $p(\theta|\mathcal{D})$ for $q(\theta)$, yielding the minimization objective for $a \in \mathcal{A}$.

$$F_q(a) = \int_\Theta L(\theta, a) q(\theta) d\theta. \tag{2.36}$$

If $\mathcal{A}$ is a subset of $\mathbb{R}^k$, then one can sample N samples from $q(\theta)$ and get a stochastic estimation of $\nabla F_q(a)$

$$\nabla F_q(a) \approx \frac{1}{N} \sum_{\theta_i \sim q(\theta)} \nabla_a L(\theta_i, a), \tag{2.37}$$

allowing the application of stochastic gradient descent, and related techniques, for minimizing $F_q(a)$. However, sampling from $q(\theta)$ is easy, so the bottleneck mostly belongs in the evaluation of $\nabla_a L(\theta, a)$. If samples from $p(\theta|\mathcal{D})$ were made from some more advanced sampling technique instead, the bottleneck would be in the sampling algorithm performance, which may be not quite as fast. Since $q(\theta)$ must be discovered only once, then approximation may be more feasible.

The drawback is that of course $q(\theta)$ must be a good approximation of $p(\theta|\mathcal{D})$ to begin with, which may not be easy to ensure (as discussed, this is one drawback from Laplace's approximation).

2.5 Expensive and intractable likelihoods

In general, Monte Carlo techniques assumes that $p(\theta, \mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$ can be evaluated cheaply. Since usually the prior $p(\theta)$ is chosen in a manner that it is very simple, whether $p(\theta, \mathcal{D})$ is hard to evaluate depends on $p(\mathcal{D}|\theta)$. In many cases, likelihood evaluation is in fact cheap, but in some cases it may be expensive or intractable, requiring specific techniques for approximate inference.

2.5.1 Pseudo-marginals

One case of intractable likelihood is when the likelihood model depends on some unobserved variable, that must be marginalize. To illustrate, consider that $\mathcal{D} = y_1$ is a noisy observation of a phenomena z_1, whose dependence on a parameter θ is modeled as $p(z_1|\theta)$. The noise model $p(y_1|z_1)$ is also available. Then, the likelihood $p(y_1|\theta)$ comes from marginalization

$$p(y_1|\theta) = \int p(y_1|z_1)p(z_1|\theta)d\theta.$$

As a general case, consider a likelihood dependent on a latent variable

$$p(\mathcal{D}|\theta) = \int p(\mathcal{D}|\omega, \theta)p(\omega|\theta)d\omega. \tag{2.38}$$

Assume the integral in (2.38) is not available analytically, hence so is not $p(\mathcal{D}|\theta)$. Usually what is available are Monte Carlo estimates of $p(\mathcal{D}|\theta)$, say by i.i.d. samples of $w|\theta$,

$$\hat{p}(\mathcal{D}|\theta) = \frac{1}{N} \sum_{\omega_i \sim p(\omega|\theta)} p(\mathcal{D}|\omega_i, \theta), \tag{2.39}$$

or by importance sampling. In this case, [5] shows that when using an unbiased and positive estimate $\hat{p}(\mathcal{D}|\theta)$ at each step of the Metropolis-Hastings algorithm, resulting in an unbiased estimate of the unnormalized posterior $\hat{p}(\mathcal{D}|\theta)p(\theta)$, the

resulting stationary distribution is $p(\mathcal{D}|\theta)$. The result does not give an answer on whether Metropolis-Hastings using $\hat{p}(\mathcal{D}|\theta)$ is efficient, and how to make so. This itself is a current topic of research (some examples can be found in [4, 100]).

2.5.2 Approximate Bayesian computation

Now consider a model that $p(\mathcal{D}|\theta)$ is not readily available, but for each fixed $\theta \in \Theta$, one can sample the *random variable* $\mathcal{D}|\theta$ with ease. For clarity, we will refer to this random variable as $\mathcal{D}'|\theta$, while keeping $\mathcal{D}$ denoting the fixed data.

As an example, consider the data $\mathcal{D}$ consists of observation point x_N of a long Markov chain, with known transition probability distribution $p(x_{i+1}|x_i)$, and one wants to infer the point x_0 where the chain was initiated. The likelihood $p(x_N|x_0)$ is given by

$$p(x_N|x_0) = \int \dots \int p(x_N|x_{N-1}) \dots p(x_1|x_0) dx_1 \dots dx_{N-1}, \tag{2.40}$$

which is hard to even compute some pseudo-marginal. However, given some x_0, sampling x_N is just a question of simulating the chain for N steps, with transition $p(x_{i+1}|x_i)$. Some other examples of models whose likelihood is hard to evaluate, but sampling is easy, are found in evolutionary genetics [83, 1].

In approximate Bayesian computation (ABC) [30, 1], one wishes to construct an artificial likelihood $p_{\text{ABC}}(\mathcal{D}|\theta)$, in such a way that for each θ, when the simulated data $\mathcal{D}'|\theta$ is "similar" to $\mathcal{D}$, $p_{\text{ABC}}(\mathcal{D}|\theta)$ is higher than when $\mathcal{D}'|\theta$ is not. For doing this, one takes:

- a function S that takes a (simulated or real) dataset $\mathcal{D}$ and return some d-dimensional statistics of it. For example, if $\mathcal{D} = \{y_1, \dots, y_N\}$, the statistics

may be the first d empirical moments of $\mathcal{D}$, or simply $\mathcal{D}$, making S the identity function (in this case $d = N$).

- A function $k : \mathbb{R}^d \to \mathbb{R}$, integrating to one, such that k achieves its maximum at 0. For instance, $k(x) = \mathcal{N}(x; 0, h^2 I)$ can be used.

With those, one defines the ABC approximation for the likelihood

$$p_{\text{ABC}}(\mathcal{D}|\theta) = \int \frac{1}{h} k\left(\frac{S(\mathcal{D}) - S(\mathcal{D}')}{h}\right) p(\mathcal{D}'|\theta) d\mathcal{D}'. \tag{2.41}$$

To see why this is an approximation of the true likelihood $p(\mathcal{D}|\theta)$, assume that $S(\mathcal{D}) = S(\mathcal{D}')$ if and only if $\mathcal{D} = \mathcal{D}'$, and consider $k(x) = \mathcal{N}(x; 0, h^2 I)$. Then, as $h \to 0$, $h^{-1}k((S(\mathcal{D}) - S(\mathcal{D}'))/h)$ goes to the Dirac delta function $\delta(\mathcal{D} - \mathcal{D}')$. But, we have that

$$\int \delta(\mathcal{D} - \mathcal{D}') p(\mathcal{D}'|\theta) d\mathcal{D}' = p(\mathcal{D}|\theta) \tag{2.42}$$

, so $p_{\text{ABC}}(\mathcal{D}|\theta)$ goes to $p(\mathcal{D}|\theta)$ when h goes to 0.

With the approximate likelihood (2.41), one has the corresponding approximate ABC posterior

$$p_{\text{ABC}}(\theta|\mathcal{D}) \propto p_{\text{ABC}}(\mathcal{D}|\theta) p(\theta). \tag{2.43}$$

Notice the ABC likelihood still is not analytically available. However, since samples from $p(\mathcal{D}'|\theta)$ are available, one can use the pseudo-marginal technique presented in previous section to sample from the approximate ABC posterior. The question on how to choose appropriate summary statistics is addressed for example in [30].

2.5.3 Expensive likelihoods

In some cases, the likelihood $p(\mathcal{D}|\theta)$ is expensive to evaluate but not intractable, such that one can have tens or hundreds of evaluations in limited time,

but not much more. Moreover, unlike the previously presented case, sampling from the model is just as expensive, if not more, than evaluating $p(\mathcal{D}|\theta)$. Such likelihoods arise, for example, in Bayesian inverse problems [108], where the mapping from parameters to observations is done by expensive simulations.

An approach is, given a limited number of likelihood evaluations $\Omega_N = \{(\theta_i, p(\mathcal{D}|\theta_i)\}$, construct an approximate model $\hat{p}_N(\theta|\mathcal{D})$ of $p(\theta|\mathcal{D})$, and inference is performed with the approximation, usually with MCMC. This model should, given new evaluations of the likelihood $\Omega_{N'}$, be able to incorporate those in an online manner.

Gaussian processes, presented in next chapter, are particularly suitable for this task, and are used in [89, 114, 10, 59, 24], using Monte Carlo methods on the approximation. Other approximations include GRIMA [15] and polynomial approximations [68]. The work presented here falls in the contest of expensive likelihoods methods, using Gaussian processes for approximation, and variational inference for approximate inference, as in [2].

3 GAUSSIAN PROCESSES

3.1 Parametric and nonparametric regression

Consider the standard regression problem: given $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, with $x_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$, one wants, for a $x_* \in \mathcal{X}$, find $p(y_*|x_*)$. If we assume a model M for $y|x$, for all $x \in \mathcal{X}$, parameterized by θ, then by the Bayesian view one should marginalize over θ resulting in

$$p(y_*|x_*, M, \mathcal{D}) = \int_\Theta p(y_*|x_*, \theta, M, \mathcal{D}) p(\theta|\mathcal{D}) d\theta, \tag{3.1}$$

reducing the problem to one of posterior inference in θ, discussed in the previous chapter (equation (2.5)).

Since the model M is parameterized, in a sense the model will be always limited, since there cannot be a mapping from finite parameters to all distributions. Nonparametric models by contrast are models that cannot be parameterized by a finite set of parameters, and Bayesian nonparametric regression is when a nonparametric model is used in the Bayesian setting [39, 49].

3.2 Gaussian process regression

Bayesian nonparametric regression seems to be an impossible task, since it requires working with distributions in infinite-dimensional spaces. However, Gaussian process regression [87] does this, by choosing a suitable model, given by a Gaussian process

Definition 3.2.1. A *Gaussian process* (GP) is a distribution over the space of func-

tions from $\mathcal{X}$ to $\mathbb{R}$ such that, for each $\mathbf{x} = (x_1, ..., x_N) \in \mathcal{X}^N$, $\mathbf{f} := (f(x_1), ..., f(x_N))$ follows a multivariate normal distribution [87].

A GP is completely specified by a *mean function*

$$m : \mathcal{X} \to \mathbb{R}$$

and a *covariance function*

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R},$$

in such a way that, for $\mathbf{x} = (x_1, \ldots, x_N)$, letting

$$\mathbf{m}(\mathbf{x}) := (m(x_1), ..., m(x_N))^T \quad K(\mathbf{x}, \mathbf{x}') = \left(k(x_i, x'_j) \right)_{i,j}, \tag{3.2}$$

then $f(\mathbf{x}) \sim \mathcal{N}(\mathbf{m}(\mathbf{x}), K(\mathbf{x}, \mathbf{x}))$.

Notice that this requires the function k to be a *positive-semidefinite* (PSD) function, that is, for any $N \in \mathbb{N}$, for any $\mathbf{x} = (x_1, ..., x_N) \in \mathcal{X}^N$, the matrix $K(\mathbf{x}, \mathbf{x})$ defined by $(K(\mathbf{x}, \mathbf{x}))_{i,j} = k(x_i, x_j)$ is PSD. Conversely, any pair (m, k), with k PSD defines a GP [26], thus ensuring a correspondence between GPs and (m, k) pairs as above. In the context of GPs, k is also called an *kernel*. [1]

A *Gaussian process regression* is done, with $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, by using the model M that says, for each x, $p(y|x, M) = p(y|f(x), M)$, with the prior for f being distributed as $GP(m, k)$. To see that this prior is attractive, assume for now $y = f(x)$. By letting $\mathbf{x} = (x_1, \ldots, x_N)$ [2] and $\mathbf{y} = (y_1, \ldots, y_N)^T$, and assuming $K(\mathbf{x}, \mathbf{x})$ to be non-singular, one can find the posterior distribution for $y_*|x_*, \mathcal{D}, M$

[1] PSD functions have the property that, for $x, x' \in \mathcal{X}$, $k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$, where $\Phi : \mathcal{X} \to \mathcal{H}$ is a map from $\mathcal{X}$ to a Hilbert space $\mathcal{H}$ [99]. In this context, k is called a *kernel function*, and machine learning techniques that uses them are called kernel methods [99]. Hence, in the context of GPs, the terms *kernel function* and covariance function are used interchangeably.

[2] Usually, $x_i \in \mathbb{R}^D$, and $\mathbf{x}$ is written as a matrix whose i-th row is x_i. In this case, we may use $\mathbf{X}$ to denote this matrix instead, and reserve $\mathbf{x}$ to denote each point in $\mathbb{R}^D$. We will however use a more general notation.

without resorting to Bayes' rule, which is important since, in infinite-dimensional spaces, Bayes' rule is more involved, and may not result in a computable expression by itself [58]. [3]

Too see why is this, notice that, by the definition of a GP, for any other $\mathbf{x}^\star \in \mathcal{X}^M$,

$$\begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}(\mathbf{x}) \\ \mathbf{m}(\mathbf{x}^*) \end{bmatrix}, \begin{bmatrix} K(\mathbf{x},\mathbf{x}) & K(\mathbf{x},\mathbf{x}^*) \\ K(\mathbf{x}^*,\mathbf{x}) & K(\mathbf{x}^*,\mathbf{x}^*) \end{bmatrix} \right). \tag{3.3}$$

Hence, by conditioning on $f(\mathbf{x})$, by Appendix B.3:

$$\begin{aligned} f(\mathbf{x}^*)|\mathbf{x}^\star, \mathcal{D}, M = \mathbf{f}^*|\mathbf{f}, M &\sim \mathcal{N}(\mu^\star, \Sigma^*) \\ \mu^\star &= \mathbf{m}(\mathbf{x}^\star) + K(\mathbf{x}^\star,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}(f(\mathbf{x}) - m(\mathbf{x})) \\ \Sigma^\star &= K(\mathbf{x}^\star,\mathbf{x}^\star) - K(\mathbf{x}^\star,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}K(\mathbf{x},\mathbf{x}^\star). \end{aligned} \tag{3.4}$$

Since $\mathbf{x}^*$ was chosen arbitrarily, this implies that $f|\mathcal{D}, M$ itself follows a GP, with mean function and covariance functions given by:

$$\begin{aligned} m_{\mathcal{D}}(x) &= \mu^*(x) = m(x) + K(x,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}(f(\mathbf{x}) - m(x)) \\ k_{\mathcal{D}}(x,x') &= k(x,x') - K(x,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}K(\mathbf{x},x'). \end{aligned} \tag{3.5}$$

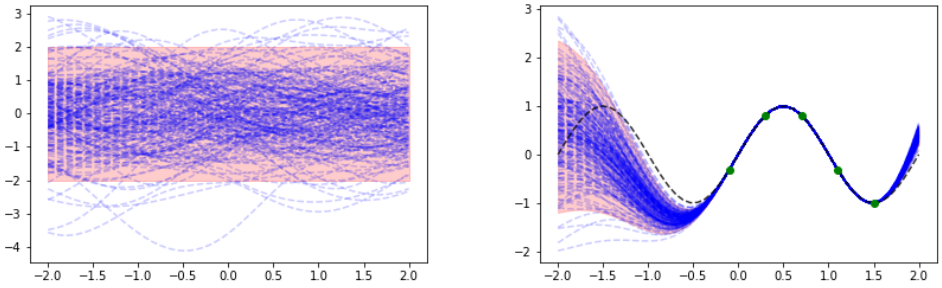
Since $y_* = f(x_*)$ and $\mathbf{y} = f(\mathbf{x}^*)$, $y_*|x_*, \mathcal{D}, M \sim \mathcal{N}(m_{\mathcal{D}}(x_*), k_{\mathcal{D}}(x_*, x_*))$. An illustration of GP regression is shown if Figure 3.1.

3.2.1 Gaussian noise

This equation can be generalized by assuming $p(y|x, M) = \mathcal{N}(y|f(x), \sigma_n^2)$. Then, letting $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, $\mathbf{y} = f(\mathbf{x})$, $\mathbf{y}^* = f(\mathbf{x}^*)$ and

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}(\mathbf{x}) \\ \mathbf{m}(\mathbf{x}^*) \end{bmatrix}, \begin{bmatrix} K(\mathbf{x},\mathbf{x}) + \sigma_n^2 I & K(\mathbf{x},\mathbf{x}^*) \\ K(\mathbf{x}^*,\mathbf{x}) & K(\mathbf{x}^*,\mathbf{x}^*) + \sigma_n^2 I \end{bmatrix} \right). \tag{3.6}$$

[3] For a derivation of Bayes' rule for infinite-dimensional spaces (which involves measure theory), see [107], Section 6.6.



(a) GP prior (b) GP posterior

Figure 3.1: Gaussian process regression of $f(x) = \sin(\pi x)$. In (a), it is shown the prior space distribution, with samples path in blue and confidence interval in red. In (b), the posterior space distribution, after 4 measurements (in green) of $f(x)$ (black). Generating code can be found in `https://github.com/DFNaiff/Dissertation/blob/master/illustrations_dissertation/gp_prior_posterior`.

Conditioning $\mathbf{y}^*$ on $\mathbf{y}$ and, letting $K_\sigma(\mathbf{x}, \mathbf{x}) := K(\mathbf{x}, \mathbf{x}) + \sigma_n I$, we have

$$\begin{aligned} &\mathbf{y}^*|\mathbf{x}^\star, \mathcal{D}, M = \mathbf{y}^*|\mathbf{y}, M \sim \mathcal{N}(\mu^\star, \Sigma^*) \\ &\mu^\star = \mathbf{m}(\mathbf{x}^\star) + K(\mathbf{x}^\star, \mathbf{x}) K_\sigma(\mathbf{x}, \mathbf{x}) {-1}(\mathbf{y} - m(\mathbf{x})) \\ &\Sigma^\star = K(\mathbf{x}^\star, \mathbf{x}^\star) - K(\mathbf{x}^\star, \mathbf{x}) K_\sigma(\mathbf{x}, \mathbf{x})^{-1} K(\mathbf{x}^\star, \mathbf{x}) + \sigma_n I. \end{aligned} \tag{3.7}$$

Notice (3.7) reduces to (3.4) when $\sigma_n^2 = 0$.

3.2.2 General noise

In the general case where $p(y|x, M) = p(y|f(x))$, there is not a closed form solution and one must resort to explicit marginalization and Bayes' rule:

$$\begin{aligned} &p(y_*|x_*, \mathcal{D}) = \\ &= \int p(y_*|f(x_*)) p(f(x_*)|x_*, \mathbf{x}, \mathbf{y}) df(x_*) \\ &= \int p(y_*|f(x_*)) \int p(f(x_*)|x_*, f(\mathbf{x}), \mathbf{x}) p(f(\mathbf{x})|\mathbf{x}, \mathbf{y}) df(\mathbf{x}) df(x_*) \\ &\propto \int p(y_*|f(x_*)) \int p(f(x_*)|x_*, f(\mathbf{x}), \mathbf{x}) p(\mathbf{y}|f(\mathbf{x})) p(f(\mathbf{x})|\mathbf{x}) df(\mathbf{x}) df(x_*). \end{aligned} \tag{3.8}$$

Since $p(f(\mathbf{x})|\mathbf{x}) = \mathcal{N}(f(\mathbf{x})|m(\mathbf{x}), K(\mathbf{x},\mathbf{x}))$ and $p(f(x_*)|x_*, f(\mathbf{x}), \mathbf{x})$ is given by (3.4), these two terms may be joined together by (B.3). However, this still leaves a double integral, that must be treated by approximate inference methods.

3.2.3 Mean function

Usually, the mean function m is set to zero, letting the covariance function determine the whole structure of the regression. This is a reasonable assumption since, for any $f \sim GP(m,k)$, due to the sum of a Gaussian distribution and a constant being itself Gaussian with the constant added to its mean, we have $f - m \sim GP(0,k)$. Thus, fixed the model (m,k), one can then do the GP regression on $f-m$ and then later add m. This is particularly useful if it is assumed that f is modeled by some function m that is known to be an incomplete model, thus complementing the regression by modeling this incompleteness by a zero mean GP.

3.3 Covariance functions

As said in the previous section, covariance functions k must be PSD. This raises the question on which kind of functions are PSD, thus able to define a GP. A few functions can be easily shown to be PSD directly by their definitions, such as:

- The constant function $k(x,x') = c \geq 0$, since the matrix $K_{i,j} = c$ is PSD, for all $c \geq 0$
- $k(x,x') = \mathbb{I}_{x=x'}$, since the corresponding matrix K is the identity matrix
- If $\mathcal{X} = \{x_1, ..., x_N\}$ is a finite set of size N, and k is such that, for $\mathbf{x} = (x_1, ..., x_N)$, $K(\mathbf{x},\mathbf{x})$ is PSD, then k is PSD, since for any other subset of $\mathcal{X}$,

the corresponding kernel matrix will be a subset of K, thus also PSD [51].

- If we have an explicit feature map $\Phi : \mathcal{X} \to \mathbb{R}^N$, then $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ is PSD [99]. For instance, if $\mathcal{X} = \mathbb{R}$ and $\Phi(x) = (x, x^2, \ldots, x^N)$, then $k(x, x') = \sum_{i=1}^{N} (xx')^i$ is PSD.

However, for many covariance functions, direct proof of being PSD is infeasible. However, there are some covariance functions that can be shown to be PSD in a indirect way, as shown below.

3.3.0.1 Stationary covariance functions

Definition 3.3.1. Let $\mathcal{X} = \mathbb{R}^d$. An covariance function is *stationary* if $k(x, x') = k(x - x')$, for $k : \mathbb{R}^d \to \mathbb{R}$ [4]. Conversely, k is a *autocovariance function* if $k(x, x') = k(x - x')$ is a covariance function.

For this class of functions, we can reduce the analysis of k to that of k. In particular, the next theorem says that one can analyze the Fourier transform of k to check if it is an autocovariance function, thus k being a covariance function (here, it is conveniente to consider k as a function into $\mathbb{C}$).

Theorem 3.3.2 (Bochner's Theorem)**.** *A function $k : \mathbb{R}^d \to \mathbb{C}$, continuous at 0, is an autocovariance function if and only if*

$$k(\tau) = \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^T \tau} d\mu(\mathbf{s}), \tag{3.9}$$

where μ is a positive finite measure [106, 87]. If μ has a density S, then S an k are

[4] Here we overload the notation, letting the reader infer whether k refers to a covariance function or an autocavariance function by the number of its arguments.

Fourier duals of each other [23].

$$k(\tau) = \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^T \tau} S(\mathbf{s}) d\mathbf{s}$$
$$S(\mathbf{s}) = \int_{\mathbb{R}^D} e^{-2\pi i \mathbf{s}^T \tau} k(\tau) d\tau.$$

In this case, S is called the spectral density corresponding to k.

One particular case of stationary functions are *isotropic* functions, in which $k(\tau)$ is a function of $r = ||\tau||_2$. In this case, $S(\mathbf{s})$ is a function of $s = ||\mathbf{s}||_2$ [3]. For simplicity, those will also be referred as k ans S. We show here some examples of isotropic covariance functions, along with their spectral densities (many others can be found in [87]):

- The squared exponential (SQE) kernel, also called RBF kernel

$$k_{SQE}(r; l) = \exp\left(-\frac{r^2}{l^2}\right), \tag{3.10}$$

  whose spectral density is given by a normal distribution

$$S(s; l) = (2\pi l^2)^{D/2} \exp(-2\pi^2 l^2 s^2) \tag{3.11}$$

  The squared exponential kernel is the most widely used in the field of Gaussian process, and kernel methods in general. However, the squared exponential kernel generates functions that are infinitely differentiable, thus being far too smooth for some applications. Moreover, the resulting kernel matrix tends to be very ill-conditioned, which results in numerical issues in applications with low noise.

- The Matérn class of kernels, parameterized by $\nu > 0$, given by

$$k_{Matern,\nu}(r; l) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{l}\right), \tag{3.12}$$

  where K_ν is the modified Bessel function of second kind. The corresponding

spectral density is a 2ν-degreed multivariate t-distribution

$$S_\nu(s;l) = 2^D \pi^{D/2} \frac{\Gamma(\nu + D/2)(2\nu)^\nu}{\Gamma(\nu) l^{2\nu}} \left(\frac{2\nu}{l^2} + 4\pi^2 s^2 \right)^{-(2\nu+D)/2}. \tag{3.13}$$

If ν is a half-integer, the kernel formula is simplified to a product of a polynomial of order $\nu - 1/2$ and an exponential. The most commonly used values of ν are:

- $\nu = 1/2$, giving $k_{Matern,1/2}(s;l) = \exp(-r/l)$.
- $\nu = 3/2$, giving $k_{Matern,3/2}(s;l) = \left(1 + \sqrt{3}r/l\right) \exp(-\sqrt{3}r/l)$.
- $\nu = 5/2$, giving $k_{Matern,5/2}(s;l) = \left(1 + \sqrt{5}r/l + 5r^2/l^2\right) \exp(-\sqrt{5}r/l)$.

In the limit $\nu \to \infty$, the Matern kernel converges to the squared exponential kernel [106]. In practice, for values of $\nu \geq 7/2$, the Matern kernel is similar enough to the squared exponential kernel to be of use, thus in practice only the three values of ν shown above are used.

- The spectral mixture kernel [117]

$$k_{SM}(\tau) = \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} \exp(-2\pi^2 \tau_d^2 v_q^{(d)}) \cos(2\pi \tau^{(d)} \mu_q^{(d)}), \tag{3.14}$$

which is constructed explicitly as the Fourier dual of mixtures of multivariate normal densities. In [117], it is argued that the spectral mixture kernel approximates many of the kernels in [87], given enough mixtures Q.

3.3.1 Derived kernels

Although PSD functions are relatively hard to find, even with the use of Bochner's Theorem, one can prove that many compositions of base PSD functions are themselves PSD, thus providing many new classes of kernels. Given $\mathcal{X}$ an arbitrary set, k_1,k_2 PSD functions on $\mathcal{X}$, and k_3 a PSD function on a set $\mathcal{Y}$, we have:

- $k(x, x') := k_1(x, x') + k_2(x, x')$ is a PSD function, since, for $\mathbf{x} = (x_1, ..., x_N) \in \mathcal{X}^N$, $K(\mathbf{x}, \mathbf{x}) = K_1(\mathbf{x}, \mathbf{x}) + K_2(\mathbf{x}, \mathbf{x})$, the sum of two PSD matrices, hence a PSD matrix. Similarly, $k(x, x') := k_1(x, x')k_2(x, x')$ is a PSD function, since $K(\mathbf{x}, \mathbf{x}) = K_1(\mathbf{x}, \mathbf{x}) \odot K_2(\mathbf{x}, \mathbf{x})$ (where $A \odot B$ denotes the Hadamard product between A and B), and, according to Schur product theorem [51], $K(\mathbf{x}, \mathbf{x})$ is also PSD as the Hadamard product of two PSD matrices.

- $k([x, y], [x', y']) = k_1(x, x')+k_3(y, y')$ is a PSD function, and so is $k([x, y], [x', y']) = k_1(x, x')k_3(y, y')$. This follows from the fact that both sum and Hadamard product of PSD matrices are PSD.

- For a map $f : \mathcal{Y} \to \mathcal{X}$, $k(y, y') = k_1(f(x), f(x'))$ is aPSD. This follows directly from the fact that if $y_i = f(x_i)$, $i = 1, ..., n$, $K(\mathbf{y}, \mathbf{y}) = K_1(f(\mathbf{x}), f(\mathbf{x}'))$. This property allows us to construct non-stationary kernels from stationary kernels, by using input warping functions. Moreover, this implies that we can substitute $r = ||\mathbf{x} - \mathbf{x}'||_2$ for $r = \sqrt{(\mathbf{x} - \mathbf{x})^T A^{-1}(\mathbf{x} - \mathbf{x})}$, where A is a positive definite matrix, by setting $f(x) = A^{-1/2}x$

The last item allows us to construct kernels with general *outputscale* and *lengthscale* from stationary kernels. That is, if k_0 is an autocovariance function such that $k_0(0) = 1$, then

$$k(x - x') = \theta k\left(\frac{x_1 - x'_1}{l_1}, \ldots, \frac{x_D - x'_D}{l_D}\right) \tag{3.15}$$

is a kernel with outputscale θ and lengthscales $l_1, \ldots, l_D$. We call such kernels *anisotropic* (although strictly speaking, every non-isotropic kernel is anisotropic).

3.4 Model selection

In the above discussion, the model $M = (m, k)$ was assumed to be fixed. In practice, since we have many different kernel functions, each parameterized by a continuous set of parameters (called *hyperparameters*), we need a way to choose the correct model. In the noisy measurement case, we also need to deal with the noise distribution parameters. Fortunately, the Bayesian framework gives a natural way to choose the model.

Assume $p(y|f(x)) = \mathcal{N}(f(x), \sigma_n^2)$. Then,

$$\mathcal{D}|M, \sigma_n = \mathbf{y}|\mathbf{x}, M, \sigma_n \sim \mathcal{N}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}) + \sigma_n \mathbf{I}).$$

Therefore, the likelihood for the model is given by

$$\begin{aligned} \log p(\mathcal{D}|M, \sigma_n) = &-\frac{1}{2}(\mathbf{y} - m(\mathbf{x}))^T (K(\mathbf{x}, \mathbf{x}) + \sigma_n \mathbf{I})^{-1} (\mathbf{y} - m(\mathbf{x})) + \\ &-\frac{1}{2}\log\det(K(\mathbf{x}, \mathbf{x}) + \sigma_n \mathbf{I}) - \frac{1}{2} N \log(2\pi). \end{aligned} \tag{3.16}$$

The important thing to notice is that the likelihood above is actually a marginal likelihood, since

$$p(\mathcal{D}|M, \sigma_n) = \mathbb{E}_{f \sim GP(m,k)} \left[p(\mathcal{D}|f, \sigma_n) \right]. \tag{3.17}$$

Hence, it should display the Occam's razor effect. In fact, the log-determinant term does exactly this, acting as a sort of regularizer. However, this does not mean that GP regression is protected from overfitting (see [71]). Moreover, since the objective function is non-convex, there may be local optima that returns spurious results.

A fully Bayesian approach to GP regression is desirable in order to incorporate fully the hyperparameter knowledge. However, when the number of data is considerably large, Monte Carlo methods becomes inefficient (although it can be used still, see for example [74, 80]). In [76], an approach based on Bayesian Monte Carlo (to be presented in Chapter 4) is also explored. However, efficient marginal-

ization of hyperparameters remains an open problem.

3.5 Computational issues

3.5.1 Jittering

If the noise σ_n^2 is zero, the matrix $K(\mathbf{x}, \mathbf{x})$ may be ill-conditioned, which is usually the case for the SQE kernel. One way to mitigate this problem is to force the existence of an "artificial noise" on $K(\mathbf{x}, \mathbf{x})$, that is, one substitute it for $K(\mathbf{x}, \mathbf{x}) + \sigma_j^2 I$, where σ_j^2 is not a real noise parameter now, but just an stabilizer. In this case, the error caused by the addition of artificial noise is considerably smaller than the error of numerical operations in ill-conditioned matrices, if they are able to be performed at all. The Cholesky decomposition (described below) also helps with the stability of inverse matrix operations.

3.5.2 Scaling with data

The main issue with GP regression is that, given N training points, for a fixed covariance function the evaluations in (3.7) requires at least one operation with the inverse of a $N \times N$ matrix $K(\mathbf{x}, \mathbf{x} + \sigma_n \mathbf{I})$, whose computational cost is of order $\mathcal{O}(N^3)$. The problem is worsened in the case of training a model, since this operation has to be done for each evaluation of $\log p(\mathcal{D}|M, \sigma_n, \theta)$ while training.

Since $K(\mathbf{x}, \mathbf{x}) + \sigma_n \mathbf{I}$ is a positive definite matrix, one can try to mitigate the computational cost by computing the Cholesky decomposition

$$K(\mathbf{x}, \mathbf{x}) + \sigma_n \mathbf{I} = LL^T.$$

Then, given the decomposition, the inverse operations involve inverses of triangular matrices, whose operations costs are of order $\mathcal{O}(N^2)$, while the determinant term in $\log p(\mathcal{D}|M, \sigma_n, \theta)$ can be calculated as $\log\det(LL^T) = 2\sum_i \log L_{ii}$. However, Cholesky decomposition, although faster than other methods like the LU decomposition, still has a computational cost of $\mathcal{O}(N^3)$, hence the scaling problem still exists.

3.6 Online learning

One interesting aspect of GPs is its ability to accumulate online data in a relatively simple manner, provided we do not change its hyperparameters. Consider fixed an initial data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, and a GP model (m, k), if we have a kernel matrix $K_\mathcal{D}$, and its Cholesky factor $L_\mathcal{D}$, resulting in a readily accessible posterior mean function $m_\mathcal{D}(x)$ and covariance function $k_\mathcal{D}(x, x')$. Now, suppose some new data $\mathcal{D}' = \{(x'_j, y'_j)\}_{j=1}^M$ is available, and the practitioner wants to incorporate into a new posterior mean $m_{\mathcal{D}\cup\mathcal{D}'}(x, x')$ and covariance $k_{\mathcal{D}\cup\mathcal{D}'}(x, x')$. A naive manner for doing this would be constructing a new kernel matrix $K_{\mathcal{D}\cup\mathcal{D}'}$ from scratch, and compute its Cholesky factor $L_{\mathcal{D}\cup\mathcal{D}'}$, resulting in a operation cost $\mathcal{O}((M+N)^3)$. Fortunately, there is a clever way to obtain $L_{\mathcal{D}\cup\mathcal{D}'}$ from $L_\mathcal{D}$ with $\mathcal{O}(M^3 + MN^2)$ cost (assuming $K_{\mathcal{D}\cup\mathcal{D}'}$ stays positive-definite). The following argument is adapted from [75], where it is considered the upper Cholesky factor.

To see this, consider $\mathbf{x}_N = (x_i)_{i=1}^N$, $\mathbf{x}_M = (x'_j)_{j=1}^M$ and $\mathbf{x} = \mathbf{x}_N \cup \mathbf{x}_M$ (here $\cup$ denotes concatenation). Then, noting $K_\mathcal{D} = K(\mathbf{x}_N, \mathbf{x}_N)$, we have

$$K_{\mathcal{D}\cup\mathcal{D}'} = \begin{bmatrix} K(\mathbf{x}_N, \mathbf{x}_N) & K(\mathbf{x}_N, \mathbf{x}_M) \\ K(\mathbf{x}_M, \mathbf{x}_N) & K(\mathbf{x}_M, \mathbf{x}_M) \end{bmatrix}. \tag{3.18}$$

Then $L_{\mathcal{D}\cup\mathcal{D}'}$ must be of the form

$$L_{\mathcal{D}\cup\mathcal{D}'} = \begin{bmatrix} L_\mathcal{D} & 0 \\ S & \tilde{L} \end{bmatrix}, \tag{3.19}$$

with $\tilde{L}$ being lower triangular. This is because

$$L_{\mathcal{D}\cup\mathcal{D}'}L_{\mathcal{D}\cup\mathcal{D}'}^T = \begin{bmatrix} L_{\mathcal{D}} & 0 \\ S & \tilde{L} \end{bmatrix}\begin{bmatrix} L_{\mathcal{D}}^T & S^T \\ 0 & \tilde{L}^T \end{bmatrix} = \begin{bmatrix} L_{\mathcal{D}}L_{\mathcal{D}}^T & L_{\mathcal{D}}S^T \\ SL_{\mathcal{D}}^T & SS^T + \tilde{L}\tilde{L}^T \end{bmatrix}$$
$$= \begin{bmatrix} K(\mathbf{x}_N, \mathbf{x}_N) & K(\mathbf{x}_N, \mathbf{x}_M) \\ K(\mathbf{x}_M, \mathbf{x}_N) & K(\mathbf{x}_M, \mathbf{x}_M) \end{bmatrix} = K_{\mathcal{D}\cup\mathcal{D}'}. \tag{3.20}$$

This readily shows not only that $L_{\mathcal{D}\cup\mathcal{D}'}$ must be of the format in (3.19), but it gives a way to get S and $\tilde{L}$: calculate $S = \left(L_{\mathcal{D}}^{-1}K(\mathbf{x}_N, \mathbf{x}_M)\right)^T$, and $\tilde{L}$ is the lower Cholesky factor of $K(\mathbf{x}_M, \mathbf{x}_M) - SS^T$, whose operations are of cost $\mathcal{O}(N^2M)$ and $\mathcal{O}(M^3)$.

4 BAYESIAN MONTE CARLO

Consider the integral

$$Z = \int_{\mathbb{R}^D} f(x)p(x)dx. \tag{4.1}$$

In the following discussion, we drop $\mathbb{R}^D$ from the integral symbol, for simplicity. If $p(x)$ is a distribution whose sampling is easy, the bottleneck for a simple Monte Carlo method for estimating Z would be the evaluation cost of f. If evaluating f is costly then, a naive Monte Carlo method often becomes infeasible. In this section, we present a GP-based method that tries to circumvent this bottleneck.

4.1 GP approximation for the integrand

In Bayesian Monte Carlo (BMC), or Bayesian quadrature [41, 77] [1], f itself is treated as a random function, and a GP prior $GP(m, k)$ is put on f. Therefore, given a set $\mathcal{D} = \{(x_i, f(x_i))\}_{i=1}^N$ of N evaluations, the posterior random function $f_\mathcal{D}$ is also distributed according to a GP. In particular, since linear maps of GPs are themselves GPs [87, 46], this implies that the random variable

$$Z_\mathcal{D} = \int f_\mathcal{D}(x)p(x)dx \tag{4.2}$$

is a Gaussian random variable. Since we can find the mean by

$$\mathbb{E}[Z_\mathcal{D}] = \mathbb{E}\left[\int f_\mathcal{D}(x)p(x)dx\right] = \int \mathbb{E}[f_\mathcal{D}(x)]p(x)dx, \tag{4.3}$$

[1] The original name Bayesian quadrature describes more accurately the method, however the name Bayesian Monte Carlo will be used in this text. At the literature, both names can be found in roughly equal proportion

and the variance by,

$$\begin{aligned}\mathrm{Var}(Z_{\mathcal{D}}) &= \mathbb{E}[(Z_{\mathcal{D}} - \mathbb{E}[Z_{\mathcal{D}}])^2] \\ &= \mathbb{E}\left[\left(\int (f_{\mathcal{D}}(x) - \mathbb{E}[f_{\mathcal{D}}(x)])p(x)dx\right)^2\right] \\ &= \int\int \mathbb{E}\left[(f_{\mathcal{D}}(x) - \mathbb{E}[f_{\mathcal{D}}(x)])(f_{\mathcal{D}}(x') - \mathbb{E}[f_{\mathcal{D}}(x')])\right] p(x)p(x')dxdx' \\ &= \int\int \mathrm{Cov}(f_{\mathcal{D}}(x), f_{\mathcal{D}}(x'))p(x)p(x')dxdx'.\end{aligned} \tag{4.4}$$

We have a complete description of the distribution of $Z_{\mathcal{D}}$. Now, by substituting (3.7) in (4.3) and (4.4), we have

$$\begin{aligned}\mathbb{E}[Z_{\mathcal{D}}] &= \int m(x)p(x)dx - \mathbf{z}^T K^{-1}(\mathbf{f} - m(\mathbf{x})) \\ \mathrm{Var}[Z_{\mathcal{D}}] &= \Gamma - \mathbf{z}^T K^{-1}\mathbf{z},\end{aligned} \tag{4.5}$$

where $\mathbf{z} = (z_1, ..., z_N)^T$, with

$$z_i = \int k(x, x_i)p(x)dx, \tag{4.6}$$

and

$$\Gamma = \int\int k(x, x')p(x)p(x')dxdx'. \tag{4.7}$$

In general, a good estimate of $Z_{\mathcal{D}}$ is its mean, although if there is an asymmetric loss function associated with estimating $Z_{\mathcal{D}}$, its variance should be taken in account.

An illustration of the Bayesian Monte Carlo approach to integration is shown in Figure 4.1. There, the distribution is $p(x) = \mathcal{N}(x|0, 0.5)$, and $f(x) = -x^2$. The true value of the integral, and BMC estimation are shown. Notice that, in this example, for $|x| > 2$ the GP estimate $m_{\mathcal{D}}(x)$ of $f(x)$ becomes very inaccurate. However, since low probability mass is assigned outside the interval, the BMC estimation is still close to the target.

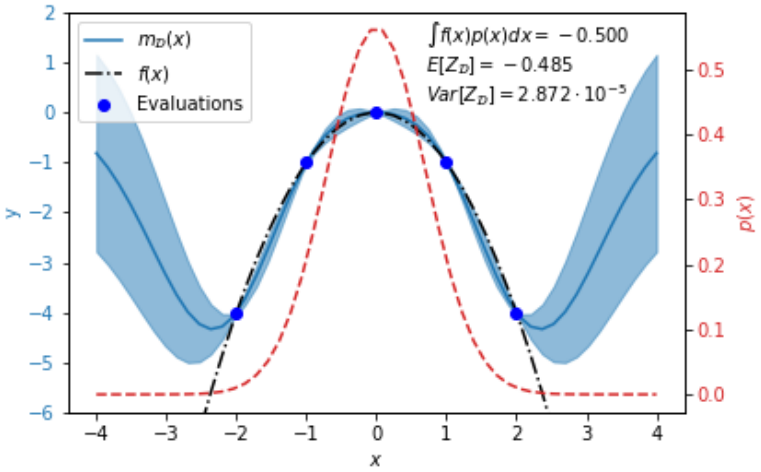


Figure 4.1: Illustration of Bayesian Monte Carlo integration. Here, dash-dot black is $f(x)$, whose value is on the left-axis, while in blue is it's GP mean (dark), and covariance (light), given 5 evaluations (blue circles). In red is $p(x)$, whose value is on the right axis. The true value of the integral, the BMC mean and variance are shown on the top right.

4.1.1 Philosophical remark

At first one may find strange to consider f as a random function, in order to give a prior for it. After all, f is a known, fixed function. However, notice that f is only actually known in so far as one can evaluate it, and if evaluations for f are limited, so is the knowledge of it. And by the discussion in Chapter 1, any object the learner has limited knowledge about should be considered a random variable, independent of the fact that the object is actually random or not.

If this philosophical approach is not convincing, maybe it is better to just think of the BMC method as an artificial way to integrate functions, and follow the famous dictum in quantum mechanics: "Shut up and calculate" [69].

4.2 Kernel-distribution combinations

In general, neither (4.6) nor (4.7) are analytically available, just like the original integral (4.1). However, evaluating the kernel function is in general cheap, so if evaluation of f is expensive, there may be still computational gains in using BMC. For some particular distributions $p(x)$, combined with some suitable kernel choices, (4.6) nor (4.7) does lend analytical solutions, or can be treated in a relatively cheap manner. In the following, $m(x) = 0$ for simplicity, thus omitting the term.

4.2.1 SQE kernel with Gaussian distributions

Assume $p(x) = \mathcal{N}(x|\mu, \Sigma)$, and k is a anisotropic squared exponential kernel, with vertical scale θ and length scales $l = (l_1, ..., l_D)$. Notice then that, by letting $A := \text{diag}(l_1^2, ..., l_D^2)$, we have

$$k(x, x') = \theta \exp\left(-\frac{1}{2}(x - x')^T A^{-1}(x - x')\right) = \det(2\pi A)\mathcal{N}(x'|x, A) = \det(2\pi A)\mathcal{N}(x|x', A). \tag{4.8}$$

Then, by (B.3):

$$\begin{aligned} &k(x, x')\mathcal{N}(x|\mu, \Sigma) = C(x')\mathcal{N}(x|\tilde{\mu}_{x'}, \tilde{\Sigma}) \\ &C(x') = \frac{\theta}{\det(I + A^{-1}\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x' - \mu)^T (A + \Sigma)^{-1}(x' - \mu)\right) \\ &\tilde{\mu}_{x'} = (A^{-1} + \Sigma^{-1})^{-1}(A^{-1}x' + \Sigma^{-1}\mu) \\ &\tilde{\Sigma} = (A^{-1} + \Sigma^{-1})^{-1}, \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} &k(x, x')\mathcal{N}(x|\mu, \Sigma)\mathcal{N}(x'|\mu, \Sigma) = \hat{C}\mathcal{N}(x|\tilde{\mu}_{x'}, \tilde{\Sigma})\mathcal{N}(x'|\tilde{\mu}_x, \tilde{\Sigma}) \\ &\hat{C} = \frac{\theta}{\det(I + 2A^{-1}\Sigma)^{1/2}}. \end{aligned} \tag{4.10}$$

Hence, by substituting (4.9) and (4.10) into (4.6) and (4.7), we find

$$\begin{aligned} z_i &= \frac{\theta}{\det(I + A^{-1}\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu)^T (A + \Sigma)^{-1}(x_i' - \mu)\right) \\ \Gamma &= \frac{\theta}{\det(I + 2A^{-1}\Sigma)^{1/2}}. \end{aligned} \tag{4.11}$$

Substituting in (4.5) we find the desired result:

$$\begin{aligned} \mathbb{E}[Z_{\mathcal{D}}] &= \mathbf{z}^T K^{-1}\mathbf{f} \\ \mathrm{Var}[Z_{\mathcal{D}}] &= \frac{\theta}{\det(I + 2A^{-1}\Sigma)^{1/2}} - \mathbf{z}^T K^{-1}\mathbf{z} \\ z_i &= \frac{\theta}{\det(I + A^{-1}\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu)^T (A + \Sigma)^{-1}(x_i' - \mu)\right) \\ \Gamma &= \frac{\theta}{\det(I + 2A^{-1}\Sigma)^{1/2}}. \end{aligned} \tag{4.12}$$

4.2.2 Mixture distributions

Consider taking expectations in respect to a mixture distribution

$$p(x) = \sum_{i=1}^{M} \alpha_i p_i(x). \tag{4.13}$$

From (4.6) and (4.7), it is straightforward to see that in this case

$$\begin{aligned} z_i &= \sum_{i=1}^{M} \alpha_i \int k(x, x_i) p_i(x) dx \\ \Gamma &= \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j \int\int k(x, x') p_i(x) p_j(x') dx dx'. \end{aligned} \tag{4.14}$$

Then, provided it is possible to calculate the integrals above for each component $p_i(x)$, one can easily use mixture distributions from these coefficients.

An important case is considering mixtures of normal distributions $p(x) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(x|\mu_j, \Sigma_j)$, and the squared-exponential kernel. Then, by substituting in (4.6) and (4.7), and considering the results for normal distributions in (4.11), we

find

$$
\begin{aligned}
z_i &= \sum_{j=1}^{M} \alpha_j z_{i,j} \\
z_{i,j} &= \frac{\theta}{\det(I + A^{-1}\Sigma_j)^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_j)^T (A + \Sigma_j)^{-1}(x_i' - \mu_j)\right) \\
\Gamma &= \sum_{j=1}^{M}\sum_{m=1}^{M} \alpha_j \alpha_m \Gamma_{j,m}, \\
\Gamma_{j,m} &= \frac{\theta}{\det(I + A^{-1}(\Sigma_j + \Sigma_m))^{1/2}} \exp\left(-\frac{1}{2}(\mu_j - \mu_m)^T (A + \Sigma_j + \Sigma_m)^{-1}(\mu_j - \mu_m)\right).
\end{aligned}
\tag{4.15}
$$

4.2.3 Tensor product kernels and diagonal-covariance Gaussian distributions

Consider tensor product kernels in $\mathbb{R}^D$ of the form

$$
k(x, x') = \prod_{d=1}^{D} k_d(x_d, x_d'), \tag{4.16}
$$

and a Gaussian distribution $p(x) = \mathcal{N}(x|\mu, \Sigma)$ with diagonal covariance $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^d)$, so that $p(x) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d, \sigma_d^2)$. Then,

$$
\begin{aligned}
z_i &= \int \prod_{d=1}^{D} k_d(x_d, x_{i,d}) \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d, \sigma_d^2) dx \\
&= \prod_{d=1}^{D} \int k_d(x, x_{i,d}) \mathcal{N}(x|\mu_d, \sigma_d^2) dx,
\end{aligned}
\tag{4.17}
$$

and

$$
\begin{aligned}
\Gamma &= \int\int \prod_{d=1}^{D} k_d(x_d, x_d') \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d, \sigma_d^2) \prod_{d=1}^{D} \mathcal{N}(x_d'|\mu_d, \sigma_d^2) dx dx' = \\
&= \prod_{d=1}^{D} \int\int k_d(x, x') \mathcal{N}(x|\mu_d, \sigma_d^2) \mathcal{N}(x'|\mu_d, \sigma_d^2) dx dx'.
\end{aligned}
\tag{4.18}
$$

Consider each individual component in (4.17). Even with none of then being analytically computable, they can easily be approximated using Gauss-Hermite quadrature, that approximates

$$\int f(x)\mathcal{N}(x|\mu,\sigma^2)dx \approx \frac{1}{\sqrt{\pi}}\sum_{k=1}^{K} w_k f(\sqrt{2}\sigma x_k + \mu), \tag{4.19}$$

where x_i are the roots of the physicists' Hermite polynomial

$$H_K(x) = (-1)^N e^{x^2}\frac{d^K}{dx^K}e^{-x^2}, \tag{4.20}$$

and w_k are the associated weights

$$w_k = \frac{2^{K-1}K!\sqrt{\pi}}{K^2(H_{K-1}(x_k))^2}. \tag{4.21}$$

An analysis of this approximation can be found in [64]. It is important to notice that for each K, $\{(x_k, w_k)\}_{k=1}^K$ are fixed, unlike standard Monte Carlo methods that would require sampling from $\mathcal{N}(x|\mu,\sigma^2)$. Applying (4.19) to (4.17), one finds that

$$z_i \approx \prod_{d=1}^{D}\frac{1}{\sqrt{\pi}}\sum_{k=1}^{K} w_k k_d(\sqrt{2}\sigma_d x_k + \mu_d, x_{i,d}), \tag{4.22}$$

and that

$$\Gamma \approx \prod_{d=1}^{D}\frac{1}{\pi}\sum_{k,k'=1}^{K} w_k w_{k'} k_d(\sqrt{2}\sigma_d x_k + \mu_d, \sqrt{2}\sigma_d x_{k'} + \mu_d). \tag{4.23}$$

By the discussion in section 4.2.2, one can easily extend this to mixtures of Gaussians with diagonal covariance. Therefore, one is free to use more flexible kernels than the squared exponential one, provided they are tensor product kernels. The trade-off is that $p(x)$ is a more restrictive distribution, but, as discussed next, this is not an insurmountable restriction. Moreover, the techniques presented in chapters 4 and 5 uses exactly those kinds of distributions as approximations.

4.2.4 Importance reweighting for Bayesian Monte Carlo

One can use the above results for doing integral estimations for general distributions using squared exponential kernels, without Monte Carlo integration of kernels, by an importance re-weighting trick

$$\int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)}q(x)dx, \tag{4.24}$$

where $q(x)$ may be either a normal distribution or a mixture of normals. This becomes interesting because since mixtures of normals can approximate continuous densities arbitrarily close as the number of mixtures goes to infinity [28], thus providing good re-weighting distributions.

4.3 Bayesian Monte Carlo for positive integrands

For many cases the integral we are interested is one arising from marginalization:

$$p(\mathcal{D}) = \int L(x)p(x)dx = \int p(\mathcal{D}|x)p(x)dx. \tag{4.25}$$

In particular, in this case $L(x)$ must be a positive function. However, applying the BMC method naively can result in rather inaccurate evaluations, due to the fact that in general, GP regression can predict negative means even when all function evaluations are positive. This way, the positivity of the GP mean for $L(x)$ is not guaranteed, resulting in possibly pathological predictions [41].

In [75], it is proposed to make a GP regression by putting an prior over $\log L(x) \sim GP(m, k)$. However, this means that for predictive distribution in the original space, $L(x|x_D) \sim \text{Lognormal}(m_{\mathcal{D}}(x), k_{\mathcal{D}}(x, x))$, which results in

$$\mathbb{E}\left[\int L(x|x_{\mathcal{D}})p(x)dx\right] = \int e^{m_{\mathcal{D}}(x)+\frac{1}{2}k_{\mathcal{D}}(x,x)}p(x)dx, \tag{4.26}$$

which is still non-tractable integral, requiring further approximations. It is proposed in [78] a somehow complicated heuristic to circumvent this problem, relying in a number of approximations whose accuracy is questionable, and an inner application of BMC.

In [44], the transformation used is the square-root transformation, so the prior for GP regression is over $\tilde{L}(x) = \sqrt{2L(x) - \alpha}$, with α being a small positive scalar, resulting in $L(x|x_\mathcal{D}) = \alpha + \frac{1}{2}\tilde{L}(x|x_\mathcal{D})^2$. However, this way we have $\mathbb{E}[L(x|x_\mathcal{D})] = \alpha + \frac{k_\mathcal{D}(x,x)}{2}(1 + m_\mathcal{D}(x)^2)$, hence, just like (4.26), we can't arrive at a tractable mean. In order to circunvent this, in [44] two approaches are proposed. The first is a linearization of $\alpha + \frac{1}{2}\tilde{L}(x|x_\mathcal{D})^2$ around $m_\mathcal{D}(x)$, resulting in

$$L(x|x_\mathcal{D}) \approx L^\mathcal{L}(x|x_\mathcal{D}) = \alpha - \frac{1}{2}m_\mathcal{D}(x)^2 + m_\mathcal{D}(x)\tilde{L}(x), \tag{4.27}$$

which, since it is a affine transformation of $\tilde{L}(x)$, results an approximate GP distribution for $L(\cdot|x_\mathcal{D})$:

$$\begin{aligned} L^\mathcal{L}(\cdot|x_\mathcal{D}) &\sim GP(m^\mathcal{L}_\mathcal{D}(x), k^\mathcal{L}_\mathcal{D}(x)) \\ m^\mathcal{L}_\mathcal{D}(x) &= \alpha + \frac{1}{2}m_\mathcal{D}(x) \\ k^\mathcal{L}_\mathcal{D}(x,x') &= m_\mathcal{D}(x)k_\mathcal{D}(x,x')m_\mathcal{D}(x). \end{aligned} \tag{4.28}$$

The second proposal is to approximate $L(\cdot|x_\mathcal{D})$ by a random GP-distributed function $L^\mathcal{M}(\cdot|x_\mathcal{D}) = GP(m^\mathcal{M}_\mathcal{D}, k^\mathcal{M}_\mathcal{D})$, where $m^\mathcal{M}_\mathcal{D}(x)$ and $k^\mathcal{M}_\mathcal{D}(x)$ are chosen so that $L^\mathcal{M}(\cdot|x_\mathcal{D})$ is moment-matched with $L(\cdot|x_\mathcal{D})$, that is, $m^\mathcal{M}_\mathcal{D}(x) = \mathbb{E}[L(x|x_\mathcal{D})]$ and $k^\mathcal{M}_\mathcal{D}(x,x') = \mathrm{Cov}(L(x|x_\mathcal{D}), L(x'|x_\mathcal{D}))$. This results in the approximation

$$\begin{aligned} L^\mathcal{M}(\cdot|x_\mathcal{D}) &\sim GP(m^\mathcal{M}_\mathcal{D}(x), k^\mathcal{M}_\mathcal{D}(x)) \\ m^\mathcal{M}_\mathcal{D}(x) &= \alpha + \frac{1}{2}(m_\mathcal{D}(x) + k_\mathcal{D}(x,x)) \\ k^\mathcal{M}_\mathcal{D}(x,x') &= \frac{1}{2}k_\mathcal{D}(x,x')^2 + m_\mathcal{D}(x)k_\mathcal{D}(x,x')m_\mathcal{D}(x). \end{aligned} \tag{4.29}$$

In particular, both approaches results in tractable integrals for gaussian distributions and SQE kernels.

The idea of moment-matching is extended to a general setting in [22], where it

is considered the integral (4.1), where f is now a function from $\mathbb{R}^D$ to a strict subset $\mathcal{Y}$ of $\mathbb{R}$. In the case presented previously, $\mathcal{Y} = (0, \infty)$, while another important case is $\mathcal{Y} = (0, 1)$. Considering an bijective map $\epsilon : \mathbb{R} \to \mathcal{Y}$, it is placed a GP prior $GP(m, k)$ over $g = \epsilon^{-1} \circ f$. Then, the posterior distribution for $f(\cdot|x_\mathcal{D}$ is approximated by a moment-matched GP with mean $m_\mathcal{D}^\mathcal{M}(x) = \mathbb{E}[\epsilon(g(x|x_\mathcal{D}))]$ and $k_\mathcal{D}^\mathcal{M}(x, x') = \mathrm{Cov}(\epsilon(g(x|x_\mathcal{D})), \epsilon(g(x'|x_\mathcal{D})))$. In particular, for $\epsilon^{-1}(x) = \log(x)$, the same warping considered in [75], the moment matched mean and covariance becomes

$$\begin{aligned} m_\mathcal{D}^\mathcal{M}(x) &= e^{m_\mathcal{D}(x)+\frac{1}{2}k_\mathcal{D}(x,x)} \\ k_\mathcal{D}^\mathcal{M}(x, x') &= e^{m_\mathcal{D}(x)+\frac{1}{2}k_\mathcal{D}(x,x)} e^{m_\mathcal{D}(x')+\frac{1}{2}k_\mathcal{D}(x',x')} \left(e^{k_\mathcal{D}(x,x')} - 1\right). \end{aligned} \tag{4.30}$$

However, when integrated this GP does not result in a integrable mean or variance. One further proposal of [22] is to do a Taylor expansion of $m_\mathcal{D}^\mathcal{M}(x)$,

$$m_\mathcal{D}^\mathcal{M}(x) \approx 1 + m_\mathcal{D}(x) + \frac{1}{2}k_\mathcal{D}(x, x) + \frac{1}{2}\left(m_\mathcal{D}(x) + \frac{1}{2}k_\mathcal{D}(x, x)\right)^2 + \ldots, \tag{4.31}$$

and of $k_\mathcal{D}^\mathcal{M}(x, x')$,

$$\begin{aligned} k_\mathcal{D}^\mathcal{M}(x, x') \approx 1 &+ k_\mathcal{D}(x, x') + \frac{1}{2}k_\mathcal{D}(x, x')^2 \\ &+ k_\mathcal{D}(x, x')\left(m_\mathcal{D}(x) + \frac{1}{2}k_\mathcal{D}(x, x) + m_\mathcal{D}(x') + \frac{1}{2}k_\mathcal{D}(x', x')\right)^2 + \ldots \end{aligned}, \tag{4.32}$$

which, depending on the mean function and kernel function (for instance, zero mean and SQE kernel), is integrable when truncated [22].

The use of function warping raises a question on how to choose the GP hyperparameters. In [22], it is argued that for the moment-matched procedure, one should choose the hyperparameters considering the approximated GP distribution for f, $GP(m^\mathcal{M}, k^\mathcal{M}(x, x'))$ and the data $(\mathbf{x}, f(\mathbf{x}))$, as opposed to the exact GP distribution for g, where it is considered $GP(m, k)$ and the data $(\mathbf{x}, g(\mathbf{x}))$. The authors of [22] call the first approach *f-space* optimization, and the second one *g-space* op-

timization, and it is argued that empirically, f-space optimization results in better integral estimations for their test cases. [2]

4.4 Choosing evaluation points

The integral estimate variance $\mathrm{Var}[Z_{\mathcal{D}}]$ yields a natural metric for choosing a set of evaluation points $X_{\mathcal{D}} = \{x_1, ..., x_N\}$. Namely, since $\mathrm{Var}[Z_{\mathcal{D}}]$ does not depend on the evaluation values $\{f(x_1), \ldots, f(x_N)\}$, one can, given a budget of N evaluations, minimize the function $\alpha_{\mathrm{OBQ}} : \mathbb{R}^{dN} \to \mathbb{R}$ given by

$$\alpha_{\mathrm{OBQ}}(X_{\mathcal{D}}) = \mathrm{Var}[Z_{\mathcal{D}}](X_{\mathcal{D}}) = \Gamma(X_{\mathcal{D}}) - \mathbf{z}(X_{\mathcal{D}})^T K(X_{\mathcal{D}})^{-1}\mathbf{z}(X_{\mathcal{D}}) \tag{4.33}$$

However, the evaluation of this function has a $\mathcal{O}(N^3)$ cost, due to the need for matrix inversion, is not necessarily convex, and is defined on a very high dimensional space, so its minimization will be feasible only in specific cases. An easier way would be to take a greedy approach, that is, given $X_{\mathcal{D}_m} = \{x_1, ..., x_m\}$ previously chosen evaluation points, with $m < n$, we choose x_{m+1} such that the variance of $Z_{\mathcal{D}_m \cup \{x_{m+1}, f(x_{m+1})\}}$ is minimized, that is, we are looking to minimize the objective function

$$\alpha^m_{\mathrm{SBQ}}(x_{m+1}) = \alpha^m_{\mathrm{OBQ}}(x_{m+1}; x_1, \ldots, x_m) \tag{4.34}$$

$$\alpha^m_{SBQ} : \mathbb{R}^d \to \mathbb{R}, \quad x_{m+1} \to \mathrm{Var}[Z_{\mathcal{D}^m \cup \{x_{m+1}, f(x_{m+1})\}}](x_{m+1}), \tag{4.35}$$

This algorithm is referred as Sequential Bayesian Quadrature (SBQ) in [18], while in the same work the first objective is referred as Optimal Bayesian Quadrature (OBQ). Notice that the kernel matrix Cholesky decomposition can be updated in $\mathcal{O}(N^2)$ by the discussion in Section 3.6, and that this operation is differentiable,

[2] Tests with f-space optimization in this work weren't successful.

since the Cholesky decomposition is differentiable [102, 73].

One can simplify this objective function, by substituting it for an heuristic of finding the point of maximum variance of the integrand, that is, substituting

$$\arg\min_{x_{m+1}} \int f_{\mathcal{D}\cup\{x_{m+1}\}}(x)p(x) \tag{4.36}$$

for

$$\arg\max_{x_{m+1}} \mathrm{Var}[f_{\mathcal{D}}(x_{m+1})p(x_{m+1})] = \arg\max_{x_{m+1}} k_{\mathcal{D}}(x_{m+1}, x_{m+1})p(x)^2 \tag{4.37}$$

This maximization objective

$$\alpha^m_{\mathrm{US}}(x_{m+1}) = k_{\mathcal{D}}(x_{m+1}, x_{m+1})p(x)^2, \tag{4.38}$$

in referred as *uncertainty sampling* (US).

In the approaches above, since the variance of $f_{\mathcal{D}}(x)$ and $\mathcal{Z}_{\mathcal{D}}$ is independent of evaluation values, there is no active selection for evaluation points, and in principle one can choose then beforehand. This is not true anymore if one optimizes the GP hyperparameters as the evaluation points are selected, or if one considers one of the approaches to treat positive integrands, as shown in previous sections. However, the heuristic motivating the maximization of (4.38) can be extended for warped approaches, as proposed in [44]. For the model (4.28), this returns the maximization objective [3]

$$\alpha^m_{\text{WSABI-L}}(x_{m+1}) = k^{\mathcal{L}}_{\mathcal{D}^m}(x, x)p(x)^2 = k_{\mathcal{D}^m}(x, x)m_{\mathcal{D}^m}(x)^2p(x)^2, \tag{4.39}$$

while for (4.29),

$$\alpha^m_{\text{WSABI-M}}(x_{m+1}) = k^{\mathcal{M}}_{\mathcal{D}^m}(x, x)p(x)^2 = \left(k_{\mathcal{D}^m}(x, x)^2 k_{\mathcal{D}^m}(x, x)m_{\mathcal{D}^m}(x)^2\right)p(x)^2. \tag{4.40}$$

Finally, for the model (4.30) proposed in [22], we arrive at:

$$\alpha^m_{\mathrm{MMLT}_1}(x_{m+1}) = e^{2m_{\mathcal{D}}(x)+k_{\mathcal{D}}(x,x)}\left(e^{k_{\mathcal{D}}(x,x')} - 1\right)p(x)^2. \tag{4.41}$$

[3] In these functions, WSABI refers to *warped sequential active Bayesian integration*, with the letter L standing for linearization and M for moment-matched, and MMLT stands for moment-matched log transform

Recent work [57] analyzes jointly those methods under the umbrella term *adaptive Bayesian quadrature*, where their convergence rates are studied. In particular, there it is considered a version of (4.41) without the $p(x)$ term, resulting in the maximization objective

$$\alpha^m_{\mathrm{MMLT}_2}(x_{m+1}) = e^{2m_{\mathcal{D}}(x)+k_{\mathcal{D}}(x,x)}\left(e^{k_{\mathcal{D}}(x,x')}-1\right). \tag{4.42}$$

In the spirit of Bayesian optimization (discussed below), the maximization objectives presented, and others will be called *acquisition functions*, nomenclature that implies they are criteria for acquiring new information.

4.5 Bayesian Monte Carlo and Bayesian Optimization

Bayesian Monte Carlo belongs to the general family of surrogate model or response surface methods, that are popular in engineering [16, 56, 7]. Namely, surrogate models tries to approximate a function f of hard evaluation by a function $\hat{f}$ that is easy to evaluate, and try to work with it. Gaussian processes are popular as surrogate models [4], since they offer a measure of uncertainty, are flexible, and since the original function evaluation is hard, relatively few data will be available, which mitigates the scaling problem of GPs.

In particular, an important GP-based surrogate method is Bayesian optimization [98, 19, 105], which tries to optimize an expensive function, usually without gradient information. The idea is, from function evaluations $\mathcal{D}_N = \{(x_i, f(x_i)\}_{i=1}^N$, to construct a GP model $\hat{f}_N$, and with it, sequentially use some criteria α_f to choose the x_{N+1} evaluation. Such functions α_f are called acquisition functions in the context. Bayesian optimization in particular is a popular method in machine learning,

[4] In engineering literature they are often called kriging

to tune training parameters of learning algorithms.

5 VARIATIONAL INFERENCE

5.1 Variational inference

Variational inference [12, 14, 72, 119] is an approximate inference technique that, as Laplace's approximation, tries to approximate a density $g(\theta)$ by some density $q(\theta)$, using the unnormalized density $\bar{g}(\theta) = Zg(\theta)$. In the context of variational inference, this $q(\theta)$ will be called the *variational approximation*. Usually in the context of Bayesian inference, $g(\theta) = p(\theta|\mathcal{D})$ and $\bar{g}(\theta) = p(\mathcal{D}|\theta)p(\theta)$.

Unlike Laplace's approximation, variational inference is concerned in choosing $q(\theta)$ by minimizing a global measure of dissimilarity between the distributions $q(\theta)$ and $g(\theta)$, called a *divergence*. A divergence D on a space S of probability densities with the same support is a function $D(\cdot||\cdot) : S \times S \to \mathbb{R}$ such that:

$$\begin{aligned} D(q||p) &\geq 0,\ p, q \in S \\ D(q||p) &= 0 \iff p = q. \end{aligned} \tag{5.1}$$

Thus divergences are a weaker form of a distance, and in general most important classes of divergences do not satisfy neither symmetry nor the triangle inequality. The objective of variational inference is then, given the target distribution g, and a set of candidate distributions $\mathcal{Q}$, to use a divergence D to find an approximation $q^* \in \mathcal{D}$ for g such that

$$q^* = \arg\min_{q \in \mathcal{Q}} D(q||g). \tag{5.2}$$

If $g \in \mathcal{Q}$, then obviously $q^* = g$. However, $\mathcal{Q}$ is a set of distributions chosen so that its elements are easy to work with, and this is not in general the case for g. Usually $\mathcal{Q}$ is parameterized by a set of continuous parameters $\Lambda \subset \mathbb{R}^m$, such

that $q(\theta) = q(\theta; \lambda)$. Then, the problem of minimizing (5.2) becomes a continuous optimization problem

$$\lambda^* = \arg\min_{\lambda \in \Lambda} D(q(\cdot; \lambda)||g), \tag{5.3}$$

and $q^*(\theta) = q(\theta; \lambda^*)$

5.1.1 KL divergence and evidence lower bound

Arguably the most important divergence between probability distributions, widely used in information theory, is the *Kullback-Leibner* (KL) divergence D_{KL}, given by [1]

$$D_{KL}(q||p) = -\mathbb{E}_{\theta \sim q(\theta)} \left[\log \frac{p(\theta)}{q(\theta)} \right]. \tag{5.4}$$

Variational inference has in general the minimization objective $D_{KL}(q||g)$, although some recent methods have been concerned with other divergence objectives [48, 62, 113] [2]. Notice that, since $D_{KL}(q||g) \neq D_{KL}(g||q)$, minimizing $D_{KL}(q||g)$ is different from minimizing $D_{KL}(g||q)$. In fact, the later minimization objective ends up with the related *expectation propagation* technique for approximate inference [12]. In this work we are mainly concerned with the variational inference $D_{KL}(q||g)$ objective.

Since $g(\theta) = \bar{g}(\theta)/Z$, $D_{KL}(q||g)$ can be rewritten as:

$$D_{KL}(q||g) = -\left(\mathbb{E}_{\theta \sim q(\theta)}[\log \bar{g}(\theta)] - \mathbb{E}_{\theta \sim q(\theta)}[\log q(\theta)] \right) + \log Z, \tag{5.5}$$

The quantity inside parenthesis is called the *evidence lower bound* (ELBO)

$$\mathcal{L}_{\bar{g}}(q) = \mathbb{E}_{\theta \sim q(\theta)}[\log \bar{g}(\theta)] + \mathcal{H}(q), \tag{5.6}$$

where $\mathcal{H}(q) := -\mathbb{E}_{\theta \sim q(\theta)}[\log(q(\theta))]$ is the differential entropy of q.

[1] The KL divergence has origins in information theory, and for discrete distributions, it can be interpreted as the average additional information one has to transmit a receiver when you are modeling a random variable distributed according to q by p [72]

[2] Some authors reserve the term variational inference just for the objective $D_{KL}(q||g)$

In general, the dependence on $\bar{g}$ will be omitted until Section 5.5, and the ELBO will be denoted as $\mathcal{L}(q)$. Minimizing $D_{KL}(q||g)$ is equivalent to maximizing $\mathcal{L}(q)$. This way, there is no need to calculate the normalization factor Z, thus markedly improving the flexibility of the method, and the goal becomes

$$q^* = \arg\max_{q \in \mathcal{Q}} \mathcal{L}(q). \tag{5.7}$$

The ELBO is so called because, considering again $p(\theta|\mathcal{D})$, we have that

$$\begin{aligned} \log p(\mathcal{D}) &= \log \mathbb{E}_{\theta \sim p(\theta)}[p(\mathcal{D}|\theta)] \\ &= \log \mathbb{E}_{\theta \sim q(\theta)} \left[\frac{p(\mathcal{D}|\theta)p(\theta)}{q(\theta)} \right] \\ &\geq \mathbb{E}_{\theta \sim q(\theta)} \left[\log \frac{p(\mathcal{D}|\theta)p(\theta)}{q(\theta)} \right] = \mathcal{L}(q). \end{aligned} \tag{5.8}$$

So the ELBO provides a lower bound for the evidence of the model. This means that, when doing model selection between various models $M_1,\ldots,M_t$, one can find their corresponding variational distributions $q^*_{M_1}, \ldots, q^*_{M_t}$ and $\mathcal{L}(q^*_{M_t}), \ldots, \mathcal{L}(q^*_{M_t})$, and then choose the model M with the maximum $\mathcal{L}(q^*_M)$, as a proxy for (2.16). Notice that this is a heuristic, and there is no guarantee that the model with maximum ELBO is actually the one with maximum evidence.

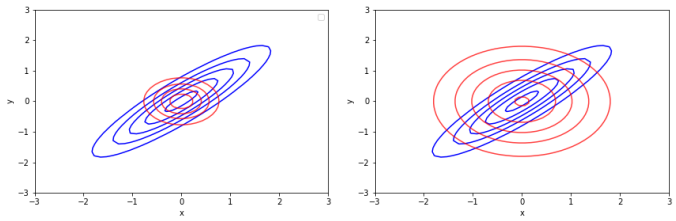
5.1.1.1 Qualitative interpretations

One possible interpretation for maximizing the ELBO is that maximizing the first ELBO term

$$\mathbb{E}_{\theta \sim q(\theta)}[\log \bar{g}(\theta)], \tag{5.9}$$

is the algorithm "trying" to make q have a high probability density wherever $\bar{g}$ has a high unnormalized density, while maximizing the second ELBO term,

$$\mathcal{H}(q) = -\mathbb{E}_{\theta \sim q(\theta)}[\log q(\theta)], \tag{5.10}$$



(a) $D_{KL}(q||g)$ (b) $D_{KL}(g||q)$

Figure 5.1: Difference of behavior when minimizing $D_{KL}(q||g)$ (a) and when minimizing $D_{KL}(g||q)$ (b). Here the true distribution (in blue) is approximated by a multivariate normal distribution with diagonal covariance (in red). Example inspired by [12]. Generating code can be found in `https://github.com/DFNaiff/Dissertation/blob/master/illustrations_dissertation/kl_illustrative_2.py`.

acts as a sort of regularizer preventing q to degenerate to a point mass at the maximum of $\bar{g}$.

It is informative to understand qualitatively which kind of approximations of g variational inference will seek. Suppose some algorithm minimizes $D_{KL}(q||g)$ for $q \in \mathcal{Q}$. Since the algorithm "wants" to make the integrand $q(\theta)(\log q(\theta) - \log g(\theta))$ small, where $g(\theta)$ is close to zero, the $-\log g(\theta)$ term will quickly become large, unless $q(\theta)$ is also close to zero there. However, where $g(\theta)$ is reasonable far away from zero, the algorithm will not "feel" as much pressure to match $q(\theta)$ to the same value, provided that the algorithm assign large values for $q(\theta)$ where $g(\theta)$ is already large. Then, variational inference will tend to underestimate the region where $g(\theta)$ is large. By contrast, expectation propagation will have the reverse behavior, overestimating the region where $g(\theta)$ is far from zero [12]. This is shown in Figure 5.1.

5.2 Mean field variational inference

Traditionally, variational inference has mainly been concerned with factorized variational approximations of the form $q(\theta;\lambda) = \prod_{i=1}^{D} q_i(\theta_i;\lambda_i)$, called *mean field approximation*. The use of such variational posterior greatly simplifies the optimization of ELBO by coordinate descent.

To see this, consider a single term j of the variational approximation $q(\theta) = \prod_{i=1}^{D} q_i(\theta)$. The ELBO for $q(\theta)$ is:

$$\begin{aligned}\mathcal{L}(q) &= \int \log \bar{g}(\theta) \prod_{i=1}^{D} q_i(\theta_i) d\theta - \int \left(\sum_{j=1}^{D} \log q_i(\theta_i) \right) \prod_{i=1}^{D} q_i(\theta_i) d\theta \\ &= \int q_j(\theta_j) \left(\int \log \bar{g}(\theta) \prod_{i \neq j} q_i(\theta_i) d\theta_i \right) d\theta_j + \sum_{i=1}^{D} \mathcal{H}(q_i) \\ &= \int q_j(\theta_j) \log \phi_j(\theta_j) d\theta_j + \mathcal{H}(q_j) + \sum_{i \neq j} \mathcal{H}(q_i),\end{aligned} \tag{5.11}$$

With

$$\log \phi_j(\theta_j) := \int \log(\bar{g}(\theta)) q_{-j}(\theta_{-j}) d\theta_{-j} = \mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log \bar{g}(\theta)],$$

using the notation $q_{-j}(\theta_{-j}) = \prod_{i \neq j} q_i(\theta_i)$. Now, assuming $\exp(\log \phi_j(\theta_j))$ to be integrable over the support of $g(\theta)$, fix every q_i for $i \neq j$, so that the only term to be maximized is q_j. Then, maximizing (5.11) in respect to q_j is equivalent to maximizing the ELBO between q_j and $\exp(\log \phi_j(\theta_j))$. Since there are not any constraints in the choice for q_j

$$q_j^*(\theta_j; q_{-j}) \propto \exp \mathbb{E}_{\theta_{-j} \sim q_{-j}}[\log \bar{g}(\theta)]. \tag{5.12}$$

This readily gives an algorithm to find $q^* = \prod q_j^*$: initialize $q_1, \ldots, q_D$ in a appropriate manner, and then optimize cyclically (5.12).

Convergence is guaranteed due to the convexity of the bound with respect to each factor [12, 17]. This algorithm interacts well with target densities whose

conditional distributions $g(\theta_j|\theta_{-j})$ belongs to the exponential family, that is,

$$g(\theta_j|\theta_{-j}) = h(\theta_j) \exp\left(\eta_j(\theta_{-j})^T t(\theta_j) - a(\eta_j(\theta_{-j}))\right), \tag{5.13}$$

(5.12) reduces to [14]

$$q_j^*(\theta_j; q_{-j}) \propto h(\theta_j) \exp\left(\mathbb{E}_{\theta_j \sim q_{-j}}[\eta_j(\theta_{-j})]^T t(\theta_{-j})\right), \tag{5.14}$$

thus if $\nu_j = \mathbb{E}_{\theta_{-j} \sim q_{-j}}[\eta_j(\theta_{-j})]^T$ is available analytically, the coordinate descent becomes relatively simple.

Research to extend mean field variational inference to large datasets or dimensionality exists [47, 50, 119], however factorized limitations are limited, particularly in its independence assumption. Moreover, many distributions are not in the exponential family, which is the focus of the mean field method, so more generic methods are desirable.

5.3 Generic variational inference

Some recent advances in variational inference [119] are concerned with expanding both the set of possible variational approximations $\mathcal{Q}$ and approximating general classes of posterior distributions $g(\theta)$.

Considering again $\mathcal{Q}$ to be parameterized by a continuous set of parameters Λ, and using the overloaded notation $\mathcal{L}(\lambda) = \mathcal{L}(q(\cdot;\lambda))$, return to the optimization problem

$$\begin{aligned} \lambda^* = \arg\max_{\lambda \in \Lambda} \mathcal{L}(\lambda) &= \arg\max_{\lambda \in \Lambda} \left(\mathbb{E}_{\theta \sim q(\theta;\lambda)}[\log \bar{g}(\theta)] + \mathcal{H}(q(\cdot;\lambda))\right) \\ &= \arg\max_{\lambda \in \Lambda} \mathbb{E}_{\theta \sim q(\theta;\lambda)} \left[\log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right)\right] \end{aligned} \tag{5.15}$$

In general the expectations involved cannot be calculated analytically. However, if one represents $\nabla\mathcal{L}(\lambda)$ as expectations, then by using Monte Carlo methods it is

possible to use stochastic gradients methods [90, 60, 84, 93] to maximize $\mathcal{L}(\lambda)$ [3].

5.3.1 REINFORCE

One proposal for doing this is given in [118] and [86], where $\nabla\mathcal{L}(\lambda)$ is rewritten as [4]

$$\nabla\mathcal{L}(\lambda) = \mathbb{E}_{\theta\sim q(\theta;\lambda)}\left[\log\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\nabla_\lambda \log q(\theta;\lambda)\right], \tag{5.16}$$

and is approximated with its Monte Carlo estimator

$$\nabla\mathcal{L}(\lambda) \approx \frac{1}{K}\sum_{i\in[K],\theta_i\sim q(\theta;\lambda)} \log\frac{\bar{g}(\theta_i)}{q(\theta_i;\lambda)}\nabla_\lambda \log q(\theta_i;\lambda), \tag{5.17}$$

where $[K] = \{1,\ldots,K\}$. In practice, this estimation suffers from high variance, which may hinder optimization. In [118], (5.16) is substituted for

$$\nabla\mathcal{L}(\lambda) = \mathbb{E}_{\theta\sim q(\theta;\lambda)}\left[\left(\log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) + C\right)\nabla_\lambda \log q(\theta;\lambda)\right], \tag{5.18}$$

where C is an arbitrary constant, which is adjusted to control variance [5]. In [86], the variance is controlled by Rao-Blackwellization and control variates instead. Other variance reduction methods for this gradient formulation are proposed in [111] and [94].

5.3.2 Reparameterization trick

An alternative to calculate the gradient of $\mathcal{L}(\lambda)$ as an expectation is known as the *reparameterization trick* [61], which is a general technique for calculating

[3] In this aspect, modern variational inference research benefits greatly from deep learning research, the later relying heavily on stochastic gradient descent, driving much of the recent development of these algorithms.

[4] A quick derivation of this approximation is done in the Appendix **??**.

[5] The reason that this constant can be added is because $\int C\nabla_\lambda \log q(\theta;\lambda)q(\theta;\lambda)d\theta = C\int \nabla_\lambda p(\theta;\lambda)d\theta = C\nabla_\lambda \int p(\theta;\lambda)d\theta = 0$.

gradients of expectations of continuous random variables.

To explain the general ideia, assume X_λ is a continuous random variable distributed according to $f(x;\lambda)$, and one wants to calculate the gradient (in relation to λ) of

$$\mathbb{E}_{X_\lambda}\left[h(X_\lambda)\right] = \int h(x) f(x;\lambda) dx. \tag{5.19}$$

We approximate (5.19) by Monte Carlo

$$\mathbb{E}_{X_\lambda}\left[h(X_\lambda)\right] \approx \sum_{i=1}^{N} h(x_{i,\lambda}), \quad x_{i,\lambda} \sim f(x;\lambda). \tag{5.20}$$

Now suppose that there is some random variable Y, *not depending on* λ, with density $r(y)$, such that $X_\lambda = s(Y;\lambda)$ (for instance, if $X_{\mu,\sigma} \sim \mathcal{N}(\mu,\sigma^2)$, then being $Y \sim \mathcal{N}(0,1)$, we have that $X_{\mu,\sigma} = s(Y;\mu,\sigma) = \sigma Y + \mu$). In this case, the Monte Carlo estimator (5.20) is rewritten as

$$\mathbb{E}_{X_\lambda}\left[h(X_\lambda)\right] \approx \sum_{i=1}^{N} h(s(y_i,\lambda)), \quad y_i \sim r(y), \tag{5.21}$$

which is an expression *whose gradient in relation to* λ *can be taken*, and is the Monte Carlo estimator of

$$\int h(s(y;\lambda)) r(y) dy = \mathbb{E}_Y[h(s(Y;\lambda))] \tag{5.22}$$

Formally, by letting A be the support of Y and B_λ be the support of X_λ, if $s(y;\lambda)$ if bijective with injective derivative:

$$\begin{aligned}
\mathbb{E}_{X_\lambda \sim f(x;\lambda)}[h(x)] &= \int_{B_\lambda} h(x) q(x;\lambda) dx \\
&= \int_A h(s(y;\lambda)) q(s(y;\lambda);\lambda) |\det(s'(y;\lambda))| dy \\
&= \int_A h(s(y;\lambda)) r(y) dy = \mathbb{E}_{Y\sim r(y)}[h(s(y;\lambda))],
\end{aligned} \tag{5.23}$$

which shows that the reparameterization trick is valid.

To apply this to $\mathcal{L}(\lambda)$, assume $\theta_\lambda \sim q(\theta;\lambda)$ is such that $\theta_\lambda = s(\epsilon;\lambda)$, with $\epsilon \sim r(\epsilon)$. Then, applying reparameterization, we have

$$\begin{aligned}\nabla\mathcal{L}(\lambda) &= \nabla\left(\mathbb{E}_{\theta\sim q(\theta;\lambda)}\left[\log\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right]\right)\\ &= \nabla\left(\mathbb{E}_{\epsilon\sim r(\epsilon)}\left[\log\frac{\bar{g}(s(\epsilon;\lambda))}{q(s(\epsilon;\lambda);\lambda)}\right]\right)\\ &\approx \nabla_\lambda\left(\frac{1}{K}\sum_{i\in[K],\epsilon_i\sim r(\epsilon)}\log\frac{\bar{g}(s(\epsilon_i;\lambda))}{q(s(\epsilon_i;\lambda);\lambda)}\right)\end{aligned} \tag{5.24}$$

In [119], it is argued that the observed lower variance of this estimation methods, if compared to the one given by (5.16), may be due to the fact that reparameterization trick takes in account the gradient of the target distribution, instead of just the gradient of the variational distribution as in (5.16). Moreover, in cases that the entropy of $q(\cdot;\lambda)$ can be estimated analytically, the reparameterization trick can be applied only to $\bar{g}(\theta)$, leading to lower variance. Finally, it is important to notice that this format is more readily integrated in an automatic differentiation package, since it suffices to calculate the sum inside the gradient and backpropagate it in relation to λ.

5.4 Mixtures of gaussians for variational approximations

The idea of using mixture distributions for variational inference dates back to the late 90s [13, 53], originally developed for a limited number of target distributions, and later is explored for approximating general distributions in [37, 95], leading to recent work in it [2, 6, 45, 54, 70]. The technique presented here uses the reparameterization trick, in a vein similar to the one presented in [70]. In general, the mixture distribution considered is one of Gaussian distributions (as it is in this work), although many of those extends to more general mixtures.

More formally, consider as the set of candidate proposals

$$\mathcal{Q}_k := \left\{ \sum_{i=1}^{k} w_i f(\cdot;\lambda_i) \middle| f(\cdot;\lambda_i) \in \mathcal{Q}_1, 1 \leq i \leq k; (w_1, ..., w_k) \in \Delta_k \subset \mathbb{R}^k \right\}, \quad (5.25)$$

wher Δ_k denotes the probability simplex $\{(x_1, \ldots, x_k) \in \mathbb{R}^{+k} | \sum_{i=1}^{k} x_k = 1\}$, and $\mathcal{Q}_1 = \{f(\cdot;\lambda) | \lambda \in \Lambda \subset \mathbb{R}^m\}$ is a parameterized set of distributions. In the mixture of multivariate normals case,

$$\mathcal{Q}_1 := \{f(\cdot;\mu,\Sigma) = \mathcal{N}(\cdot;\mu,\Sigma) | \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d\times d}, \Sigma \geq 0\} \quad (5.26)$$

is the set of multivariate normal distributions. In many cases, it is interesting to restrict $\mathcal{Q}_1$ further so that Σ is diagonal. Mixtures of distributions are interesting as variational approximations since their expectations are easily available

$$\mathbb{E}_{X\sim\sum_i w_i f_i}[h(X)] = \sum_i w_i \mathbb{E}_{X_i\sim f_i}[h(X_i)], \quad (5.27)$$

as well as their covariances

$$\begin{aligned} &\mathrm{Cov}_{X\sim\sum_i w_i f_i}(X) = \sum_i w_i \left(\Sigma_i + \mu_i \mu_i^T\right) - \mu\mu^T, \\ &\Sigma_i = \mathrm{Cov}_{X_i\sim f_i}(X_i), \quad \mu_i = \mathbb{E}_{X_i\sim f_i}[X], \quad \mu = \mathbb{E}_{X\sim\sum_i w_i f_i}[X]. \end{aligned} \quad (5.28)$$

Furthermore, samples of mixtures can be easily generated from the base distributions, by choosing mixture i with probability w_i, and then sampling X from $f(\cdot;\lambda_i)$ [6]. Furthermore, letting $\mathcal{Q}_\infty = \cup_{i=1}^{\infty} \mathcal{Q}_i$ and $\mathcal{Q}_1$ be these set of Gaussian distributions, $\mathcal{Q}_\infty$ is dense in the set of continuous distributions [28], so in this case any continuous distribution can be approximated arbitrarily close, in principle.

Considering mixtures of Gaussians, for fixed k, in order to find the parameters

$$\lambda = (w_1, \mu_1, \Sigma_1, \ldots, w_k, \mu_k, \Sigma_k)$$

of

$$q_k^* = \arg\max_{q_k \in \mathcal{Q}_k} \mathcal{L}(q_k), \quad (5.29)$$

[6] Batching this process in order to sample many variables, escaping loops in interpreted languages with support for numerical operations (such as Python with Numpy) requires some care, but it is possible in few lines.

one needs first to find suitable parameterizations for Σ_i and $\mathbf{w} := (w_1, ..., w_k)$. For the covariance matrix, one can either consider only diagonal matrices

$$\Sigma_i = \text{diag}(\sigma_{i,1}^2, \ldots, \sigma_{i,D}^2), \tag{5.30}$$

or consider matrices of the form

$$\Sigma_i = \mathbf{u}_i \mathbf{u}_i^T + \text{diag}(\sigma_{i,1}^2, \ldots, \sigma_{i,D}^2),$$

or use more advanced parameterizations such as the ones found in [82]. Let σ_i be the parameters for Σ_i, so that $\Sigma_i = \Sigma_i(\sigma_i)$. For $\mathbf{w}$, using some monotone bijective differentiable function $\phi : \mathbb{R} \to \mathbb{R}^+$ (for example, $\phi = \exp$), one can then consider the corresponding differentiable map $\Phi : \mathbb{R}^k \to \Delta_k$ as

$$\Phi(\nu_i) = \frac{\phi(\nu_i)}{\sum_{i=1}^k \phi(\nu_k)} = w_i(\nu_i). \tag{5.31}$$

The parameter of interest λ becomes $(\nu_1, \mu_1, \sigma_1, \ldots, \nu_k, \mu_k, \sigma_k)$, and

$$q_k(\theta) = \sum_{i=1}^k w_i(\nu_i) f_{\mathcal{N}(\mu_i, \Sigma(\sigma_i))}(\theta). \tag{5.32}$$

Thus, the ELBO objective becomes

$$\begin{aligned} \mathcal{L}(\lambda) &= \int \log \bar{g}(\theta) q_k(\theta) d\theta - \int \log(q_k(\theta)) q_k(\theta) d\theta \\ &= \sum_{i=1}^k w_i(\nu_i) \mathbb{E}_{\theta_i \sim \mathcal{N}(\mu_i; \Sigma(\sigma_i))} \left[\log \frac{\bar{g}(\theta_i)}{q_k(\theta_i; \lambda)} \right]. \end{aligned} \tag{5.33}$$

One can adapt the reparameterization trick to rewrite $\mathcal{L}(\lambda)$ in a manner suitable to stochastic gradient optimization. First notice that any gaussian random variable $X \sim \mathcal{N}(\mu, \Sigma)$ can be written as $X = \mu + AZ$, where $Z \sim \mathcal{N}(0, I)$ and A is some matrix such that $\Sigma = AA^T$. For instance, A can be the lower Cholesky factor of Σ, if the parameterization of Σ only supports positive-definite matrices. Then $A(\sigma)$ is differentiable [102, 73], and $s(\epsilon; \mu_i, \sigma_i) = \mu_i + A(\sigma)\epsilon$, and (5.33) is

approximated by Monte Carlo,

$$\begin{aligned}\mathcal{L}(\lambda) &= \sum_{i=1}^{k} w_i(\nu_i)\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log\frac{\bar{g}(s(\epsilon;\mu_i,\sigma_i))}{q_k(s(\epsilon;\mu_i,\sigma_i);\lambda)}\right] \\ &\approx \sum_{i=1}^{k} w_i(\nu_i)\left(\frac{1}{K_i}\sum_{k\in[K_i],\epsilon_{i,j}\sim\mathcal{N}(0,I)}\log\frac{\bar{g}(s(\epsilon_{i,j};\mu_i,\sigma_i))}{q_k(s(\epsilon_{i,j};\mu_i,\sigma_i);\lambda)}\right),\end{aligned} \tag{5.34}$$

which is an expression that can be differentiated to find an approximation for $\nabla\mathcal{L}(\lambda)$.

In principle there is no way to know how many mixtures are necessary to return a good approximation. However, one can,for each $i \in \mathbb{N}$, starting with $i = 1$, sequentially find q_i^* close to $\arg\max_{q_i\in\mathcal{Q}_i}\mathcal{L}(q)$, and go to sequentially from Q_i to $Q_{i+1} \supset Q_i$, until the variational approximation is good enough. However, this procedure runs into computational issues, namely the number of optimization parameters scale linearly with the number of mixtures k, in a non-convex problem with non-trivial gradient evaluation, whose cost also scales linearly with k. Hence, the cost of improving the mixtures this way quickly becomes rather large. We next present an approach that mitigates this problem, at the cost of making a greedy approximation, thus potentially more inefficient in the number of mixtures.

5.4.1 Boosting mixtures of gaussians

Boosting [33, 34, 35] is a standard technique in machine learning, usually used in classification problems, which tries to combine slighty better than chance algorithms, or *weak learners* in a reliable, accurate algorithm (a *strong learner*). The general framework is transferred to the problem of variational inference in concurrent works by [70] and [45], using increasing mixtures distributions.

In this setting, start first with some distribution $q_1 \in \mathcal{Q}_1$. Then, recursively,

given a proposal q_{i-1}^*, it is considered the proposal set

$$\begin{aligned}\mathcal{Q}_i = \mathcal{Q}_i(q_{i-1}) = \{&(1-w_i)q_{i-1} + w_i f(\cdot;\mu_i,\Sigma_i)| \\ &w_i \in [0,1], \mu_i \in \mathbb{R}^d, \Sigma_i \in \mathbb{R}^{d\times d}, \Sigma_i \geq 0\},\end{aligned} \tag{5.35}$$

and then some q_i is chosen from $\mathcal{Q}_i$, in a manner that $\mathcal{L}(q_i)$ is reasonably greater than the previous value $\mathcal{L}(q_{i-1})$. One straightforward approach is to seek $\lambda_i(w_i^*, \mu_i^*, \Sigma_i^*)$ such as the maximization objective becomes

$$\mathcal{L}_i(\lambda_i) := \mathcal{L}((1-w_i)q_{i-1} + w_i f(\cdot;\mu_i,\Sigma_i)), \tag{5.36}$$

which is the procedure proposed in [70]. In [45], it is shown that the KL divergence satisfies the conditions estabilished by [120] that ensures if

$$\mathcal{L}((1-w_i)i - 1 + w_i f_i) \geq \sup_{f\in Q_1, w\in[0,1]} \mathcal{L}((1-w)q_{i-1} + wf) - \epsilon_i, \tag{5.37}$$

as $\epsilon_i \to 0$, then

$$\lim_{i\to\infty} \sup_{q\in Q_\infty} L(q) - \mathcal{L}(q_i) = 0, \tag{5.38}$$

provided that every $q \in \mathcal{Q}_1$ is bounded from below. In [45], it is shown that the KL divergence satisfies those two conditions, if every $q \in \mathcal{Q}_1$ is assumed to be bounded from below by a positive constant. Although this is not the case for Gaussian distributions, it is argued that since in actual implementations the practitioner works with a bounded set of interest, the result holds in practice.

5.4.2 Gradient boosting mixture of gaussians

Another boosting proposal, due to [45], is to instead of trying to optimize jointly $(w_i^*, \mu_i^*, \Sigma_i^*)$ at each step, choosing first $f_i = f(\cdot;\mu_i,\Sigma_i)$, and then choose w_i as to maximize:

$$\begin{aligned}\mathcal{L}_i(w_i) &:= \mathcal{L}((1-w_i)q_{i-1} + w_i f_i) \\ &= \int \log \frac{\bar{g}(\theta)}{(1-w_i)q_{i-1}(\theta) + w_i f_i(\theta)} ((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))d\theta.\end{aligned} \tag{5.39}$$

Since the KL divergence is convex in q, the ELBO is concave, so minimizing w is easy, provided we can easily calculated $\mathcal{L}_i'(w_i)$. Since we have

$$\begin{aligned}\mathcal{L}_i'(w_i) &= \int \frac{\partial}{\partial w_i}\left(\log \frac{\bar{g}(\theta)}{(1-w_i)q_{i-1}(\theta)+w_i f_i(\theta)}((1-w_i)q_{i-1}(\theta)+w_i f_i(\theta))\right)d\theta \\ &= \int \log(\bar{g}(\theta))(f_i(\theta)-q_{i-1}(\theta))d\theta - \\ &\quad \int \log((1-w_i)q_{i-1}(\theta)+w_i f_i(\theta))(f_i(\theta)-q_{i-1}(\theta))d\theta,\end{aligned} \tag{5.40}$$

we can then approximate the derivative by Monte Carlo

$$\begin{aligned}\mathcal{L}_i'(w_i) =& \frac{1}{J}\sum_{j\in[J]\theta_j\sim f_i}(\log(\bar{g}(\theta_j)) - \log((1-w_i)q_{i-1}(\theta_j)+w_i f_i(\theta_j))) \\ &- \frac{1}{K}\sum_{\theta_k\in[K],\theta_k\sim q_{i-1}}(\log(\bar{g}(\theta_k)) - \log((1-w_i)q_{i-1}(\theta_k)+w_i f_i(\theta_k)))\,,\end{aligned} \tag{5.41}$$

and using it to maximize $\mathcal{L}_i'(w_i)$.

The question becomes then how to choose f_i. In [45], the technique of gradient boosting [36] is borrowed for this purpose, so that f_i is chosen as to minimize $\nabla D_{KL}(q_{i-1}||g)\cdot f$, where $\nabla D_{KL}(q||g)$ is the functional derivative of $D_{KL}(q||g)$ as a function of q. For $D_{KL}(q||p)$, we can use Taylor expansion to find the functional derivative

$$\begin{aligned}D_{KL}(q+\delta h||p) &= \int (q+\delta h)\log\frac{q+\delta h}{p} \\ &= \int q\left(\log\frac{q}{p}+\frac{\delta h/p}{q/p}+\mathcal{O}(\delta^2)\right)+\delta\int h\log\frac{q}{p}+\mathcal{O}(\delta^2) \\ &= D(q||p)+\delta\int\left(1+\log\frac{q}{p}\right)h+\mathcal{O}(\delta^2).\end{aligned} \tag{5.42}$$

Here the argument θ is omitted to simplify the expression. Hence

$$\begin{aligned}f_i = \arg\min_f \nabla D_{KL}(q_{i-1}||g)\cdot f &= \arg\min_f \int\left(1+\log\frac{q_{i-1}(\theta)}{g(\theta)}\right)f(\theta)d\theta = \\ &= \arg\min_f \int \log\frac{q_{i-1}(\theta)}{g(\theta)}f(\theta)d\theta.\end{aligned} \tag{5.43}$$

Since the parameterization for $f_i(\theta)$ allows degeneration to a point mass at $\arg\min_\theta \log(q_{i-1}(\theta)/g(\theta))$, and this point mass optimizes (5.43), further constraints over $f_i(\theta)$ are needed for getting a non-degenerate new basis function. In [45] (5.43) is regularized by the logarithm of the L_2 norm of $f_i(\theta)$

$$f_i = \arg\min_f \left(\int \log \frac{q_{i-1}(\theta)}{g(\theta)} f(\theta) d\theta + \frac{\lambda}{2} \log ||f||_2^2 \right) \tag{5.44}$$

while in [65], (5.43) is regularized by negative of the entropy of f, as a proxy for the regularization of its L_∞ norm. Since for $f(\theta) = \mathcal{N}(\theta|\mu, \Sigma)$, $\log ||f||_2^2 = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\log(2\pi)$ and $-\mathcal{H}(f) = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\log(2\pi e)$, both approaches are equivalent for mixture of Gaussians, yielding the maximization objective in relation to μ_i, Σ_i:

$$\begin{aligned} \text{RELBO}(\mu_i, \Sigma_i) = &\int \log(\bar{g}(\theta)) \mathcal{N}(\theta|\mu, \Sigma) d\theta - \int \log(q_{i-1}(\theta)) \mathcal{N}(\theta|\mu, \Sigma) d\theta + \\ &\frac{\lambda}{4} \log|\Sigma|, \end{aligned} \tag{5.45}$$

where λ is a regularization constant (the name RELBO comes from [65]). This constant may be set either as a fixed value or decaying as i increases, as to permit increasingly narrower distributions as the algorithm runs. For example, in [65] λ is set as $1/\sqrt{i+1}$. [7]. Finally, the gradients of (5.45) in relation to μ_i, Σ_i can be estimated by the reparameterization trick.

The resulting pseudo-algorithm is shown in Figure 1.

5.5 Using Bayesian Monte Carlo in Variational Inference

All the approaches previously presented suffers from one major flaw: the need for a large number of evaluations of the unnormalized posterior $\bar{g}$ by Monte

[7] In [45] it is argued that this objective is still somewhat hard to maximize, so it is proposed an heuristic based on a local Laplace approximation. This heuristic was previously explored by the author of this work, however it ran into implementation issues, and moreover, it does not exactly attend the desiderata in this work, namely as few function evaluations as possible

procedure VARIATIONALBOOSTING($\log \bar{g}, \mu_0, \Sigma_0$)
  ▷ μ_0, Σ_0 the are initial boosting values
  $w_0 := 1.0$
  for $t = 1, ..., T$ **do**
    $\mu_t, \Sigma_t := \arg\max RELBO(\mu_t, \Sigma_t)$ ▷ Using reparameterization
    $w_t := \arg\max \mathcal{L}_i(w_i)$ ▷ Using $\mathcal{L}_t'(w_t)$ for gradient descent
    for $j = 0, ..., t-1$ **do**
      $w_j \leftarrow (1 - w_t)w_j$
    end for
  end for
  return $\{(\mu_t, \Sigma_t, w_t)\}_{t=1}^{T}$
end procedure

Algorithm 1: Variational boosting algorithm.

Carlo methods, in order to have a good estimation of the gradient. This is the same problem of integration that Bayesian Monte Carlo tries to solve, so one can try to use this technique when $\bar{g}$ is expensive to evaluate. In [2], it is developed a method called Variational Bayesian Monte Carlo (VBMC) that hinges on exactly this idea.

Consider a parameterized variational proposal $q(\theta; \lambda)$, and the corresponding ELBO for $\bar{g}(\theta)$:

$$\mathcal{L}(\lambda) = \mathcal{L}_{\bar{g}}(\lambda) = \int \log \bar{g}(\theta) q(\theta; \lambda) d\theta - \int \log(q(\theta; \lambda)) q(\theta; \lambda) d\theta.$$

As in Chapter 4, setting as prior for $\log \bar{g}(\theta)$ the Gaussian process $GP(m, k)$, and given set of evaluations $\mathcal{D}_0 = \{(x_i, f(x_i))\}_{i=1}^{N}$, $\mathcal{L}_{\bar{g}}(\lambda)$ can be replaced by the Gaussian distributed random variable

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(\lambda) := \mathcal{L}_{\bar{g}_{\mathcal{D}}}(\lambda) := Z_{\mathcal{D}}(\lambda) - \int \log(q(\theta; \lambda)) q(\theta; \lambda) d\theta, \\ Z_{\mathcal{D}}(\lambda) := \int \log \bar{g}_{\mathcal{D}}(\theta) q(\theta; \lambda) d\theta, \end{aligned} \tag{5.46}$$

with mean

$$\begin{aligned}\bar{\mathcal{L}}_{\mathcal{D}}(\lambda) := \mathbb{E}\left[\mathcal{L}_{\log \bar{g}_{\mathcal{D}}}(\lambda)\right] &= \mathbb{E}[Z_{\mathcal{D}}(\lambda)] - \int \log(q(\theta;\lambda))q(\theta;\lambda)d\theta \\ &= \int \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]q(\theta;\lambda)d\theta - \int \log(q(\theta;\lambda))q(\theta;\lambda)d\theta \\ &= \mathcal{L}_{\mathbb{E}[\log \bar{g}_{\mathcal{D}}]}(\lambda),\end{aligned} \tag{5.47}$$

and variance

$$\mathrm{Var}(\mathcal{L}_{\mathcal{D}}(\lambda)) = \mathrm{Var}(Z_{\mathcal{D}}(\lambda)), \tag{5.48}$$

and $\mathbb{E}[Z_{\mathcal{D}}(\lambda)]$ and $\mathrm{Var}[Z_{\mathcal{D}}(\lambda)]$ being given by (4.5). This approach enjoy the same benefits as the BMC approach:

- If the evaluation of $\bar{g}$ is expensive, it is not feasible to perform a Monte Carlo estimation of $\int \log \bar{g}(\theta)q(\theta;\lambda)d\theta$ at each step in optimizing the ELBO.
- The function $\bar{g}$ does not need to be differentiable, and its evaluations may be noisy (although this last case is not considered in this work, the extension is straightforward).

In case $q(\theta;\lambda)$ is a mixture of i Gaussians $q_i(\theta;\lambda)$, as in [2], the methods presented in 5.4 and the one in Chapter 4, by the discussion in Section 4.2, given a suitable kernel k, $\mathbb{E}[Z_{\mathcal{D}}(\lambda)]$ and $\mathrm{Var}[Z_{\mathcal{D}}(\lambda)]$ can be easily treated. In [2], k is the SQE kernel (3.10) with outputscale r_0 and lengthscales $l_1, \ldots, l_D$, which by (4.15) makes $\mathbb{E}[Z_{\mathcal{D}}(\lambda)]$ and $\mathrm{Var}[Z_{\mathcal{D}}(\lambda)]$ both analytical.

The entropy term $-\int \log(q(\theta;\lambda))q(\theta;\lambda)d\theta$ does not have in general a closed form, and must still be treated. Approximating $-\log(q(\theta;\lambda))$ by BMC is possible, but since this term changes with λ, this will be costly. However, since $\log q(\theta;\lambda)$ is easy to evaluate, by construction, the entropy term can be treated with the reparameterization trick without much additional cost. Other possible approach to deal with this term, when using mixture of Gaussians, is to use one of the approximations

proposed in [52].

An important point is that, since, by in the first equation of (4.5), only $\mathbf{z}$ and $\int m(\theta)q(\theta;\lambda)d\theta$ depends on $q(\theta;\lambda)$, one can rewrite

$$\begin{aligned} \int \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]q_k(\theta;\lambda)d\theta &= M(\lambda) + \mathbf{z}^T\mathbf{w} \\ \mathbf{w} &= K^{-1}\mathbf{y} \\ M(\lambda) &= \int m(\theta)q(\theta;\lambda)d\theta \\ \mathbf{z}_i &= \int k(x,x_i)q(\theta;\lambda)dx. \end{aligned} \tag{5.49}$$

This way, with $\mathbf{w}$ being previously computed, the computation of $\bar{\mathcal{L}}_{\mathcal{D}}(\lambda)$ can be calculated in $\mathcal{O}(N)$ time for each λ, while $\mathrm{Var}(\mathcal{L}_{\mathcal{D}}(\lambda))$ can be calculated in $\mathcal{O}(N^2)$. The new variational objective becomes then simply $\bar{\mathcal{L}}_{\mathcal{D}}(\lambda)$.

5.5.1 Quadratic mean function

With the objective $\bar{\mathcal{L}}_{\mathcal{D}}(\lambda)$, the actual distribution that is being approximated by mixture of Gaussians is proportional to $\exp \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$. From the variational inference approximation's point of view, the only information that it has on the original distribution $\bar{g}$ is by its GP approximation.

This raises some issues: if the mean function of the GP prior $m(\theta)$ is either zero or a constant, and the kernel is one that decays to zero as $|x - x'|$ goes to infinity, such as (3.10), (3.12) or (3.14), then $\exp \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$ is not integrable, so it cannot define an unnormalized probability distribution. Hence, care must be taken with the mean function $m(\theta)$.

One approach is to simply let $m(\theta)$ have a very low value, so that in practice it resembles enough a probability distribution so that, for a reasonable number of

mixtures k, convergence issues does not appear. This is the default approach used in the developed algorithm presented in Chapter 4. An alternative approach, used in [2], is to change the mean function so that $\exp(m(\theta))$, thus $\exp \log \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$ is an unnormalized probability distribution, while having the integral $\int m(\theta) q_k(\theta) d\theta$ being analytically tractable. In [2], the following mean function is proposed:

$$m_Q(\theta; l, c) = -\frac{1}{2} \sum_{i=1}^{D} \frac{(\theta_i - c_i)^2}{l_i^2}, \tag{5.50}$$

which then both makes $\exp(m_Q(\theta; l, c))$ integrable and yields, for $q_i(\theta)$ being a mixture of Gaussians,

$$\int m_Q(\theta; l, c) q_k(\theta) d\theta = -\frac{1}{2} \sum_{j=1}^{k} \sum_{i=1}^{D} \left[\frac{(c_i - \mu_{k,i})^2}{l_i^2} + \frac{\Sigma_{k,i,i}}{l_i^2} \right]. \tag{5.51}$$

This comes immediately from the expectation of quadratic forms of normal distribution formula [81].

5.5.2 Remarks on acquisition functions for VBMC

As in Bayesian Monte Carlo, a question that arises is one of active sampling. Notice that, fixed evaluations $\mathcal{D}$, and assuming a fixed set of mixtures of k Gaussians $\mathcal{Q}_k$, associated with a fixed set of parameters Λ_k, and letting

$$\lambda^*(\mathcal{D}) = \arg\min_{\lambda} \mathcal{L}_{\mathcal{D}}(\lambda), \tag{5.52}$$

the final ELBO is a random variable $\mathcal{L}_{\mathcal{D}}(\lambda^*(\mathcal{D}))$, which is normally distributed with variance $\mathrm{Var}(Z_{\mathcal{D}}(\lambda^*))$. Since we want to have a high degree of certainty about the variational approximation's quality, one wants the final ELBO objective to have as lower variance as possible, since it is a measure of its uncertainty.

In this setting, one option would be, in a greedy approach, to choose x_{N+1}, so that, expanding $\mathcal{D}$ to $\mathcal{D}'(x, \log \bar{g}(\theta_{N+1})) = \mathcal{D} \cup \{(x_{N+1}, \log \bar{g}(\theta_{N+1}))\}$, it minimizes the variance of $\mathcal{L}_{\mathcal{D}'}(\lambda^*(\mathcal{D}'))$. Since $\log \bar{g}(\theta_{N+1})$ is not known in advance, one could

substitute this instead for

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}'}\left[\mathrm{Var}(Z_{\mathcal{D}'}(\lambda^*(\mathcal{D}'))\right] \\ &\mathcal{D}' = \mathcal{D} \cup \{(\theta_{N+1}, \log \bar{g}_{\mathcal{D}}(\theta_{N+1}))\}. \end{aligned} \tag{5.53}$$

However, it is not clear how to work with this random variable, and minimize a measure of its uncertainty. However, this intuition helps with constructing heuristic acquisition functions to seek θ_{N+1}.

5.5.2.1 Uncertainty sampling and prospective prediction

One option in order to tackle this problem is to consider the current variational proposal $q_k(\theta;\lambda)$ fixed, and consider the minimization objective

$$\alpha^{\mathcal{D}}_{\mathrm{VR}}(\theta_{N+1}) = \mathrm{Var}\left(Z_{\mathcal{D}'(\theta_{N+1})}(\lambda)\right). \tag{5.54}$$

However, since this is an expensive minimization objective, a proxy for this approach is to consider the integrand of

$$\int \log \bar{g}_{\mathcal{D}}(\theta) q_k(\theta;\lambda) d\theta,$$

and seek the integrand maximum variance instead, resulting in the *uncertainty sampling* maximization objective

$$\alpha^{\mathcal{D}}_{\mathrm{US}}(\theta_{N+1}) = k_{\mathcal{D}}(\theta_{N+1}, \theta_{N+1}) q_k(\theta_{N+1};\lambda)^2. \tag{5.55}$$

The above objective puts much weight in the current variational proposal, which may hinder exploration. Considering that future proposals will seek regions with high posterior density, the other objective proposed in [2] is formed by multiplying $\alpha^{\mathcal{D}}_{\mathrm{US}}(\theta_{N+1})$ by an exploration factor for high posterior denstiies $\exp(m_{\mathcal{D}}(\theta_{N+1}))$. This result in the *prospective prediction* maximization objective,

$$\alpha^{\mathcal{D}}_{\mathrm{PROP}}(\theta_{N+1}) = k_{\mathcal{D}}(\theta_{N+1}, \theta_{N+1}) \exp(m_{\mathcal{D}}(\theta_{N+1})) q_k(\theta_{N+1};\lambda)^2. \tag{5.56}$$

It is reported in [2] that the prospective prediction objective results in better ap-

proximations than the uncertainty sampling one. The objectives (5.55) and (5.56), along with some newly proposed ones ((6.13),(6.14) and (6.15)), are used in the algorithm developed in this work, to be shown in the next chapter.

6 BOOSTED VARIATIONAL BAYESIAN MONTE CARLO

In this chapter, it is presented a modification of the Variational Bayesian Monte Carlo approach presented in Section 5.5, using ideas of boosting presented in 5.4.1, along with some other minor modifications of the previous approach.

6.1 Boosting Variational Bayesian Monte Carlo

The idea underlying this approach is rather simple: instead of using a GP surrogate model for doing variational inference with mixtures of Gaussians as in 5.5, use it for variational inference with the gradient boosting approach discussed in section 5.4.1.

To expand on this, consider again the algorithm in Figure 1. In line 5, the objective (5.45) (here called *f-step*) consists of three terms

$$\begin{aligned}\text{RELBO}(\mu_i, \Sigma_i) = &\int \log(\bar{g}(\theta))\mathcal{N}(\theta|\mu, \Sigma)d\theta \\ &- \int \log(q_{i-1}(\theta))\mathcal{N}(\theta|\mu, \Sigma)d\theta \\ &+ \frac{\lambda}{4}\log|\Sigma|,\end{aligned}$$

while in line 6, for the objective (5.39) (here called *w-step*), the gradient $\mathcal{L}_i'(w_i)$

consists of

$$\begin{aligned}\mathcal{L}_i'(w_i) = &\int \log(\bar{g}(\theta)) f_i(\theta) \\ &- \int \log(\bar{g}(\theta)) q_{i-1}(\theta) \\ &+ \int \log((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))(f_i(\theta) - q_{i-1}(\theta))d\theta\end{aligned}$$

Then, in the same vein of the VBMC method, set a $GP(m, k)$ prior for $\log \bar{g}$, and given evaluations $\mathcal{D}_0 = \{(x_i, f(x_i))\}_{i=1}^N$, substitute $\log \bar{g}$ for $\mathbb{E} \log \bar{g}_{\mathcal{D}}$. Since all the terms involving $\log \bar{g}$ are expectations of either Gaussian distributions or mixture of Gaussian distributions, then one can approximate those by Bayesian Monte Carlo, as seen in (4.12) and (4.15), resulting in the maximization objective for the f-step:

$$\begin{aligned}\text{RELBO}_{\mathcal{D}}(\mu_i, \Sigma_i) = &\int \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)] \mathcal{N}(\theta|\mu_i, \Sigma_i) d\theta - \\ &\int \log(q_{i-1}(\theta)) \mathcal{N}(\theta|\mu_i, \Sigma_i) d\theta + \frac{\lambda}{4} \log|\Sigma_i|,\end{aligned} \tag{6.1}$$

and the maximization objective for the w-step

$$\begin{aligned}\mathcal{L}_{i,\mathcal{D}}(w) = &\int \log \bar{g}_{\mathcal{D}}(\theta)((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))d\theta - \\ &\int \log((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))d\theta\end{aligned} \tag{6.2}$$

with gradient

$$\begin{aligned}\mathcal{L}_{i,\mathcal{D}}'(w) = &\int \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)] f_i(\theta) - \int \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)] q_{i-1}(\theta) + \\ &\int \log((1-w_i)q_{i-1}(\theta) + w_i f_i(\theta))(f_i(\theta) - q_{i-1}(\theta))d\theta,\end{aligned} \tag{6.3}$$

where

$$f_i(\theta) = \mathcal{N}(\theta|\mu_i, \Sigma_i), \quad q_{i-1}(\theta) = \sum_{k=0}^{i-1} w_k \mathcal{N}(\theta|\mu_k, \Sigma_k) \tag{6.4}$$

The integrals not involving $\log \bar{g}_{\mathcal{D}}$ can be easily approximated by the reparameterization trick, while $\log|\Sigma|$ is trivial to compute. This readily results in an algorithm for variational inference, given fixed evaluations points $\{x_n\}_{n=1}^N$, shown in Figure 2. Here it is named *Naive Boosted Variational Bayesian Monte Carlo* (Naive BVBMC).

procedure NAIVEBVBMC($\log \bar{g}, \mu_0, \Sigma_0, \{x\}_{n=1}^N$)
  ▷ μ_0, Σ_0 the are initial boosting values
  for $n = 1, ..., N$ **do**
    $y_n := \log \bar{g}(x_n)$
  end for
  $\mathcal{D} := \{(x_n, y_n)\}_{i=1}^N$
  GPModel := PosteriorGP(m,k_{RBF},$\mathcal{D}$)
  GPModel.MaximizeLogLikelihood() ▷ Using (3.16)
  $\mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$:= GPModel.$m_{\mathcal{D}}$
  $w_0 := 1.0$
  for $t = 1, ..., T$ **do**
    ▷ Using BMC and reparameterization
    $\mu_t, \Sigma_t := \arg\max RELBO_{\mathcal{D}}(\mu_t, \Sigma_t)$
    $w_t := \arg\max \mathcal{L}_{t,\mathcal{D}}(w_i)$ ▷ Using $\mathcal{L}'_{t,\mathcal{D}}(w_t)$ for gradient descent
    for $j = 0, ..., t-1$ **do**
      $w_j \leftarrow (1 - w_t) w_j$
    end for
  end for
  return $\{(\mu_t, \Sigma_t, w_t)\}_{t=1}^T$
end procedure

Algorithm 2: Naive boosted variational bayesian monte carlo

6.1.1 Practical issues

The algorithm in Figure 2 run into some practical issues, described below, that requires fixes of an heuristic nature. Here we present the heuristics that were used to make the algorithm stable. The resulting algorithm in found in 3.

6.1.1.1 RELBO stabilization

By letting $r_{\mathcal{D}}(\theta) := \exp \mathbb{E} \log_{\mathcal{D}}(\theta)$, consider again the maximization objective (6.1), rewritten as

$$\text{RELBO}_{\mathcal{D}}(\mu_i, \Sigma_i) = \int \log \left(\frac{r_{\mathcal{D}}(\theta)}{q_{i-1}(\theta)} \right) \mathcal{N}(\theta; \mu_i, \Sigma_i) d\theta + \log |\Sigma_i|. \tag{6.5}$$

Assume Σ_i fixed for now, so that the only term being optimized is μ_i. Then, what the maximizer "wants" to do is set μ_i in a place that $\mathcal{N}(\theta; \mu_i, \Sigma_i)$ allocates probability mass where $\log r_{\mathcal{D}}(\theta)/q_{i-1}(\theta)$ is large. Now, consider the tail behavior of this quantity. We have that $q_{i-1}(\theta) \approx C_1 \exp(-||A_1(\theta - c_1)||_2^2)$ on the tail, while either $r_{\mathcal{D}}(\theta) \approx \exp(-C)$, if $m(\theta) = -C$, or $r_{\mathcal{D}}(\theta) \approx \exp(-||A_2(\theta - c_2)||_2^2)$, in case of $m(\theta) = -||A_2(\theta - c_2)||_2^2$. In the first case, then $\log r_{\mathcal{D}}(\theta)/q_{i-1}(\theta) \to \infty$ as $\theta \to \infty$, while in the second case, whether this happens depends on the relation between A_1 and A_2. However, if $\log r_{\mathcal{D}}(\theta)/q_{i-1}(\theta) \to \infty$ as $\theta \to \infty$, then the maximizer will try to get μ_i to ∞, thus resulting in bad proposals, that ends up with negligible weights w_i.

One approach to deal with this problem is to replace the term $\log r_{\mathcal{D}}/q_{i-1}(\theta)$ by $\log r_{\mathcal{D}}/(q_{i-1}(\theta) + \delta_D)$, where δ_D is a small positive constant. This approach is similar to the one in [45], where a regularizer is added to both numerator and denominator, although it is done in the approximate heuristic proposed there. Thus, the f-step maximization objective becomes

$$\text{RELBO}_{\mathcal{D}}^{\delta_D}(\mu_i, \Sigma_i) = \int \log \left(\frac{r_{\mathcal{D}}(\theta)}{q_{i-1}(\theta) + \delta_D} \right) \mathcal{N}(\theta; \mu_i, \Sigma_i) d\theta + \log |\Sigma_i|. \tag{6.6}$$

In the current work, $\delta_D = e^{-20}$ by default.

6.1.1.2 Output scaling

One issue that arises with unnormalized log-densities is that one has no control on its scale. Fortunately, scaling the output can be done easily. Given data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, suppose one makes a linear transformation $\tilde{y}_i = \frac{y_i - b}{a}$, and, letting $\tilde{\mathcal{D}} = \{x_i, \tilde{y}_i\}$, consider the GP posterior $f_{\tilde{\mathcal{D}}} \sim GP(m_{\tilde{\mathcal{D}}}, k_{\tilde{\mathcal{D}}})$. Then, the random function $af_{\tilde{\mathcal{D}}} + b$ is GP distributed, and

$$\int (af_{\tilde{\mathcal{D}}} + b)(x)p(x)dx \tag{6.7}$$

is a GP random variable, with

$$\mathbb{E}\left[\int (af_{\tilde{\mathcal{D}}} + b)(x)p(x)dx\right] = a\mathbb{E}\left[\int f_{\tilde{\mathcal{D}}}(x)p(x)dx\right] + b, \tag{6.8}$$

and variance

$$\mathrm{Var}\left(\int (af_{\tilde{\mathcal{D}}} + b)(x)p(x)dx\right) = a^2\mathrm{Var}\left(\int f_{\tilde{\mathcal{D}}}(x)p(x)dx\right). \tag{6.9}$$

This implies that one can do BMC, hence also VBMC and BVBMC using affinely scaled output variables, without difficulty.

Two heuristic for scaling were implemented, the *normalize* heuristic, given by

- Letting $y_i = \log g(x_i)$, calculating the sample mean $m_y = \frac{1}{N}\sum_{i=1}^N y_i$ and the (unbiased) sample standard deviation $\sigma_y = \sqrt{\frac{1}{N-1}\sum_{i=1}^N (y_i - m_y)}$.
- Transforming the output $\tilde{y}_i = (y_i - m_y)/\sigma_y$.
- Train a GP (along with hyperparameters) on $\tilde{\mathcal{D}} = \{x_i, \tilde{y}_i\}_{i=1}^N$.
- Substituting, when needed, $\log g_{\mathcal{D}}(x)$ (and their integrals) for $\sigma_y \log g_{\tilde{\mathcal{D}}}(x) + \mu_y$

and the *zeromax* heuristic, where

- Letting $y_i = \log g(x_i)$, calculating the sample maximum $m_y = \max\{y_i\}_{i=1}^{N}$ and making $\sigma_y = 1$.
- Do the other steps as in *normalize* heuristic.

In toy problems presented in the next section, both heuristics worked well, but for more complex problems the *zeromax* heuristic was found to be more stable.

6.1.1.3 Component initialization

A question that still wasn't addressed is on how to choose the first boosting component. Two options were tested:

- Initializing the first mixture component with zero mean and a large covariance.
- Choosing the first component by maximizing the ELBO between the component and GP surrogate.

In general, the second method was chosen in this work, although performance in both cases was comparable.

6.1.1.4 Component pruning

When running the BVBMC algorithm, many mixture components may end up with negligible weights. This may both increase the computational cost of the algorithm, and hinder the performance of joint parameter updating (explained in 6.1.1.6). In order to diminish this problem, an weight threshold β may be established, so that, if $(w_1, \ldots, w_N)$ are the current mixture weights, every i-th mixture

components with $w_i < \beta$ is removed, with the remainder weights being renormalized to one. This pruning may be done either every iteration, or only when doing joint parameter updating (for which pruning proved necessary).

6.1.1.5 Mean functions

As discussed in Section 5.5.1, one option for the mean function set $m(\theta) = m_Q(\theta; l, c)$ as in (5.50). However, one issue that arose in this work is that optimizing l, c by maximizing the log-likelihood often resulted in very large values (in absolute value) for l and c, destabilizing the algorithm.

In this light, a different approach for the mean is proposed, by letting

$$m_F(\theta) = C \tag{6.10}$$

where C is a large negative constant, that is *fixed before hyperparameter optimization*. Strictly speaking, in this case $\exp \mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$ is not anymore a probability distribution. However, if C is sufficiently low, it resembles a probability distribution enough so that the algorithm works well in practice.

Since C is not set by optimizing the log-likelihood, one must decide on how to set it. If the data was scaled beforehand using *zeromax*, some good options that worked well in practice were $C = -10, -20, -30$. For scaling using *normalize*, the following way to set C is proposed:

$$C = \tilde{y}_{\min} - K_C(\tilde{y}_{\max} - \tilde{y}_{\min}), \tag{6.11}$$

where $\tilde{y}_{\max}$ and $\tilde{y}_{\min}$ are the maximum and minimum of the normalized output values, and $K_C > 0$ is a constant. In this work, $K_C = 1$ was found to be a good choice.

6.1.1.6 Periodic joint parameter updating

Sometimes, boosting may get "stuck", in the sense that the variational proposal takes too long to improve. In that case, one may desire to periodically update all variational parameters in the sense of the original VBMC algorithm. However, since this optimization is expensive, it should be used sparsely, for example every 10 boosting steps, or every 3 steps when boosting does not show improvement, when checking the ELBO.

6.1.1.7 Other kernel functions

As discussed in Section 4.2.3, tensor product of one-dimensional kernels can be easily integrated with Bayesian Monte Carlo with diagonal covariance Gaussian distributions. In this light, by letting $k_{\text{Matern},\nu}(|x - x'|; l)$ be the 1-d Matérn kernel, the product of Matérn kernels is defined

$$k_{\text{PMat},\nu}(x, x'; \theta, l) = \theta \prod_{d=1}^{D} k_{\text{Matern},\nu}(|x_i - x_i'|; l_d). \tag{6.12}$$

It must be noted that the product of Matérn kernels is not the Matérn kernel for $d > 1$, although it is a stationary kernel.

In experiments, the product of Matérn kernels tended to be more stable than the squared exponential kernel, particularly for $\nu = 1/2, 3/2$, although for $\nu = 5/2$ the cases where it became unstable were rare. This may be due to the fact that the squared exponential kernels assumes far too much smoothness in its defined GP, hence estimating large oscillations outside the domain of interest, resulting in spurious multimodality.

6.1.2 Other acquisition functions for active evaluation

One problem that may arises in prospective prediction is that the term $\exp(m_{\mathcal{D}}(\theta_{N+1}))$ may become unstable for high values. Hence, one change proposed in this work is to substitute exp for the function $\text{softplus}(x) = \log(1 + \exp(x))$, resulting in the *soft prospective prediction*

$$\alpha^{\mathcal{D}}_{PP}(\theta_{N+1}) = k_{\mathcal{D}}(\theta_{N+1}, \theta_{N+1})\text{softplus}(m_{\mathcal{D}}(\theta_{N+1}))q_k(\theta_{N+1}; \lambda)^2. \tag{6.13}$$

Another option is to disregard the current proposal altogether, and instead noticing that, ideally, the proposal is going to approximate the unnormalized posterior $\bar{g} = \exp \log \bar{g}$. Hence, one can take as an inspiration the warping approach in Section 4.3 and use the *moment-matched log transform* objective

$$\alpha^{m}_{MMLT}(x_{m+1}) = e^{2m_{\mathcal{D}}(x)+k_{\mathcal{D}}(x,x)} \left(e^{k_{\mathcal{D}}(x,x')} - 1\right). \tag{6.14}$$

Finally, one can adapt the uncertainty sampling approach to the warped approach, resulting in the *prospective moment-matched log transform* objective

$$\alpha^{m}_{MMLT_P}(x_{m+1}) = e^{2m_{\mathcal{D}}(x)+k_{\mathcal{D}}(x,x)} \left(e^{k_{\mathcal{D}}(x,x')} - 1\right) q_k(\theta_{N+1}; \lambda)^2. \tag{6.15}$$

In general, the best performing acquisition functions were (5.56) and (6.14), and best results were obtained alternating between the two, either cyclically or randomly.

6.2 Implementation

The algorithm was implemented in Python, mainly using the PyTorch package [79]. The reason for using PyTorch is that it combines strong automatic differentiation capabilities, allowing to avoid computing derivatives by hand, with easy usability when compared to other packages with similar strenghts such as TensorFlow. Since derivatives are necessary for the many inner optimization procedures in the algorithm,automatic differentiation greatly facilitated development.

Code can be found in `https://github.com/DFNaiff/BVBMC`. An example usage of the package is shown in Figure 6.1, with associated densities shown in 6.2. [1]

6.2.1 Backpropagation

Usually, in an algorithm with so many optimization steps, one would either have to worry about carefully keeping control of gradients, or resort to gradient free optimization. However, backpropagation [43], a technique from the class of automatic differentiation algorithms, allows the computer to provide arbitrary derivatives of functions, provided the forward history of the function is tracked.

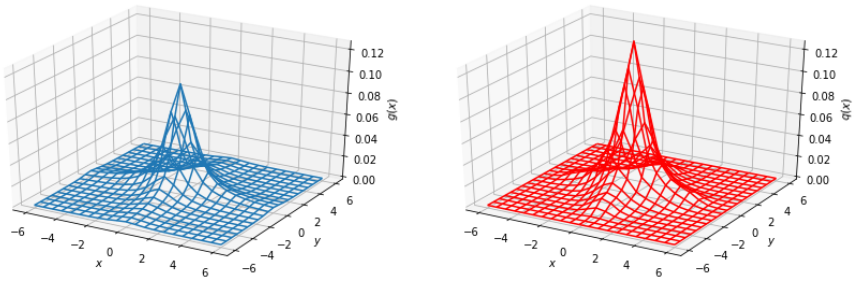
A formal discussion on backpropagation is beyond the scope of this work. A good reference can be found in [43]. Greatly simplifying, backpropagation hinges on the fact that almost every function in a computer comes from composition, so the chain rule can be applied. So, if the forward history of the composition is stored in a suitable data structure (a *computational graph*), then, given the derivatives of those unit compositions, chain rule can be applied to get any derivative, without ever needing for then to being calculated by hand.

Backpropagation is ubiquitous in deep learning , and almost all packages dealing with neural networks implement some form of it. In fact, most of then are backpropagation packages, in which neural networks are build on top of it. Some of the most popular are Theano [109], TensorFlow [29] and PyTorch [79]. Of those, PyTorch is the one with easiest usability, so it was chosen to implement the BVBMC package.

[1] The author intends to change the package name used in development, *variational_boosting_bmc*, to something akin to *bvbmc*, but the change wasn't made at this document writing.

```
#Import necessary packages
import torch #PyTorch package
from variational_boosting_bmc import VariationalBoosting #BVBMC package

#Approximating unnormalized 2-d Cauchy
def logjoint(theta):
   return torch.sum(-torch.log(1+theta**2))

#Set up parameters
dim=2 #Dimension of problem
samples = torch.randn(20,dim) #Initial samples
mu0 = torch.zeros(dim) #Initial mean
cov0 = 20.0*torch.ones(dim) #Initial covariance
acquisition = "prospective" #Acquisition function

#Initialize algorithm
vb = VariationalBoosting(dim,logjoint,samples,mu0,cov0)
vb.optimize_bmc_model() #Optimize GP model
vb.update_full() #Fit first component

#Training loop
for i in range(100):
   _ = vb.update() #Choose new boosting component
   vb.update_bmcmodel(acquisition=acquisition) #Choose new evaluation
   vb.cutweights(1e-3) #Weights prunning
   if ((i+1)%20) == 0:
      vb.update_full(cutoff=1e-3) #Joint parameter updating

vb.save_distrib("finaldistrib") #Save distribution
```

Figure 6.1: Usage of the BVBMC Python package, approximating a product of Cauchy distributions.



(a) True density (b) Estimated density

Figure 6.2: True density (left) and estimated density (right), found by running code in Figure 6.1. Generating code can be found in `https://github.com/DFNaiff/Dissertation/blob/master/illustrations_dissertation/examplebvbmc.py`.

The usefulness of backpropagation and their related packages in general escapes the community outside deep learning. An important exception is in development of probability programming languages (PPL), whose many are built on top of deep learning packages, as in PyMC3 [96], Edward [112] and Pyro [11], built on top of Theano, TensorFlow and PyTorch, respectively.

procedure BOOSTEDVBMC$_0$($\log \bar{g}, \mu_0, \Sigma_0, \{x\}_{n=1}^N, \delta_D$)
  ▷ μ_0, Σ_0 the are initial boosting values
  for $n = 1, ..., N$ **do**
    $y_n := \log \bar{g}(x_n)$
  end for
  $\mathcal{D}_0 := \{(x_n, y_n)\}_{i=1}^N$
  Make $\tilde{\mathcal{D}}$ from $\mathcal{D}$. Hold m_y and σ_y. ▷ Section 6.1.1.2
  GPModel := PosteriorGP(m,k,$\mathcal{D}$)
  GPModel.MaximizeLogLikelihood() ▷ Using (3.16)
  $\mathbb{E}[\log \bar{g}_{\mathcal{D}}(\theta)]$:= GPModel.$m_{\mathcal{D}}$
  $\alpha_0 := 1.0$
  $(\mu_0, \Sigma_0) \leftarrow \arg\max \mathcal{L}_{\mathcal{D}}(\lambda)$ ▷ Section 6.1.1.2
  for $t = 1, ..., T$ **do**
    $x' := \arg\max \alpha^m(x)$ ▷ Some acquisition function
    $y' = \log \bar{g}(x')$
    $\tilde{y}' = \frac{y - m_y}{\sigma_y}$
    $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \{(x', \tilde{y}')\}$
    ▷ Using BMC and reparameterization
    $\mu_t, \Sigma_t := \arg\max RELBO_{\mathcal{D}}^{\delta_D}(\mu_t, \Sigma_t)$
    $w_t := \arg\max \mathcal{L}_{t,\mathcal{D}}(w_i)$ ▷ Using $\mathcal{L}'_{t,\mathcal{D}}(w_t)$ for gradient descent
    for $j = 0, ..., t-1$ **do**
      $w_j \leftarrow (1 - w_t) w_j$
    end for
    if conditions met **then**
      $\lambda = \{(\mu_j, \Sigma_j, w_j)\}_{j=1}^t \leftarrow \arg\max \mathcal{L}_{\mathcal{D}}(\lambda)$ ▷ Section 6.1.1.6
    end if
  end for
  return $\{(\mu_t, \Sigma_t, w_t)\}_{t=1}^T$
end procedure

Algorithm 3: Boosted Variational Bayesian Monte Carlo

7 EXPERIMENTS

In this chapter, BVBMC is applied in a number of examples, as to show the algorithm behavior in different scenarios. The examples consists of estimating continuous probability densities in up to 10 dimensions [1].

7.1 1-d Mixture of Gaussians

As a showcase example, a one-dimensional mixture of Gaussians is considered

$$f(x) = \sum_{i=1}^{12} w_i \mathcal{N}(x; \mu_i, \sigma_i^2),$$

with $w_i = \frac{1}{12}$, $\mu_i \sim \mathcal{N}(0, \sqrt{5})$ and $\sigma_i^2 = 1$. This results in a many-peaked distribution, with mean $\mu_0 = -1.6585$ and variance $\sigma_0^2 = 25.0316$, whose density is shown in Figure 7.1.

In each example, the quality of the approximation by the BVBMC algorithm is measured, by calculating both the difference between the estimated mean μ and the true mean μ_0, in $\log_{10} |\mu - \mu_0|$, and the difference between the estimated variance σ^2 and the true variance σ_0^2, in $\log_{10}(|\sigma^2 - \sigma_0^2|/|\sigma_0^2|)$. Code for this section can be found in `https://github.com/DFNaiff/Dissertation/tree/master/tests_dissertation/illustrative`.

[1] In more than 10 dimensions, performance was very poor. What was seen is that the algorithm got "stuck" in its first component proposal, and could not find new components to be proposed with acceptable weights.

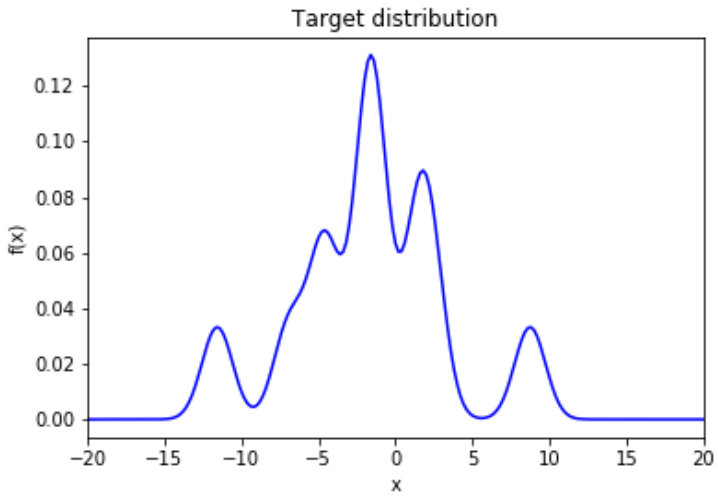


Figure 7.1: Target distribution.

7.1.1 Passive evaluation

In the first two examples, the target evaluations of $\log f(x)$ were done in a uniform grid from -20 to 20 with 51 points, resulting in the GP replicating almost exactly the target distribution. This is done to illustrate the algorithm's capacity to generate good approximate distributions, if the GP approximates closely the (unnormalized) target distribution. Output scaling was done by normalizing, as discussed in Section 6.1.1.2. The GP mean was chosen to be constant, setting it with the heuristic corresponding to the *normalize* scaling, as discussed in Section 6.1.1.5.

7.1.1.1 Influence of kernel in approximation

It was tested the kernel influence on the final approximation. For this, the tested kernels were $k_{\text{PMat},1/2}$, $k_{\text{PMat},2/2}$, $k_{\text{PMat},5/2}$ and k_{SQE}. The algorithm was run for 50 iterations, with joint parameter updating done every 10 steps. The results

are shown in Figure 7.2.

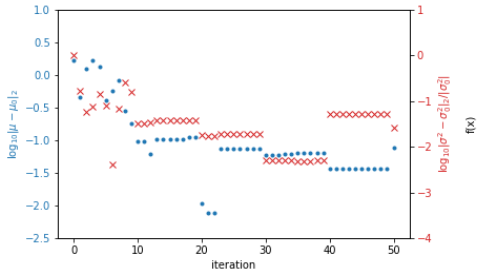
There is an interesting behavior to be seen, in that the GP approximation for the kernels $k_{\text{PMat},1/2}$ and $k_{\text{PMat},3/2}$ were considerably more accurate than with kernels $k_{\text{PMat},5/2}$ and k_{SQE}. However, this accuracy in the GP approximation does not reflect in a better posterior approximation, that are seen with $k_{\text{PMat},5/2}$ and k_{SQE}. This may be due to the fact that the first two kernels makes for GP approximations that are too rough, making it difficult to be approximated by the BVBMC algorithm.

7.1.1.2 Influence of training routine in approximation

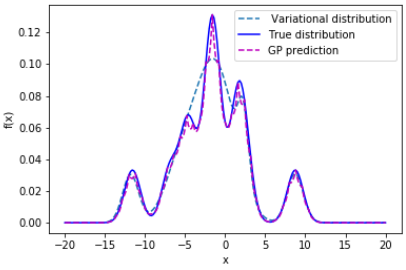
Next, it was tested the difference that the training routine makes on both accuracy and algorithm running time. For this, using the same setting as above with kernel $k_{\text{PMat},5/2}$, three training routines were tested:

- Running the algorithm for 50 iterations, with joint parameter updating done every step (training routine A).
- Running the boosting algorithm for 50 iterations, with joint parameter updating done every 10 steps (training routine B).
- Running the boosting algorithm for 50 iterations, with no joint parameter updating (training routine C).

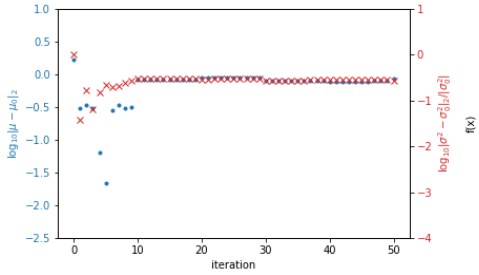
The results are shown in Figure 7.3. It can be seen that training routine B performs considerably better than routine A and routine C, while training routine A and training routine B has comparable final performance, with routine A converging in fewer iterations. As of running time, it can be see that for training routine A it increases quadratically in relation to the number of iterations, while for training



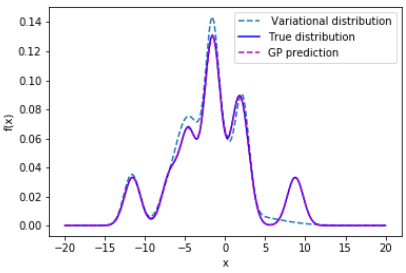
(a) $k_{\mathrm{PMat},1/2}$, moments.



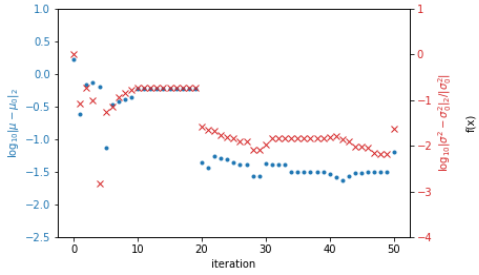
(b) $k_{\mathrm{PMat},1/2}$, final result.



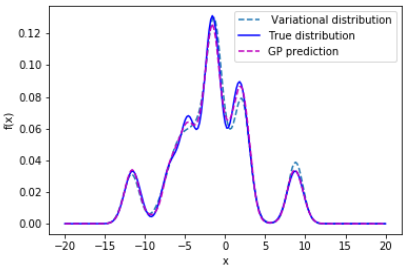
(c) $k_{\mathrm{PMat},3/2}$, moments.



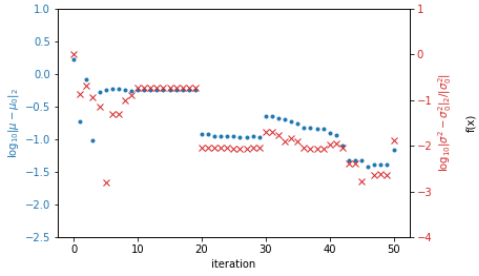
(d) $k_{\mathrm{PMat},3/2}$, final result.



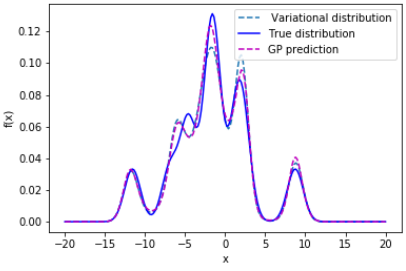
(e) $k_{\mathrm{PMat},5/2}$, moments.



(f) $k_{\mathrm{PMat},5/2}$, final result.



(g) k_{SQE}, moments.



(h) k_{SQE}, final result.

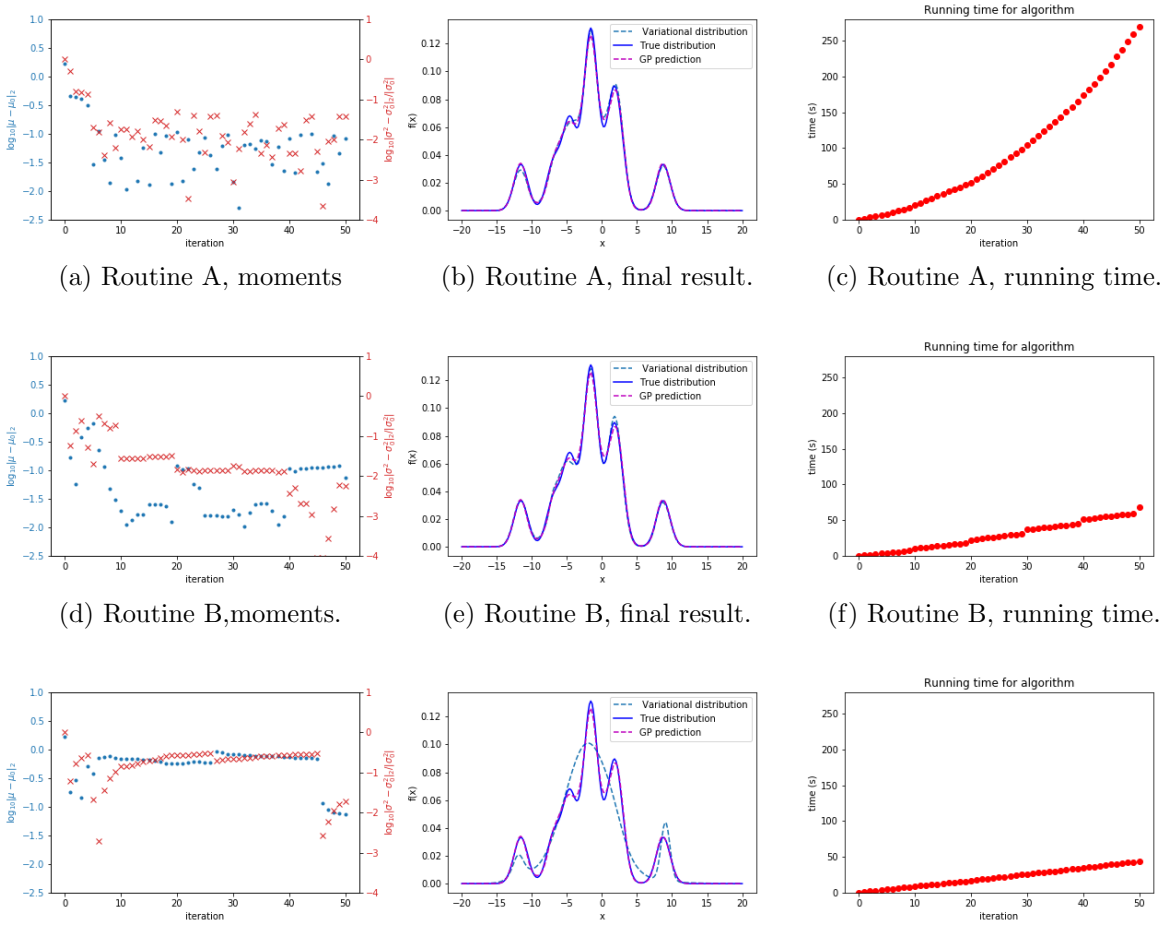
Figure 7.2: Accuracy analysis for different kernels. Each row corresponding to one kernel, in order being $k_{\mathrm{PMat},1/2}$, $k_{\mathrm{PMat},2/2}$, $k_{\mathrm{PMat},5/2}$ and k_{SQE}. The first column (MC) shows the accuracy of mean (blue), and variance (red), while the second column shows the the predicted density, the true density and the GP approximation of the density.

routine C it increases linearly. As for training routine B, the running time increases linearly except to jumps corresponding to joint parameter updating, with each jump size growing with the iteration.

It is interesting to pause and consider the behavior of training routine B, since in the next set of examples this were the routine used, with some differences of number of iterations and intervals between joint parameter updating. Notice that boosting improves considerably accuracy up to the first joint parameter updating, in 10 iterations, and after this many intervals between two joint parameter updating essentially shows no improvements. This suggests that a smarter training routine, involving boosting only in the initial steps may be desirable. This idea was not explored further in this dissertation.

7.1.2 Active evaluation

In this example, it is considered how BVBMC performs with active evaluation. For all examples, 5 initial evaluation points are sampled randomly, with distribution $\mathcal{N}(0, \sqrt{10})$. Subsequently, at each iteration an evaluation point is chosen according to the running acquisition function. Four acquisition functions were tested: uncertainty sampling (US, (5.55)), prospective prediction (PROP, (5.56)), moment-matched log transform (MMLT, (6.14)), and prospective moment-matched log transform (MMLT$_P$, (6.15)). The results are shown in Figure 7.4. There it can be seen that all acquisition function performed well, although the GP approximation is better for the PROP and MMLT acquisitions.

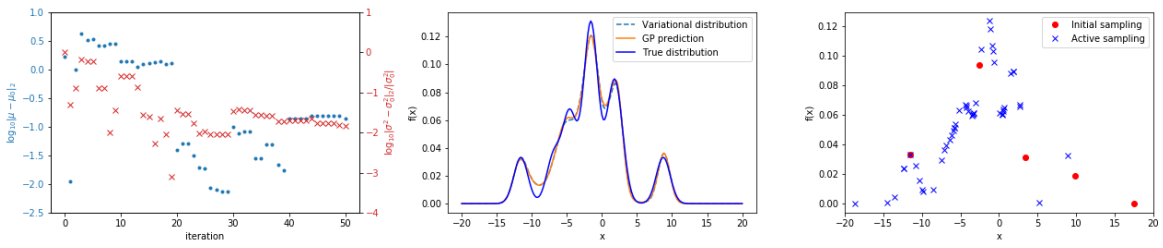


(a) Routine A, moments (b) Routine A, final result. (c) Routine A, running time.

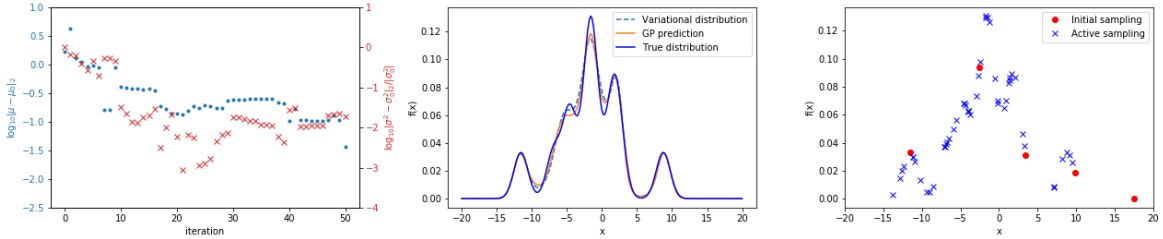
(d) Routine B,moments. (e) Routine B, final result. (f) Routine B, running time.

(g) Routine C, moments. (h) Routine C, final result. (i) Routine C, running time.

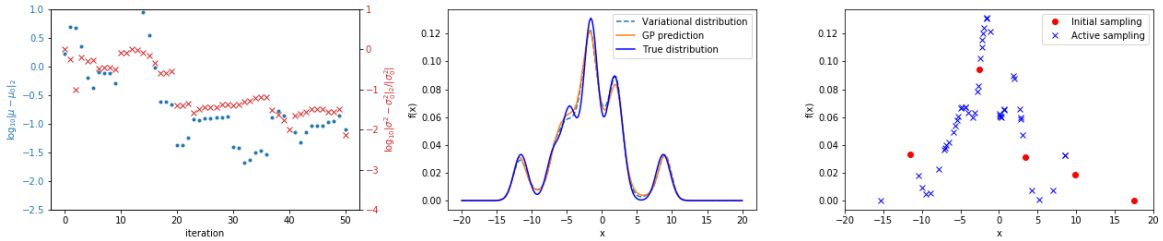
Figure 7.3: Accuracy analysis for different training routines. Each row corresponding to training routine A, B and C, respectively. The first column shows the accuracy of mean (blue) and (red), the second column show the difference between the predicted density, the true density and the GP approximation of the density, and the third column shows the algorithm running time at each step.



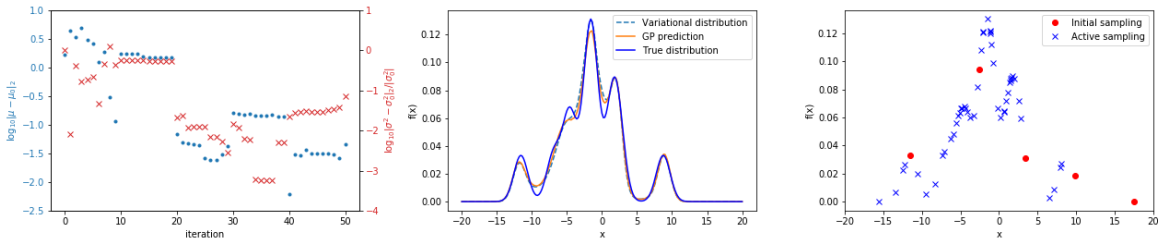
(a) US, moments. (b) US, final result. (c) US, sampling.



(d) PROP, moments. (e) PROP, final result. (f) PROP, sampling.



(g) MMLT, moments. (h) MMLT, final result. (i) MMLT, sampling.



(j) MMLT$_P$, moments. (k) MMLT$_P$, final result. (l) MMLT$_P$, sampling.

Figure 7.4: Convergence analysis for different acquisition function. Each row corresponds to acquisitions US, PROP, MMLT and MMLT$_P$, respectively. The first column shows the accuracy of mean and covariance, the second column show the difference between the predicted density, the true density and the GP approximation of the density, and the third column shows the places where the function was evaluated.

7.2 N-d toy examples

In this section, we consider the algorithm performance on a set of toy examples, the same ones considered in [2]. Code for this section can be found in `https://github.com/DFNaiff/Dissertation/tree/master/tests_dissertation/toy`.

Three classes of test cases were considered:

- *Lumpy*, a mixture of multivariate Gaussians

$$f(x) = \sum_{i=1}^{12} w_i \mathcal{N}(x; \mu_i, \Sigma_i), \tag{7.1}$$

with $(w_1, \dots, w_{12}) \sim \text{Dir}(1, \dots, 1)$, $\mu_i \sim \text{Unif}([0,1]^D)$ and $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$, with $\sigma_i^2 \sim \text{Unif}(0.2, 0.6)$. This distribution tests the algorithm performance in presence of possible multimodalities.

- *Cigar*, a anisotropic Gaussian distribution

$$f(x) = \mathcal{N}(x; 0, \Sigma), \tag{7.2}$$

where $\Sigma = Q\Lambda Q^T$, with $\Lambda = (10.0, 0.1, \dots, 0.1)$, and Q sampled from the uniform measure in the special orthogonal group. This distribution tests the algorithm performance in presence of large anisotropy.

- *Student-t*, a product of t distributions

$$f(x) = \prod_{d=1}^{D} \mathcal{T}(x_i; \nu_i), \tag{7.3}$$

with $\nu_i \sim \text{Unif}(2.5, 2 + 0.5D)$. This distribution tests the algorithm performance in presence of heavy tails [2] [3].

[2] For both *Cigar* and *Student-t*, BVBMC was applied to an unnormalized density.

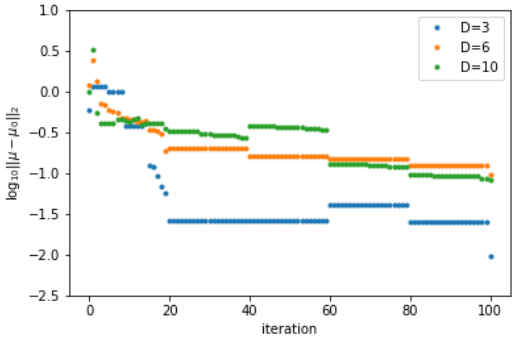
[3] Originally, $\nu_i = 2.5$ for every i, which assured reasonably heavy tails at every dimension. However, for comparison with [2], this later case is not shown here

For each case, dimensions $D = 2, 6, 10$ where tested, and the BVBMC algorithm was run for 100 iterations, with $10D$ initial samples. The GP kernel used were $k_{\text{PMat},\nu=2.5}$, with active evaluation at each iteration, according to an acquisition function randomly chosen between the pair $(\alpha_{\text{PROP}}, \alpha_{\text{MMLT}})$. Every 20 steps, joint parameter updating was done, and pruning was done at each iteration, with $\beta = 10^{-3}$.

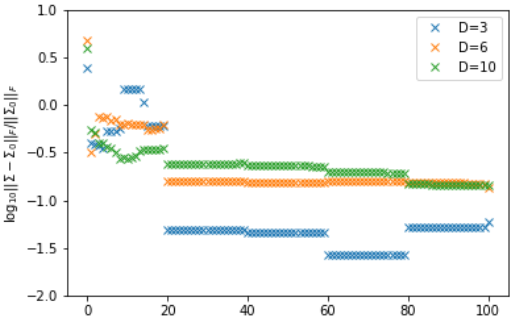
The algorithm performance was compared by checking the divergence between the true mean μ_0 and the estimated mean μ, in $\log_{10} ||\mu - \mu_0||_2$, and between the true covariance Σ_0 and estimated covariance Σ, in $\log_{10} ||\Sigma - \Sigma_0||_F / ||\Sigma_0||_F$, where $||\cdot||_F$ denotes the Frobenius norm. The results are shown in Figure 7.5. For comparison with results given by the VBMC algorithm, shown in [2], the "Gaussianized" symmetric KL divergence (gsKL) between the true distribution and estimated distribution is computed, which is defined as the symmetric KL divergence between two multivariate distributions with mean and covariance equals to the true distribution and the estimated distribution, respectively [4]. The results are shown in Table 7.1. There it can be seen that, in general performance between BVBMC and VBMC was comparable, excluding the experiment with the *Cigar* distribution with $D = 10$, in which VBMC performed much better, and $D = 2$, in which BVBMC performed better. [5]

[4] In [2], what is done are 15 evaluations for each case, and the median is taken. A better approach here would be doing the same thing for BVBMC, however due to time constraints this couldn't be done.

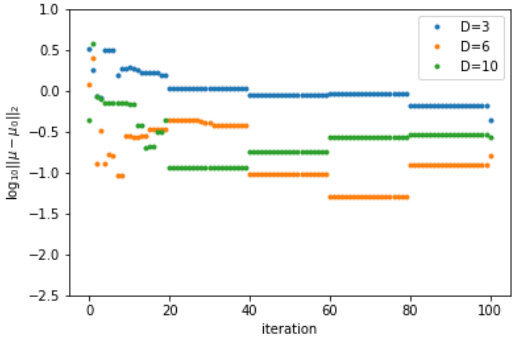
[5] It should be noted that the BVBMC algorithm have used fewer evaluations, namely 120,160 and 200 for $D = 2, 6, 10$, respectively, than the VBMC algorithm, which used 200,400,600, for $D = 2, 6, 10$. In this toy example this doesn't make a large computational difference, however in applications where likelihood evaluation is expensive it does.



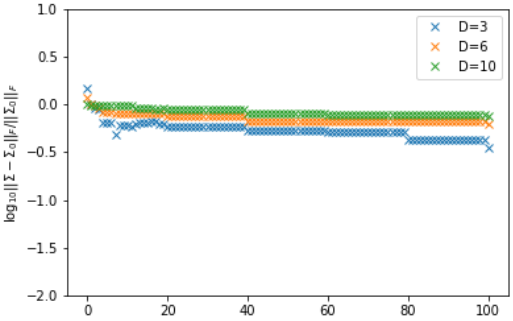
(a) Lumpy, means accuracy.



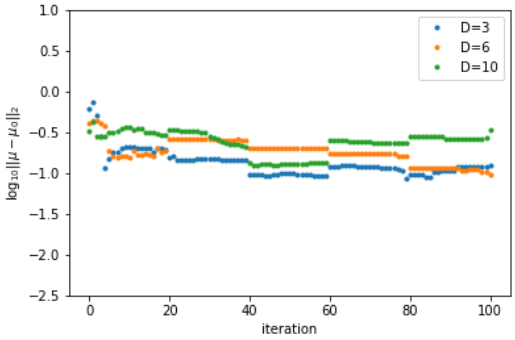
(b) Lumpy, covariances accuracy.



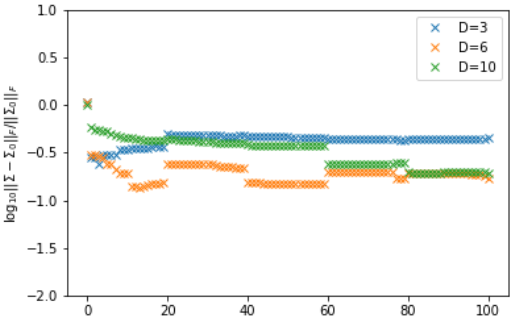
(c) Cigar, means accuracy.



(d) Cigar, covariances accuracy.



(e) Student-t, means accuracy.



(f) Student-t, covariances accuracy.

Figure 7.5: Accuracy analysis for different N-d examples. Each row corresponding to *Lumpy*, *Cigar* and *Student-t*, respectively. The first column shows the accuracy of means, while the second column shows the accuracy of covariances.

	Lumpy		Cigar		Student-t	
	BVBMC	VBMC	BVBMC	VBMC	BVBMC	VBMC
D=2	3.12×10^{-3}	6.5×10^{-4}	8.12×10^{-3}	2.1×10^{-1}	2.9×10^{-1}	2.0×10^{-3}
D=6	6.59×10^{-2}	3.5×10^{-2}	5.56×10^{-1}	1.07×10^{-1}	1.14×10^{-1}	2.3×10^{-1}
D=10	1.19×10^{-1}	4.2×10^{-1}	1.29	1.0×10^{-1}	2.56×10^{-1}	2.7×10^{-1}

Table 7.1: gsKL divergence between true distribution and estimated distribution. The values for VBMC were taken from the graphs in [2].

7.3 Contamination source estimation

In this example, a contamination source location problem was considered, inspired by a problem in [10]. This problem is a toy example of an actual inverse problem, that is, given some sensor measurements of a contaminated field (the contamination may be, for instance, radiation), find where the contaminant as located, as well as its nature. Code for this section can be found in `https://github.com/DFNaiff/Dissertation/tree/master/tests_dissertation/source1d`.

The example consider a one-dimensional domain $B = [0, 1]$, in which a contamination source $q(x, t)$ is inserted from $t = 0$ to $t = t_s$. This source is modeled as

$$q(x,t) = q_0 \exp\left(-\frac{(x-x_0)^2}{2\rho^2}\right) \mathbf{1}_{[0,t_s)}(t). \tag{7.4}$$

The contaminant itself is assumed to follow the diffusion equation

$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial^2}{\partial x^2}u(x,t) + q(x,t), \quad x \in \text{int}B. \tag{7.5}$$

Moreover, the initial contamination is considered to be 0, while the walls are considered insulated, resulting in the boundary and initial value conditions

$$u(x,0) = 0, \ \frac{\partial}{\partial x}u(0,t) = \frac{\partial}{\partial x}u(1,t) = 0. \tag{7.6}$$

It must be noticed that, for general domain length L and diffusion coefficient k, using adimensionalization one can reduce the general problem to the above.

In this setting, at each wall $x = 0, 1$, four measurements of u are made, for $t_m \in T_m = \{0.075, 0.15, 0.225, 0.3, 0.4\}$, resulting in the data

$$\mathcal{D} = \{\hat{u}(x_m, t_m)\}_{x_m \in \{0,1\}, t_m \in T_m}.$$

Moreover, the measurements are assumed to be noisy, with $\hat{u}(x_m, t_m) = u(x_m, t_m) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The noise parameter σ^2 is assumed to follow a prior InvGamma(α, β). This allows us to marginalize out σ^2, letting $\hat{u}(x_m, t_m)$ be distributed according to the generalized t-distribution [6] with 2α degrees of freedom $\mathcal{T}(u(x_m, t_m), \beta/\alpha, 2\alpha)$.

The setting above results in a 4-dimensional inference problem, for the variables (x_0, t_s, q_0, ρ), with likelihood

$$p(\mathcal{D}|x_0, t_s, q_0, \rho) = \prod_{x_m \in \{0,1\}, t_m \in T_M} \mathcal{T}(\hat{u}(x_m, t_m); u(x_m, t_m), \beta/\alpha, 2\alpha). \tag{7.7}$$

Given priors for x_0,t_s,q_0 and ρ, the associated posterior distribution for the parameters becomes

$$p(x_0, q_0, t_s, \rho|\mathcal{D}) \propto p(\mathcal{D}|x_0, q_0, T_s, \rho)p(x_0)p(q_0)p(\rho)p(t_s). \tag{7.8}$$

A synthetic data $\mathcal{D}$ was generated, with the parameters given in Table 7.2, first row. The measurement noise was $\sigma^2 = 10^{-2}$, and the equation was simulated using a finite differences routine. The priors for the values to be inferred were set as [7]

$$\begin{aligned} p(x_0) &= \text{Unif}(x_0; 0, 1) \\ p(t_s) &= \text{Unif}(t_s; 0, 0.4) \\ p(q_0) &= \text{HalfCauchy}(q_0; 10) \\ p(\rho) &= \text{HalfCauchy}(\rho; 0.1) \end{aligned} \tag{7.9}$$

In (7.8), $u(x, t)$ is also calculated by finite differences.

[6] If T_0 follows a (standardized) t-distribution, with nu degrees of freedom, then $T = \sigma T_0 + \mu$ follows a generalized t-distribution denoted by $\mathcal{T}(\mu, \sigma^2, \nu)$

[7] The Half-Cauchy distribution was used here as to represent a non-informative prior.

	x_0	t_s	q_0	ρ
True	0.230	0.300	6.366	0.050
BVBMC mean	0.328	0.213	5.435	0.140
EMCEE mean	0.352	0.206	10.228	0.218
BVBMC HDP 70%	$(1.1 \cdot 10^{-4}, 0.43)$	$(0.12, 0.36)$	$(1.0, 6.7)$	$(2.7 \cdot 10^{-3}, 0.1)$
EMCEE HDP 70%	$(3.4 \cdot 10^{-4}, 0.45)$	$(0.08, 0.33)$	$(0.4, 10.3)$	$(2.1 \cdot 10^{-3}, 0.1)$

Table 7.2: Comparison of the true parameter of the problem (first row), the estimated means using BVBMC (second row) and EMCEE (third row), and 70% highest posterior density interval for BVBMC and EMCEE.

The BVBMC algorithm assumes that the distribution to be estimated has support in $\mathbb{R}^D$. This is not the case for the problem above, since x_0 and t_s have both bounded support, while q_0 and ρ have positive support. In order to apply the BVBMC algorithm, inference was made on the warped variables $\tilde{x}_0, \tilde{t}_s, \tilde{q}_0, \tilde{\rho}$, all of them with support in $\mathbb{R}$, such that [8]:

$$\begin{aligned} x_0 &= \text{sigmoid}(\tilde{x}_0) \\ t_s &= 0.4 \times \text{sigmoid}(\tilde{t}_s) \\ q_0 &= \exp(\tilde{q}_0) \\ \rho &= \exp(\tilde{\rho}), \end{aligned} \tag{7.10}$$

with

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{7.11}$$

This results in the posterior distribution for the warped variables

$$p(\tilde{x}_0, \tilde{t}_s, \tilde{q}_0, \tilde{\rho}|\mathcal{D}) \propto p(x_0, q_0, t_s, \rho|\mathcal{D})\text{sigmoid}'(\tilde{x}_0)\text{sigmoid}'(\tilde{t}_s)\exp(\tilde{q}_0)\exp(\tilde{\rho}) \tag{7.12}$$

The BVBMC algorithm was applied to the problem, with the following setup:

- The kernel used was $k_{\text{PMat},3/2}$ (both $k_{\text{PMat},5/2}$ and k_{SQE} were also tested, with mixed results).

[8] The 0.4 factor here is due to the fact that t_s lies between 0 and 0.4.

- The algorithm was initialized with 40 samples from the prior. After this, before the training loop, 40 more evaluations points were chosen by using the α_{MMLT} acquisition function.

- The algorithm was then run for 100 iterations, with an evaluation point chosen at each iteration, with the acquisition function chosen randomly between α_{MMLT} and α_{PROP}. At each 20 iterations, the parameters were jointly optimized.

The predicted mean is shown in Table 7.2, second row. It can be seen that the estimated source location was relatively accurate, compared to the original one, while estimates for ρ and q_0 where reasonable, and the estimate for t_s did not deviate far from the prior mean. The resulting marginal univariate and bivariate distributions are shown in Figure 7.6.

For comparison, the EMCEE algorithm [31], which is a MCMC algorithm usually used for problems in astrophysics, was also tested. The EMCEE ran with 10 walkers and 10000 steps for each walker, using burn-in in the first 1000 steps. The resulting estimated means is shown in Table 7.2, third row. It can be seem that in general the estimates were in the BVBMC algorithm, except for q_0, in which BVBMC seems to be more precise. The resulting marginal univariate and bivariate distributions are shown in Figure 7.6, showing some resemblance with the results found in BVBMC.

7.4 Checking performance

In practice, one does not know the true posterior for doing comparison, and there must be some way to check whether BVBMC arrived at a good posterior.

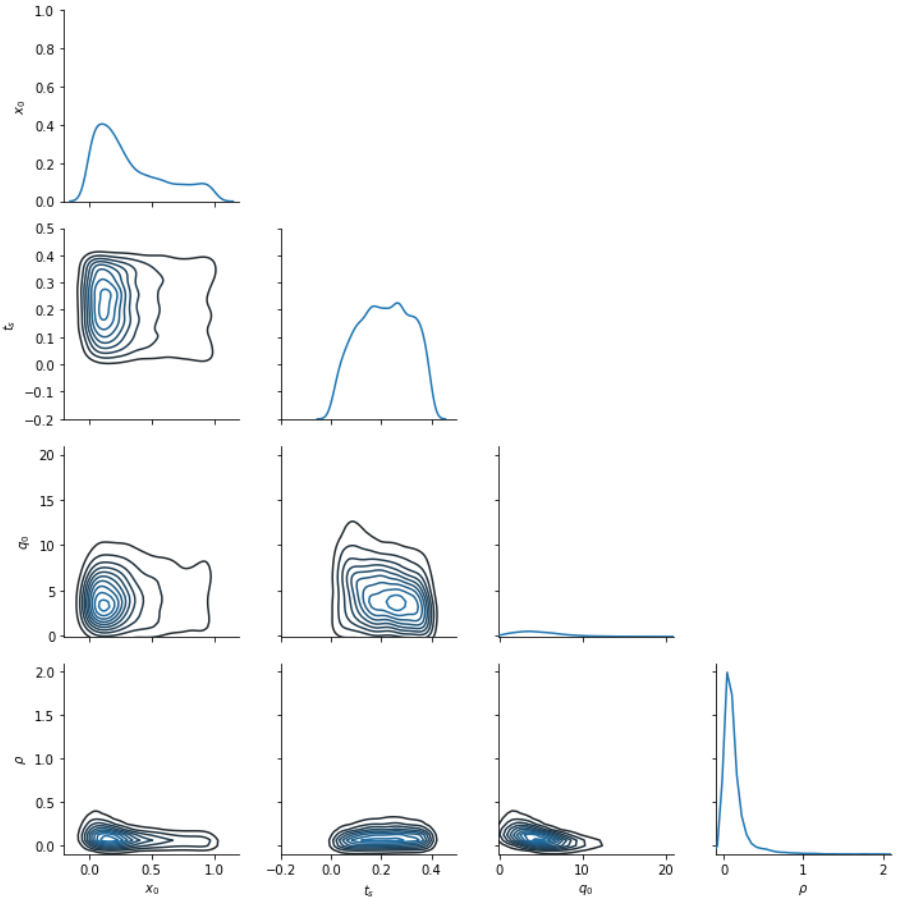


Figure 7.6: KDE plots of estimated marginals with BVBMC. On diagonal, marginal univariate distributions for x_0,t_s,q_0 and ρ are shown, while off-diagonal, the corresponding bivariate marginals for each pair is shown.

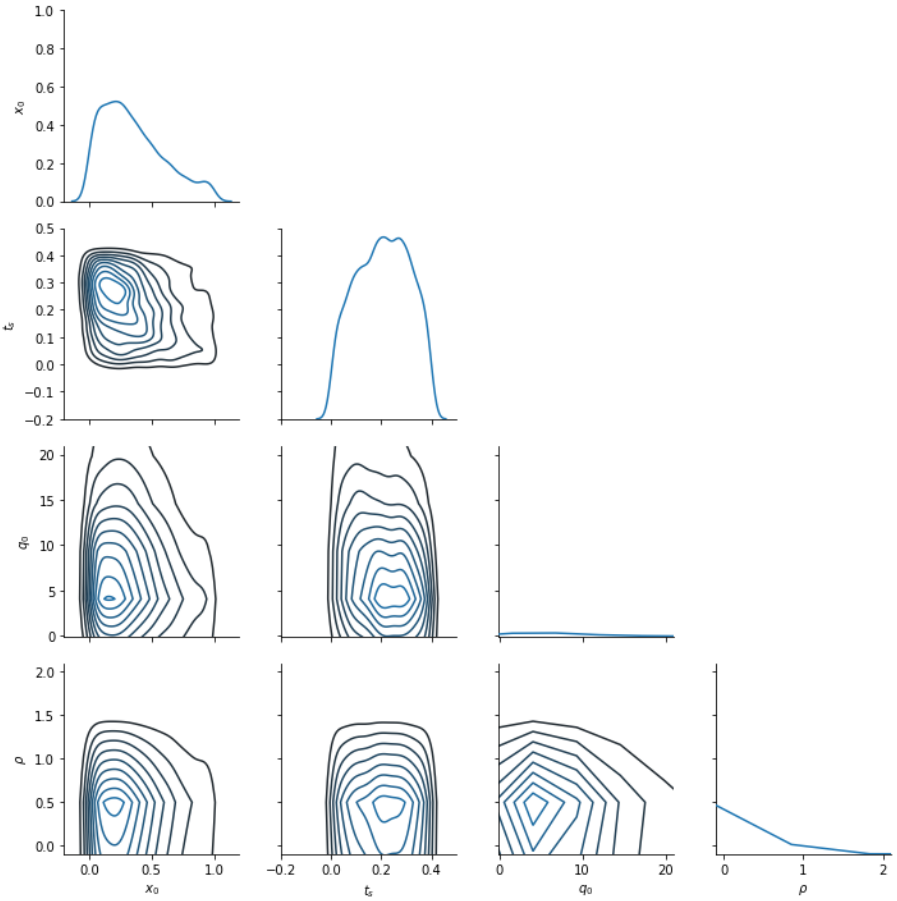


Figure 7.7: KDE plots of estimated marginals with EMCEE. On diagonal, marginal univariate distributions for x_0,t_s,q_0 and ρ are shown, while off-diagonal, the corresponding bivariate marginals for each pair is shown.

Two sources of error may arise in the BVBMC estimate and the true posterior: Whether the variational proposal does not approximate well the GP surrogate model, or the GP surrogate model does not approximate well the true unnormalized posterior. The first case may be checked using some rough estimate of the KL divergence between the variational proposal and the surrogate model, and is implemented in the method *kl_vb_bmc* of the associated package.

Checking whether the GP surrogate model resembles the true model may be harder. One option is to use leave-one-out testing [87], but one must consider that one does not care about accuracy in places where the unnormalized posterior does not contribute in probability mass. A heuristic to address this problem was not developed by the author.

8 FUTURE CHALLENGES AND CONCLUSION

Here, we present some possible directions to extend the presented method to other applications.

8.1 Reparameterization trick with Gaussian Processes

The Bayesian Monte Carlo approach for approximating the integral terms

$$\int \mathbb{E}[\log(\bar{g}_{\mathcal{D}}(\theta))]q_i(\theta)$$

suffers from the fact that the GP kernel and the distribution q_i are limited, since

$$\int k(\theta, \theta_i)q_i(\theta)d\theta$$

must be tractable. One manner to circumvent this is by abandoning the BMC approach to integration, and insteading using the reparameterization trick presented in Section 5.3.2, turning the BVBMC approach closer in spirit to the one in 5.4.2.

One disadvantage is that evaluations of Gaussian Process, although cheap, are not extremely cheap, specially for large datasets, so reparameterization may be considerably slower.

8.2 Extending BVBMC to pseudo-marginal likelihoods

Consider that, as in Section 2.5.1, that $\bar{g}(\theta) = Zp(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$ is truly unavailable, and even the pseudo-marginals $\hat{\bar{g}}(\theta) = Z\hat{p}(\theta|\mathcal{D})$ are expensive to

calculate.

Gaussian processes accommodates, for evaluation points $\{\theta_i\}_{i=1}^N$, the noisy estimates $\{\hat{\bar{g}}(\theta_i)\}_{i=1}^N$ of $\{\bar{g}(\theta_i)\}_{i=1}^N$. If one were doing GP regression on $\bar{g}(\theta)$, one could assume that $p(\hat{\bar{g}}|\bar{g})$ is roughly Gaussian, due to the central limit theorem, and use (3.7) as surrogate model.

However, in BVBMC (and VBMC), one uses the GP surrogate model on $\log \bar{g}(\theta)$. This implies that, letting $\epsilon = \hat{\bar{g}}(\theta) - \bar{g}(\theta)$ be the noise random variable, one have the model for $\log \bar{\bar{g}}(\theta)$

$$\log \bar{\bar{g}}(\theta) = \log \left(e^{\log \bar{g}(\theta)} + \epsilon \right), \tag{8.1}$$

which is a complicated noise model, to be treated as in (3.8). Furthermore, one cannot even assume this noise term to be controlled, since, by doing a rough Taylor expansion:

$$\log \bar{\bar{g}}(\theta) = \log \bar{g}(\theta) + e^{-\log \bar{g}(\theta)} \epsilon \tag{8.2}$$

which results in a very large noise for low values of $-\log \bar{g}(\theta)$. One future work could be on how to address this problem.

8.3 Scaling BVBMC to a larger number of evaluations

Given the scaling problems of GP discussed in Section 3.5.2, for unnormalized posteriors $\bar{g}(\theta)$ that can be evaluated in tens of thousands, but that evaluations in hundred of thousands or millions is hard, naive use of BVBMC runs into problems.

A possibility is two use sparse Gaussian Processes, that are briefly reviewed in the Appendix A. However, their integration with BVBMC ran into problems, so further research would be needed.

Of course, one could drop the use of GPs and use other surrogate function methods as done in [15, 68]. However, it should be noted that local approximation methods may not work with variational inference, because of its global approximation nature.

8.4 Conclusion

The method presented in this work, although still immature, has shown promise for use in Bayesian inference, where the likelihood function is expensive of evaluate, that are common in inverse problems.

The associated package in `https://github.com/DFNaiff/BVBMC`, built on top of PyTorch, is intended to be easy to use, so a practitioner can quickly employ it in their own problems, if they wish so.

REFERENCES

[1] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162:2025–35, 01 2003.

[2] Luigi Acerbi. Variational bayesian monte carlo. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8213–8223. Curran Associates, Inc., 2018.

[3] Robert J. Adler. *The geometry of random fields*. John Wiley and Sons, 1981.

[4] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, Jun 2010.

[5] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient monte carlo computations. *Annals of Statistics*, pages 697–725, 2009.

[6] Oleg Arenz, Gerhard Neumann, and Mingjun Zhong. Efficient gradient-free variational inference using policy search. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 234–243, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[7] M. J. Asher, B. F. W. Croke, A. J. Jakeman, and L. J. M. Peeters. A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8):5957–5973, Aug 2015.

[8] Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of bayes estimators. *Mathematical Programming*, 174(1-2):129–144, Jan 2018.

[9] Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv e-prints*, page arXiv:1701.02434, Jan 2017.

[10] I Bilionis and N Zabaras. Solution of inverse problems with limited forward solver evaluations: a bayesian perspective. *Inverse Problems*, 30(1):015004, dec 2013.

[11] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *arXiv e-prints*, page arXiv:1810.09538, Oct 2018.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[13] Christopher M. Bishop, Neil D. Lawrence, Tommi Jaakkola, and Michael I. Jordan. Approximating posterior distributions in belief networks using mixtures. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 416–422. MIT Press, 1998.

[14] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Feb 2017.

[15] Nikolay Bliznyuk, David Ruppert, and Christine A. Shoemaker. Local derivative-free approximation of computationally expensive posterior densities. *Journal of Computational and Graphical Statistics*, 21(2):476–495, Apr 2012.

[16] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13, Feb 1999.

[17] Stephen Boyd and Lieven Vandenberghe. Convex optimization. 2004.

[18] François-Xavier Briol, Chris J. Oates, Mark Girolami, and Michael A. Osborne. Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1162–1170, Cambridge, MA, USA, 2015. MIT Press.

[19] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

[20] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[21] Thang D. Bui, Josiah Yan, and Richard E. Turner. A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation. *arXiv e-prints*, page arXiv:1605.07066, May 2016.

[22] Henry R. Chai and Roman Garnett. Improving quadrature for constrained integrands. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2751–2759. PMLR, 16–18 Apr 2019.

[23] Chris Chatfield. *The analysis of time series: an introduction*. CRC Press, Florida, US, 6th edition, 2004.

[24] Patrick R. Conrad, Youssef M. Marzouk, Natesh S. Pillai, and Aaron Smith. Accelerating asymptotically exact mcmc for computationally intensive models via local approximations. *Journal of the American Statistical Association*, 111(516):1591–1607, 2016.

[25] Richard T. Cox and E. T. Jaynes. The algebra of probable inference. *American Journal of Physics*, 31(1):66–67, Jan 1963.

[26] R. M. Dudley. Real analysis and probability. 2002.

[27] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, Sep 1936.

[28] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability and Its Applications*, 14(1):153–158, Jan 1969.

[29] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[30] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, May 2012.

[31] Daniel Foreman-Mackey, David W. Hogg, Dustin Lang, and Jonathan Goodman. emcee: The mcmc hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306–312, Mar 2013.

[32] C. Fowlkes, S. Belongie, Fan Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, Feb 2004.

[33] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[34] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

[35] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Statist.*, 28(2):337–407, 04 2000.

[36] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[37] Samuel Gershman, Matt Hoffman, and David Blei. Nonparametric variational inference. *arXiv e-prints*, page arXiv:1206.4665, Jun 2012.

[38] C. J. Geyer. Introduction to markov chain monte carlo. In G. Jones A. Gelman, S. Brooks and X. L. Meng, editors, *Handbook of Markov Chain Monte Carlo: Methods and Applications*. CRC Press, London, 2004.

[39] Zoubin Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553, Feb 2013.

[40] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015.

[41] Zoubin Ghahramani and Carl E. Rasmussen. Bayesian monte carlo. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

[42] Alex Gittens. The spectral norm error of the naive Nystrom extension. *arXiv e-prints*, page arXiv:1110.5305, Oct 2011.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[44] T. Gunter, M. Osborne, R. Garnett, P. Hennig, and S. Roberts. Sampling for inference in probabilistic models with fast bayesian quadrature. In *Advances in Neural Information Processing Systems 27*, pages 2789–2797. Curran Associates, Inc., 2014.

[45] Fangjian Guo, Xiangyu Wang, Kai Fan, Tamara Broderick, and David B. Dunson. Boosting Variational Inference. *arXiv e-prints*, page arXiv:1611.05559, Nov 2016.

[46] P. Hennig and M. Kiefel. Quasi-newton methods – a new direction. In *Int. Conf. on Machine Learning (ICML)*, volume 29, 2012.

[47] James Hensman, Magnus Rattray, and Neil D. Lawrence. Fast variational inference in the conjugate exponential family. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2888–2896, USA, 2012. Curran Associates Inc.

[48] José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland , Daniel Hernández-Lobato, Thang Bui, and Richard E. Turner. Black-box α-divergence Minimization. *arXiv e-prints*, page arXiv:1511.03243, Nov 2015.

[49] N. Hjort, C. Holmes, P. Mueller, and S. Walker. *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, Cambridge, UK, 2010.

[50] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[51] Leslie Hogben. *Handbook of Linear Algebra*. 2nd edition, 2014.

[52] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck. On entropy approximation for gaussian mixture random vectors. *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Aug 2008.

[53] Tommi S. Jaakkola and Michael I. Jordan. *Improving the Mean Field Approximation Via the Use of Mixture Distributions*, pages 163–173. Springer Netherlands, Dordrecht, 1998.

[54] Martin Jankowiak and Theofanis Karaletsos. Pathwise derivatives for multivariate distributions. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 333–342. PMLR, 16–18 Apr 2019.

[55] E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003.

[56] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, Dec 2001.

[57] Motonobu Kanagawa and Philipp Hennig. Convergence Guarantees for Adaptive Bayesian Quadrature Methods. *arXiv e-prints*, page arXiv:1905.10271, May 2019.

[58] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *arXiv e-prints*, page arXiv:1807.02582, Jul 2018.

[59] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. Bayesian active learning for posterior estimation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3605–3611. AAAI Press, 2015.

[60] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

[61] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, Dec 2013.

[62] Yingzhen Li and Richard E. Turner. R\'enyi Divergence Variational Inference. *arXiv e-prints*, page arXiv:1602.02311, Feb 2016.

[63] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *arXiv e-prints*, page arXiv:1807.01065, Jul 2018.

[64] Qing Liu and Donald A. Pierce. A note on gauss-hermite quadrature. *Biometrika*, 81(3):624–629, 1994.

[65] Francesco Locatello, Gideon Dresdner, Rajiv Khanna, Isabel Valera, and Gunnar Raetsch. Boosting black box variational inference. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3401–3411. Curran Associates, Inc., 2018.

[66] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[67] David J.C. MacKay. Bayesian interpolation. *NEURAL COMPUTATION*, 4:415–447, 1991.

[68] Youssef M. Marzouk, Habib N. Najm, and Larry A. Rahn. Stochastic spectral methods for efficient bayesian solution of inverse problems. *Journal of Computational Physics*, 224(2):560–586, Jun 2007.

[69] N. David Mermin. Could feynman have said this? *Physics Today*, 57(5):10–11, May 2004.

[70] Andrew C. Miller, Nicholas Foti, and Ryan P. Adams. Variational Boosting: Iteratively Refining Posterior Approximations. *arXiv e-prints*, page arXiv:1611.06585, Nov 2016.

[71] Rekar O. Mohammed and Gavin C. Cawley. Over-fitting in model selection with gaussian process regression. In *Machine Learning and Data Mining in Pattern Recognition*, pages 192–205, Cham, 2017. Springer International Publishing.

[72] K. Murphy. *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012.

[73] Iain Murray. Differentiation of the Cholesky decomposition. *arXiv e-prints*, page arXiv:1602.07527, Feb 2016.

[74] Radford M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. *arXiv e-prints*, page physics/9701026, Jan 1997.

[75] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl E. Rasmussen. Active learning of model evidence using bayesian quadrature. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 46–54. Curran Associates, Inc., 2012.

[76] Michael Osborne and Stephen Roberts. Gaussian processes for prediction. 01 2007.

[77] A. O'Hagan. Bayes–hermite quadrature. *Journal of Statistical Planning and Inference*, 29(3):245–260, Nov 1991.

[78] Anthony O'Hagan. Some bayesian numerical analysis. *Bayesian Statistics*, 4(345–363):4–2, 1992.

[79] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*, 2017.

[80] Dejan Petelin, Matej Gasperin, and Václav Smídl. Adaptive importance sampling for bayesian inference in gaussian process models. *IFAC Proceedings Volumes*, 47(3):5011–5016, 2014.

[81] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. Version 20121115.

[82] José C. Pinheiro and Douglas M. Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and Computing*, 6:289–296, 1996.

[83] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, Dec 1999.

[84] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, Jan 1999.

[85] J. Quinonero Candela and CE. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1935–1959, December 2005.

[86] Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland, 22–25 Apr 2014. PMLR.

[87] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[88] Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 294–300. MIT Press, 2001.

[89] CE. Rasmussen. Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. pages 651–659. Max-Planck-Gesellschaft, 2003.

[90] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[91] Christian P. Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer-Verlag Inc, Berlin; New York, 2001.

[92] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[93] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv e-prints*, page arXiv:1609.04747, Sep 2016.

[94] Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. Overdispersed Black-Box Variational Inference. *arXiv e-prints*, page arXiv:1603.01140, Mar 2016.

[95] Tim Salimans and David Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8, 06 2012.

[96] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, 2:e55, apr 2016.

[97] Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *IN WORKSHOP ON AI AND STATISTICS 9*, 2003.

[98] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan 2016.

[99] John Shawe-Taylor and Nello Cristianini. Kernel methods for pattern analysis. 2004.

[100] Chris Sherlock, Alexandre H. Thiery, Gareth O. Roberts, and Jeffrey S. Rosenthal. On the efficiency of pseudo-marginal random walk metropolis algorithms. *Ann. Statist.*, 43(1):238–275, 02 2015.

[101] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):1–21, Sep 1985.

[102] S. P. Smith. Differentiation of the cholesky algorithm. *Journal of Computational and Graphical Statistics*, 4(2):134–147, 1995.

[103] Alex J. Smola and Peter Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

[104] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT press, 2006.

[105] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA, 2012. Curran Associates Inc.

[106] Michael L. Stein. Interpolation of spatial data. *Springer Series in Statistics*, 1999.

[107] A. M. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559, May 2010.

[108] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[109] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[110] Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *In Artificial Intelligence and Statistics 12*, pages 567–574, 2009.

[111] Michalis Titsias RC AUEB and Miguel Lázaro-Gredilla. Local expectation gradients for black box variational inference. In *Advances in Neural Information Processing Systems 28*, pages 2638–2646. Curran Associates, Inc., 2015.

[112] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv e-prints*, page arXiv:1610.09787, Oct 2016.

[113] Dilin Wang, Hao Liu, and Qiang Liu. Variational inference with tail-adaptive f-divergence. In *Advances in Neural Information Processing Systems 31*, pages 5737–5747. Curran Associates, Inc., 2018.

[114] Hongqiao Wang and Jinglai Li. Adaptive gaussian process approximation for bayesian inference with expensive likelihood functions. *Neural Computation*, 30(11):3072–3094, Nov 2018.

[115] Jiaping Wang, Yue Dong, Xin Tong, Zhouchen Lin, and Baining Guo. Kernel nyström method for light transport. *ACM Transactions on Graphics*, 28(3):1, Jul 2009.

[116] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[117] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, pages 1067–1075, 2013.

[118] David Wingate and Theophane Weber. Automated Variational Inference in Probabilistic Programming. *arXiv e-prints*, page arXiv:1301.1299, Jan 2013.

[119] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, Aug 2019.

[120] Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, Mar 2003.

APPENDIX A SPARSE GAUSSIAN PROCESSES

Given the scaling problem of GP methods, many sparsifications proposals were proposed, and the subject is a very fruitful area of research. We present below some of those approaches, although there are many other (for a recent review, see [63]). In the following, $\sigma_n > 0$, for reasons that will be clear.

A.1 Nystrom extension

One approach is to simply approximate the matrix $K_\delta(\mathbf{x}, \mathbf{x}) = K(\mathbf{x}, \mathbf{x}) + \delta_n I$ in an manner that the matrix inversion (or Cholesky decomposition) becomes cheaper. One approach is to find an low-rank approximation of $K(\mathbf{x}, \mathbf{x}) = UWU^T$, where U is $n \times m$ and W $m \times m$, and $m \ll n$. Then, one can find both the inverse and Cholesky decomposition of $VWV^T + \delta_n I$ with $\mathcal{O}(m^3)$ computational cost (B.1).

The optimal m-rank approximation of $K(\mathbf{x}, \mathbf{x})$, in both spectral norm and Frobenius norm is given by

$$\tilde{K} := \sum_{i=1}^{l} \lambda_i \mathbf{v}_i \mathbf{v}_i^T = V_l \Lambda_l V_l^T,$$

where $\lambda_1 \geq \ldots \geq \lambda_m \geq \lambda_{m+1} \geq \ldots \geq \lambda_n$ are the eigenvalues of $K(\mathbf{x}, \mathbf{x})$, and $\mathbf{v}_1, \ldots \mathbf{v}_n$ are the corresponding eigenvalues ([27], coupled with the fact that for positive-semidefinite matrices the eigenvalue an singular value decomposition are the same). Then, $||K(\mathbf{x}, \mathbf{x}) - \tilde{K}||_2 = \lambda_{m+1}$. Unfortunately, this optimal low-rank approximation itself requires calculating the spectral decomposition of $K(\mathbf{x}, \mathbf{x})$, which by itself has $\mathcal{O}(N^3)$ cost. So one has instead to find some cheap method to calculate a reasonable low-rank approximation of m.

The technique of Nystrom extensions was first introduced in [116], and has found applications in kernel methods, as seen in [32], [115]. A m-rank Nystrom extension of a matrix A $n \times n$ is formed by selecting m (ordered) sub-indices from $\{1, \ldots, n\}$ (call it I), and then by letting C be the $n \times m$ matrix formed by selecting the corresponding columns of A and W be the $m \times m$ matrix formed by selecting the intersection between the corresponding columns and corresponding rows of A, that is,

$$C_{i,j} = A_{i,I_j}, \quad W_{i,j} = A_{I_i,I_j}.$$

Then the corresponding Nystrom extension $\tilde{A}$ of A is given by $A = CW^{\dagger}C^T$, where $W^{\dagger}$ is the pseudo-inverse of W. The simplest Nystrom extension technique is the naive Nystrom extension, where the m sub-indices randomly without replacement from $\{1, \ldots, m\}$. With high probability, the optimal rank k approximation of A can be obtained by choosing $m = \mathcal{O}(k \log k)$ [42]. In particular, if the spectrum of A decays quickly, the naive Nystrom extension will result in a good low-rank approximation. This is usually the case for the kernel matrix $K(\mathbf{x}, \mathbf{x})$, thus making this technique an attractive choice at first. However, when applied to GP regression, one problem is that the predicive variance is not guaranteed to be positive, thus making the use of the Nystrom extension problematic [85]. Still, many sparsification techniques such as the ones shown below, end up using a version of it, with some correction ensuring that the posterior prediction is a valid distribution.

A.2 Prior approximations

The idea of prior approximations for sparsification of Gaussian Process is an unified view given by [85] that includes previous approaches offered in [103, 97, 104] To understand it, first note that one can over (dropping M for convenience) an alternative derivation of the distribution for $p(\mathbf{f}^*|\mathbf{y})$ given in section 2.1, by

expanding:

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}, \mathbf{f}^*|\mathbf{y}) d\mathbf{f} = \frac{1}{p(\mathbf{y})} \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, \mathbf{f}^*) d\mathbf{f}. \tag{A.1}$$

With the prior $p(\mathbf{f}, \mathbf{f}^*)$ given by (3.3), assuming $m = 0$ for simplicity. Then, as shown in C, one arrives at the exact same equation in (3.7). Then, since the matrix $K_{f,f}$ comes from $p(\mathbf{f}, \mathbf{f})$, one approach to reduce the costs of its inverse is to approximate it.

To construct those approximations, first consider some new fictitious evaluation points, called *inducing points* $\mathbf{x}_u = (x_{u,1}, \ldots, x_{u,m})$, that may or may not include the training points, and the corresponding evaluations $\mathbf{u} = (f(x_{u,1}), ..., f(x_{u,m}))^T$. Now, given the inducing points, one can write

$$p(\mathbf{f}, \mathbf{f}^*) = \int p(\mathbf{f}, \mathbf{f}^*|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}, \tag{A.2}$$

where $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{0}, K_{u,u})$. Now, it is made the assumption that $\mathbf{f}$ and $\mathbf{f}^*$ are independent given $\mathbf{u}$, thus $p(\mathbf{f}, \mathbf{f}^*|\mathbf{u}) \approx p(\mathbf{f}|\mathbf{u}) p(\mathbf{f}^*|\mathbf{u})$. Without using further approximations for $p(\mathbf{f}|\mathbf{u})$ and $p(\mathbf{f}^*|\mathbf{u})$:

$$\begin{aligned} p(\mathbf{f}|\mathbf{u}) &= \mathcal{N}(\mathbf{f}|K_{f,u}K_{u,u}^{-1}\mathbf{u}, K_{f,f} - K_{f,u}K_{u,u}^{-1}K_{u,f}) \\ p(\mathbf{f}^*|\mathbf{u}) &= \mathcal{N}(\mathbf{f}^*|K_{*,u}K_{u,u}^{-1}\mathbf{u}, K_{*,*} - K_{*,u}K_{u,u}^{-1}K_{u,*}). \end{aligned} \tag{A.3}$$

Using the notation $Q_{a,b} := K_{a,u}K_{u,u}^{-1}K_{u,b}$ and $K_{(f,*),u} := K((\mathbf{x}, \mathbf{x}^*), \mathbf{u})$:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} |\mathbf{u} \sim \mathcal{N}\left(\begin{bmatrix} K_{(f,*),u}(K_{u,u} + \sigma^2 I)^{-1}\mathbf{u} \end{bmatrix}, \begin{bmatrix} K_{f,f} - Q_{f,f} & 0 \\ 0 & K_{*,*} - Q_{*,*} \end{bmatrix} \right). \tag{A.4}$$

By using the equation B.4, we then find:

$$\begin{aligned} \begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim & \mathcal{N}\left(0, \begin{bmatrix} K_{f,f} - Q_{f,f} & 0 \\ 0 & K_{*,*} - Q_{*,*} \end{bmatrix} + K_{(f,*),u}K_{u,u}^{-1}K_{(f,*),u}^T \right). \\ & \mathcal{N}\left(0, \begin{bmatrix} K_{f,f} - Q_{f,f} + Q_{f,f} & Q_{f,*} \\ Q_{*,f} & K_{*,*} - Q_{*,*} + Q_{*,*} \end{bmatrix} \right). \end{aligned} \tag{A.5}$$

By using (A.5) in (A.1), we get:

$$\mathbf{f}^*|\mathbf{y} \approx \mathcal{N}(\mathbf{f}|Q_{*,f}(K_{f,f}+\sigma^2 I)^{-1}\mathbf{y}, K_{*,*} - Q_{*,f}(K_{f,f}+\sigma^2 I)^{-1}Q_{f,*}). \tag{A.6}$$

Nothing is really gained by considering the exact posteriors $p(\mathbf{f}|\mathbf{u})$ and $p(\mathbf{f}^*|\mathbf{u})$, since still have the inverse of $K_{f,f}+\sigma^2 I$. Thus, there is need for further approximations, in turn to simplify the covariance matrix of $\mathbf{f}|\mathbf{u}$, thus simplifying $K_{f,f}+\sigma^2 I$ into something manageable for inversion. The main approximations of this kind are shown below. Notice that if the inducing points $\mathbf{x}_u$ are a subset of $\mathbf{x}$, then $Q_{f,f}$ is in fact a Nystrom extension of $K_{f,f}$.

A.2.0.1 Subset of regressors

The subset of regressors approximation for GPs was first proposed in [103], adapting an idea from [101]. It originally considered the generative model for any $\mathbf{f}^*$ (including the training values $\mathbf{f}$):

$$\mathbf{f}^* = K_{*,u}\mathbf{w}_u, \quad \mathbf{w}_u \sim \mathcal{N}(0, K_{u,u}^{-1}). \tag{A.7}$$

In particular, this implies that $\mathbf{u} = K_{u,u}\mathbf{w}_u$, hence, within the prior approximation framework, the SoR technique approximates $\mathbf{f}|\mathbf{u}$ and $\mathbf{f}^*|\mathbf{u}$ by deterministic functions of their means, that is:

$$\begin{aligned} p(\mathbf{f}|\mathbf{u}) &\approx q_{SoR}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{f}|K_{f,u}K_{u,u}^{-1}\mathbf{u}, 0) \\ p(\mathbf{f}|\mathbf{u}) &\approx q_{SoR}(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}^*|K_{*,u}K_{u,u}^{-1}\mathbf{u}, 0). \end{aligned} \tag{A.8}$$

Then:

$$q_{SoR}(\mathbf{f}, \mathbf{f}^*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{f,f} & Q_{f,*} \\ Q_{*,f} & Q_{*,*} \end{bmatrix}\right), \tag{A.9}$$

which results in the approximation $p(\mathbf{f}|\mathbf{y}) \approx q_{SOR}(\mathbf{f}|\mathbf{y})$:

$$q_{SOR}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}|Q_{*,f}(Q_{f,f}+\sigma^2 I)^{-1}\mathbf{y}, Q_{*,*} - Q_{*,f}(Q_{f,f}+\sigma^2 I)^{-1}Q_{f,*}). \tag{A.10}$$

Since the marginal distribution of $\mathbf{f}$ is approximated by $\mathbf{Q}_{\mathbf{f},\mathbf{f}}$, there is also an approximation for the likelihood of $\mathcal{D}$:

$$\log p(\mathcal{D}|M) \approx \log p(\mathbf{y}|M_{SoR}, \mathbf{x}_u) = \log \mathcal{N}(\mathbf{y}|0, Q_{f,f} + \sigma^2 I). \tag{A.11}$$

Notice that, since the matrix $Q_{f,f} = K_{f,u}K_{u,u}^{-1}K_{u,f}$ has low rank, one can use the matrix inversion lemma B.1 to calculate the inverse of $Q_{f,f} + \sigma^2 I$ with $\mathcal{O}(m^3)$ computational cost. If $m \ll n$, this gives a considerable gain in computation.

As noted in [103] subset of regressors approximation suffers from overconfident predictive variances, since the prior approximations for both training and testing points are degenerate, so caution must be taken with those.

A.2.1 Deterministic Training Conditional

The Deterministic Training Condition approximation, also called Projected Latent Variables when first proposed by [97], or Projected Process Approximation in [87], was originally proposed as a likelihood approximation for the training observations:

$$p(\mathbf{y}|\mathbf{f}) \approx \mathcal{N}(K_{f,u}K_{u,u}^{-1}\mathbf{u}, \sigma^2 I). \tag{A.12}$$

In the prior approximation framework, an equivalent formulation can be made from by making a deterministic approximation the training points $\mathbf{f}|\mathbf{u}$, leaving $\mathbf{f}^*|\mathbf{u}$ unchanged, unlike the SoR method resulting in:

$$\begin{aligned} p(\mathbf{f}|\mathbf{u}) &\approx q_{DTC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{f}|K_{f,u}K_{u,u}^{-1}\mathbf{u}, 0) \\ p(\mathbf{f}|\mathbf{u}) &\approx q_{DTC}(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}^*|K_{*,u}K_{u,u}^{-1}\mathbf{u}, K_{*,*} - Q_{*,*}), \end{aligned} \tag{A.13}$$

resulting in the joint prior approximation:

$$q_{DTC}(\mathbf{f}, \mathbf{f}^*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{f,f} & Q_{f,*} \\ Q_{*,f} & K_{*,*} \end{bmatrix}\right), \tag{A.14}$$

which results in the posterior approximation

$$q_{DTC}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}|Q_{*,f}(Q_{f,f} + \sigma^2 I)^{-1}\mathbf{y}, K_{*,*} - Q_{*,f}(Q_{f,f} + \sigma^2 I)^{-1}Q_{f,*}), \quad \text{(A.15)}$$

and the same data likelihood as in the subset of regressors case.

$$\log p(\mathcal{D}|M) \approx \log p(\mathbf{y}|M_{DTC}, \mathbf{x}_u) = \log \mathcal{N}(\mathbf{y}|0, Q_{f,f} + \sigma^2 I) \quad \text{(A.16)}$$

The DTC approximation improves considerably the predictive variances over the SoR approximation, while retaining the same predictive means. However, it has an inconsistency property in the fact that, for the training values $\mathbf{f}$, the covariance between then is computed differently whether they are considered as training values (in this case being $Q_{f,f}$) or as test values on the same points as the training points (being $K_{f,f}$), hence [85] claiming that it does not correspond exactly to a Gaussian Process. In practice, the advantage of it in relation of the DTC approximation compensates for this theoretical issue.

A.2.2 Fully Independent Training Conditional and Fully Independent Conditional

In the Fully Independent Training Conditional, originally proposed by [104] with the name Sparse Gaussian Processes using Pseudo-Inputs, there is also an likelihood approximation as in the original formulations of the DTC approximation:

$$p(\mathbf{y}|\mathbf{f}) \approx \mathcal{N}(K_{f,u}K_{u,u}^{-1}\mathbf{u}, \text{diag}(K_{f,f} - Q_{f,f}) + \sigma^2 I) \quad \text{(A.17)}$$

In the prior approximation framework, the FITC approximates $p(\mathbf{f}|\mathbf{u})$ by the product of its marginal distributions, thus making an independence approximation

for the training points, resulting in

$$p(\mathbf{f}|\mathbf{u}) \approx q_{FITC}(\mathbf{f}|\mathbf{u}) = \prod_{i=1} p(f_i|\mathbf{u}) = \mathcal{N}(\mathbf{f}|K_{f,u}K_{u,u}^{-1}\mathbf{u}, \mathrm{diag}(K_{f,f} - Q_{f,f})), \quad \text{(A.18)}$$

keeping $\mathbf{f}|\mathbf{u}$ unchanged as in the DTC approximation, resulting in the joint prior approximation:

$$q_{FITC}(\mathbf{f}, \mathbf{f}^*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{f,f} + \mathrm{diag}(K_{f,f} - Q_{f,f}) & Q_{f,*} \\ Q_{*,f} & K_{*,*} \end{bmatrix}\right), \quad \text{(A.19)}$$

If there is only one evaluation point $\mathbf{f}^* = (\mathbf{f}_1)$, the FITC approximation can be seen as a diagonal correction of the DTC approximation for $p(\mathbf{f}, \mathbf{f}^*|\mathbf{u})$. The FITC results in the posterior approximation

$$q_{FITC}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}|Q_{*,f}(Q_{f,f} + D)^{-1}\mathbf{y}, K_{*,*} - Q_{*,f}(Q_{f,f} + D_f)^{-1}Q_{f,*}), \quad \text{(A.20)}$$

Where $D_f := \sigma^2 I + \mathrm{diag}(K_{f,f} - Q_{f,f})$. Hence the matrix inversion term is still tractable by the matrix inversion lemma. Finally, the likelihood approximation for $\mathbf{y}$ is given by:

$$\log p(\mathcal{D}|M) \approx \log p(\mathbf{y}|M_{FITC}, \mathbf{x}_u) = \log \mathcal{N}(\mathbf{y}|0, Q_{f,f} + D_f). \quad \text{(A.21)}$$

The FITC approximation also has the same inconsistency property as the DTC approximation. In [104] it is proposed instead to approximate also the test points $\mathbf{f}^*|\mathbf{u}$ prior by $\prod_i p(f_i^*|u)$, recovering the consistency property, resulting in the Fully Independent Conditional (FIC) approximation:

$$\begin{aligned} q_{FIC}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}|&Q_{*,f}(Q_{f,f} + D)^{-1}\mathbf{y}, \\ &Q_{*,*} + D_* - Q_{*,f}(Q_{f,f} + D_f)^{-1}Q_{f,*}), \end{aligned} \quad \text{(A.22)}$$

where $D_* := \mathrm{diag}(K_{*,*} - Q_{*,*})$. For a single test point, the FITC approximation and the FIC approximation returns exactly the same predictive distribution. In practice,

the FITC approximation is used far more often than the FIC one.

A.3 Posterior approximation via variational free energy

Another popular approach to sparsification is given in [110], different in spirit from the ones presented above. To understand the idea of posterior approximations, consider:

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}^*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f} \tag{A.23}$$

We have

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}) &= \mathcal{N}(\mathbf{f}|K_{f,f}(K_{f,f}+\sigma^2 I)^{-1}\mathbf{y}, K_{f,f} - K_{f,f}(K_{f,f}+\sigma^2 I)^{-1}K_{f,f}) \\ &= \mathcal{N}(\mathbf{f}|\mu, A), \end{aligned}$$

and

$$p(\mathbf{f}^*|\mathbf{f}) = \mathcal{N}(\mathbf{f}^*|K_{*,f}(K_{f,f}+\sigma^2 I)^{-1}\mathbf{y}, K_{*,*} - K_{*,f}(K_{f,f}+\sigma^2 I)^{-1}K_{f,*}). \tag{A.24}$$

We then have, by (B.4):

$$p(\mathbf{f}^*|\mathbf{y}) := \mathcal{N}(\mathbf{f}|K_{*,f}K_{*,*}^{-1}\mu, K_{*,*} - K_{*,f}(K_{f,f}^{-1} - K_{f,f}^{-1}AK_{f,f}^{-1})K_{f,*}), \tag{A.25}$$

which by the definition of μ and A above yields the usual posterior distribution.

Now consider again inducing point $\mathbf{x}_u$ with corresponding values $\mathbf{f}_u$, assuming that $\mathbf{f}$ and $\mathbf{f}^*$ are independent given $\mathbf{f}_u$. Notice that this implies $\mathbf{f}^*$ and $\mathbf{y}$ independent given $\mathbf{f}_u$, since $\mathbf{y}$ only depends on $\mathbf{f}$. Then, marginalizing on the inducing values:

$$p(\mathbf{f}^*|\mathbf{y}) \approx q(\mathbf{f}^*) = \int p(\mathbf{f}^*|\mathbf{f}_u, \mathbf{y})p(\mathbf{f}_u|\mathbf{y})d\mathbf{f}_u = \int p(\mathbf{f}^*|\mathbf{f}_u)p(\mathbf{f}_u|\mathbf{y})d\mathbf{f}_u. \tag{A.26}$$

Therefore, changing $\mathbf{f}$ for $\mathbf{f}_u$, (A.25) still holds. Since the true posterior $p(\mathbf{f}_u|\mathbf{y})$ includes the inverse of $K_{f,f}+\sigma^2 I$, one option is to approximate $p(\mathbf{f}_u|\mathbf{y})$ by a distribution $\phi(\mathbf{f}_u)$, also Gaussian with mean μ and covariance A, so that

$$q(\mathbf{f}^*) = \int p(\mathbf{f}^*|\mathbf{f}_u)\phi(\mathbf{f}_u) = \int q(\mathbf{f}^*, \mathbf{f}_u), \tag{A.27}$$

in an manner that (A.25) yields an sparse approximation.

One way to do this is to seek approximating the posterior $p(\mathbf{f}|\mathbf{y})$ itself by some distribution $q(\mathbf{f})$, involving $\phi(\mathbf{f}_u)$ on the training evaluations. As a proxy for this objective, the VFE method seeks to approximate the posterior for training and inducing values $p(\mathbf{f}, \mathbf{f}_u|\mathbf{y})$ by $q(\mathbf{f}, \mathbf{f}_u)$, which, by (A.27) is of form $q(\mathbf{f}, \mathbf{f}_u) = p(\mathbf{f}|\mathbf{f}_u)\phi(\mathbf{f}_u)$. Crucially, this augmented posterior depends on the inducing points $\mathbf{x}_u$ through $p(\mathbf{f}|\mathbf{f}_u)$, making then parameters of this approximate distribution, but not of the prior model $p(\mathbf{f})$.

A natural way to find $q(\mathbf{f}, \mathbf{f}_u)$ is by minimizing the Kullback-Leibner divergence between $q(\mathbf{f}, \mathbf{f}_u)$ and $p(\mathbf{f}, \mathbf{f}_u|\mathbf{y})$

$$KL(q(\mathbf{f}, \mathbf{f}_u)||p(\mathbf{f}, \mathbf{f}_u|\mathbf{y}) = -\int\int q(\mathbf{f}, \mathbf{f}_u) \log \frac{p(\mathbf{f}, \mathbf{f}_u|\mathbf{y})}{q(\mathbf{f}, \mathbf{f}_u)} d\mathbf{f} d\mathbf{f}_u,$$

which is equivalent to maximizing the evidence lower bound, or variational free energy (using $p(\mathbf{f}, \mathbf{f}_u|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{f}_u)p(\mathbf{f}_u)$)

$$\begin{aligned} F_V(q(\mathbf{f}, \mathbf{f}_u)) = F_V(\mathbf{x}_u, \phi) &= \int\int p(\mathbf{f}|\mathbf{f}_u)\phi(\mathbf{f}_u) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{f}_u)p(\mathbf{f}_u)}{p(\mathbf{f}|\mathbf{f}_u)\phi(\mathbf{f}_u)} d\mathbf{f} d\mathbf{f}_u \\ &= \int \phi(\mathbf{f}_u) \left(\int p(\mathbf{f}|\mathbf{f}_u) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log \frac{p(\mathbf{f}_u)}{\phi(\mathbf{f}_u)} \right) d\mathbf{f}_m. \end{aligned} \tag{A.28}$$

This quantity is maximized by maximizing

$$F_V(\mathbf{x}_u) = \log \mathcal{N}(\mathbf{y}|0, Q_{f,f} + \sigma^2 I) - \frac{1}{2\sigma^2} \mathrm{tr}(K_{f,f} - Q_{f,f}), \tag{A.29}$$

and setting $\phi(\mathbf{f}_u) = \mathcal{N}(\mathbf{f}_u|\mu^\dagger, A^\dagger)$, with

$$\begin{aligned} \mu^\dagger &= \sigma^{-2} K_{u,u} (K_{u,u} + \sigma^{-2} K_{u,f} K_{f,u})^{-1} K_{u,f} \mathbf{y} \\ A^\dagger &= K_{u,u} (K_{u,u} + \sigma^{-2} K_{u,f} K_{f,u})^{-1} K_{u,u}. \end{aligned} \tag{A.30}$$

The proof is given in E.1. Substituting back in (A.25), we arrive at

$$\begin{aligned} q_{VFE}(\mathbf{f}^*|\mathbf{y}) &:= \mathcal{N}(\mathbf{f}|\mathbf{m}^*_{VFE}, \Sigma^*_{VFE}) \\ \mathbf{m}^*_{VFE} &= \sigma^{-2} K_{*,u}(K_{u,u} + \sigma^{-2} K_{u,f} K_{f,u})^{-1} K_{u,f}\mathbf{y} \\ \Sigma^*_{VFE} &= K_{*,*} - Q_{*,*} + K_{*,u}(K_{u,u} + \sigma^{-2} K_{u,f} K_{f,u})^{-1} K_{u,*}. \end{aligned} \tag{A.31}$$

It can be show (E.2) that these predictions correspond exactly to the DTC prediction. Thus, the VFE approach differs only in how the inducing points and kernel hyperparameters are trained, by maximizing (A.29) instead of (A.16). Recent improvements of the VFE approach is be found in [21], where an unification of the VFE and FITC approaches are proposed.

A.3.1 Bayesian Monte Carlo with Sparse Gaussian Processes

The extension for Bayesian Monte Carlo for the inducing points methods for sparsification presented in the previous section is straightforward. We will consider here only the FITC and the DTC approximations (being the case for the VFE approximation exactly the same as the DTC one). Considering $D = \sigma^2 I$ in the VFE approximation and $D = \sigma^2 I + \text{diag}(K_{f,f} - Q_{f,f})$ in the FITC one, by substituting the predictive mean and variance of (A.20) and (A.15) in (4.3) and (4.4),

$$\begin{aligned} \mathbb{E}[Z_{\mathcal{D}}] &= \mathbf{z}_u K_{u,u}^{-1} K_{u,f}(Q_{f,f} + D)^{-1}\mathbf{y} \\ \text{Var}[Z_{\mathcal{D}}] &= \Gamma - \mathbf{z}_u^T K_{u,u}^{-1} K_{u,f}(Q_{f,f} + D) K_{f,u} K_{u,u}^{-1}\mathbf{z}_u, \end{aligned} \tag{A.32}$$

where

$$z_{u,i} = \int k(x, x_{u,i}) p(x) dx, \tag{A.33}$$

and Γ is the same as given in (4.7).

A.3.2 VBMC and BVBMC with Sparse Gaussian Processes

One of the techniques that was tried to expand the BVBMC method to wider applications was to use one of the sparse GP techniques shown here. However, it was found that, under low noise, the resulting matrices where very unstable, while when forcing artificially high noise, the results became innacurate.

It should be noted that, in that stage, only the SQE kernel was used, and it remains to be seen whether this problem still arises when using product of Matern kernels, which was a later development in this work.

APPENDIX B RELEVANT GAUSSIAN AND MATRIX IDENTITIES

In the following, it is presented some relevant matrix and Gaussian distribution identities. All of those identities can be found in [81], except for (B.3), which is slightly more general and can be found in [85].

B.1 Matrix inversion lemma

If all the relevant inverses exists, then

$$(Z + UWV^T) = Z^{-1} - Z^{-1}U(W^{-1} + V^T Z^{-1}U)^{-1}VZ^{-1}. \tag{B.1}$$

One consequence of the matrix inverse lemma is the formula

$$\begin{aligned}(D + A)^{-1} &= (D + DD^{-1}AD^{-1}D)^{-1} = \\ &= (D - D^{-1}D((D^{-1}AD^{-1})^{-1} + DD^{-1}D)DD^{-1}) = \\ &= D^{-1} - (D + DA^{-1}D)^{-1}.\end{aligned} \tag{B.2}$$

B.2 Product of Gaussian densities

$$\mathcal{N}(x|a, A)\mathcal{N}(Px|b, B) = z_c\mathcal{N}(x|c, C), \tag{B.3}$$

where

$$c = (A^{-1} + PB^{-1}P^T)^{-1}, \quad c = C(A^{-1}a + P^T B^{-1}b),$$

and

$$z_c = \mathcal{N}(Pa|b, B + P^T AP) = \mathcal{N}(b|Pa, B + P^T AP).$$

In particular, this implies that

$$\int \mathcal{N}(b|Px, B)\mathcal{N}(x|a, A)dx = \mathcal{N}(b|Pa, B + PAP^T). \tag{B.4}$$

B.3 Conditional of a Gaussian density

If

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right). \tag{B.5}$$

then

$$\mathbf{x}_1|\mathbf{x}_2 \sim \mathcal{N}\big(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\big). \tag{B.6}$$

APPENDIX C ALTERNATIVE DERIVATION OF GP PREDICTIONS

Consider:

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}, \mathbf{f}^*|\mathbf{y}) d\mathbf{f} = \frac{1}{p(\mathbf{y})} \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, \mathbf{f}^*) d\mathbf{f}. \tag{C.1}$$

By letting P_f be the projection $(\mathbf{f}, \mathbf{f}^*) \to \mathbf{f}$, and equivalently for P_*, and letting $K = K((\mathbf{x}, \mathbf{x}^*), (\mathbf{x}, \mathbf{x}^*))$, then

$$\begin{aligned} p((\mathbf{f}, \mathbf{f}^*|\mathbf{y})) \propto p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, \mathbf{f}^*) &= \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 I) \mathcal{N}((\mathbf{f}, \mathbf{f}^*)|0, K) \\ &= \mathcal{N}(P_f(\mathbf{f}, \mathbf{f}^*)|\mathbf{y}, \sigma^2 I) \mathcal{N}((\mathbf{f}, \mathbf{f}^*|0, K)) \\ &\propto \mathcal{N}((\mathbf{f}, \mathbf{f}^*)|c, C), \end{aligned} \tag{C.2}$$

where

$$C = (K^{-1} + P_f^T \sigma^{-2} I P_f)^{-1}, \quad c = C P_f \sigma^{-2} \mathbf{y}. \tag{C.3}$$

By the matrix inversion lemma

$$C = K - K P_f^T (P_f K P_f^T + \sigma^2 I)^{-1} P_f K. \tag{C.4}$$

Hence, by (B.4),

$$p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}|P_* c, P_* C P_*^T). \tag{C.5}$$

We have for the posterior covariance

$$\begin{aligned} P_* C P_*^T &= P_* K P_*^T - P_* K P_f^T (P_f K P_f^T + \sigma^2 I)^{-1} P_f K P_* \\ &= K_{*,*} - K_{*,f} (K_{f,f} + \sigma^2 I) K_{f,*}, \end{aligned} \tag{C.6}$$

and for the posterior mean

$$\begin{aligned} P_* c &= P_* K P_f \sigma^{-2} \mathbf{y} - P_* K P_f^T (P_f K P_f^T + \sigma^2 I)^{-1} P_f K P_f^T \mathbf{y} \\ &= K_{*,f} (\sigma^{-2} I - (K_{f,f} + \sigma^2 I)^{-1} K_{f,f} \sigma^{-2} I) \mathbf{y} \\ &= K_{*,f} (\sigma^{-2} I - (\sigma^2 I + \sigma^4 K_{f,f}^{-1})) \mathbf{y} \\ &= K_{*,f} (K_{f,f} + \sigma^2 I) \mathbf{y}. \end{aligned} \tag{C.7}$$

where the last inequality comes from (B.2) considering $D = \sigma^2 I$. Thus this derivation yields the same posterior distribution as in (3.7).

APPENDIX D SPECTRAL MIXTURE KERNELS AND BAYESIAN MONTE CARLO

A kernel-distribution combination that yields analytical mean and variances for Bayesian Monte Carlo is the spectral mixture kernel in (3.14) combined with a Gaussian distribution $p(x) = \mathcal{N}(x|\mu, \Sigma)$ with diagonal covariance $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_D^d)$, yielding

$$
\begin{aligned}
z_i = \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} \Big[& (2\pi v_q^{(d)})^{-0.5} \cos\left(2\pi C_q^{(d)} \mu_q^{(d)} (x_{i,d} - m_d)\right) e^{-\pi (C_q^{(d)})^2 \mu_q^{(d)}} \\
& \mathcal{N}(x_{i,d}|m_d, (4\pi^2 v_q^{(d)})^{-1} + \sigma_d^2) \Big] \\
\Gamma = \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} & \left[(2\pi v_q^{(d)})^{-0.5} e^{-\pi (C_q^{(d)})^2 \mu_q^{(d)}} \exp\left(-\frac{1}{2}(A_q^{(d)} b_{q,d})^2\right) (2\pi(\nu_{q,d}^2 + \sigma_d^2))^{-0.5} \right] \\
\text{where} \quad & C_q^{(d)} = (4\pi v_q^{(d)} + \sigma_d^{-2})^{-1},\ b_{q,d} = 2\pi C_q^{(d)} \mu_q^{(d)}, \\
& \nu_{q,d}^2 = (4\pi^2 v_q^{(d)})^{-1} + \sigma_d^2,\ A_q^{(d)} = (\nu_{q,d}^{-2} + \sigma_d^{-2})^{-1}.
\end{aligned}
\tag{D.1}
$$

This combination interesting because the spectral mixture kernel is far more flexible than the squared exponential one, thus enabling using Bayesian Monte Carlo to calculate expectations of more complex functions. Next it is shown the derivation for those formulas.

We will need first the result:

$$
\begin{aligned}
& \int \cos(b(x-m)) \mathcal{N}(x|m, \nu^2) \mathcal{N}(x|\mu, \sigma^2) dx \\
& = \cos(Cb(m-\mu)) \exp\left(-\frac{1}{2} C^2 b^2\right) \mathcal{N}(m|\mu, v^2 + \sigma^2) \\
& \text{where} \quad C = (\nu^{-2} + \sigma^{-2})^{-1}.
\end{aligned}
\tag{D.2}
$$

To get it, use (B.3) in the integral, so it equals to:

$$\begin{aligned}
&\mathcal{N}(m|\mu, \nu^2 + \sigma^2) \int \cos(b(x-m))\mathcal{N}(x|c, C)dx = \\
&\mathcal{N}(m|\mu, \nu^2 + \sigma^2)\Re\left[\int e^{ib(x-m)}\mathcal{N}(x|c, C)dx\right] = \\
&\mathcal{N}(m|\mu, \nu^2 + \sigma^2)\Re\left[e^{-ibm}\int e^{ibx}\mathcal{N}(x|c, C)\right] = \\
&\mathcal{N}(m|\mu, v^2 + \sigma^2)\Re\left[e^{-ibm}e^{ibc-\frac{1}{2}C^2b^2}\right] = \\
&\mathcal{N}(m|\mu, \nu^2 + \sigma^2)e^{-\frac{1}{2}C^2b^2}\cos(b(c-m)), \\
&\text{where} \quad c = C(\nu^{-2}m + \sigma^{-2}\mu).
\end{aligned}$$

From the third to the fourth line above, we use the formula for the characteristic function of a Gaussian distribution. Finally, since $c-m = C(\nu^{-2}m+\sigma^{-2}\mu-C^{-1}m) = C\sigma^{-2}(\mu - m)$, and using the symmetry of cosine, the equality follows.

Letting $p(x) = \mathcal{N}(x|b, \text{diag}(\sigma_1^2, \ldots, \sigma_D^2))$ and $k(x, x_i) = k_{SM}(x - x_i)$:

$$\begin{aligned}
&\int k_{SM}(x - x_i)\mathcal{N}(x|m, \text{diag}(\sigma_1^2, \ldots, \sigma_D^2))dx \\
&= \sum_{q=1}^{Q} w_q \int \left[\prod_{d=1}^{D} e^{-2\pi^2(x_d - x_{i,d})^2 v_q^{(d)}} \cos(2\pi(x_d - x_{i,d})\mu_q^{(d)}) \prod_{d=1}^{D} \mathcal{N}(x_d|m_d, \sigma_d^2)dx_d\right] \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} \int e^{-2\pi^2(x_d - x_{i,d})^2 v_q^{(d)}} \cos(2\pi(x_d - x_{i,d})\mu_q^{(d)})\mathcal{N}(x_d|m_d, \sigma_d^2)dx_d \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} (2\pi v)^{-0.5} \int \cos(2\pi\mu_q^{(d)}(x_d - x_{i,d}))\mathcal{N}(x_d|x_{i,d}, (4\pi^2 v)^{-1})\mathcal{N}(x_d|m_d, \sigma_d^2)dx_d \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} (2\pi v_q^{(d)})^{-0.5} \int \cos(2\pi\mu_q^{(d)}(x_d - x_{i,d}))\mathcal{N}(x_d|x_{i,d}, (4\pi^2 v)^{-1})\mathcal{N}(x_d|m_d, \sigma_d^2)dx_d \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} (2\pi v_q^{(d)})^{-0.5} \cos\left(2\pi C_q^{(d)}\mu_q^{(d)}(x_{i,d} - m_d)\right) e^{-\pi(C_q^{(d)})^2\mu_q^{(d)}} \mathcal{N}(x_{i,d}|m_d, (4\pi^2 v_q^{(d)})^{-1} + \sigma_d^2) \\
&\text{where} \quad C_q^{(d)} = (4\pi^2 v_q^{(d)} + \sigma_d^{-2})^{-1},
\end{aligned} \tag{D.3}$$

and, letting $b_{q,d} = 2\pi C_q^{(d)} \mu_q^{(d)}$ and $\nu_{q,d}^2 = (4\pi^2 v_q^{(d)})^{-1} + \sigma_d^2$

$$
\begin{aligned}
&\int\int k_{SM}(x_i - x_j)\mathcal{N}(x_i|\mu, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^2))\mathcal{N}(x_j|\mu, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^2))dx_i, dx_j \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} \Bigg[(2\pi v_q^{(d)})^{-0.5} e^{-\pi (C_q^{(d)})^2 \mu_q^{(d)}} \\
&\qquad\qquad \int \cos\left(b_{q,d}(x_{i,d} - m_d)\right) \mathcal{N}(x_{i,d}|m_d, \nu_{q,d}) \mathcal{N}(x_{j,d}|m_d, \sigma_d^2) dx_{j,d} \Bigg] \\
&= \sum_{q=1}^{Q} w_q \prod_{d=1}^{D} \left[(2\pi v_q^{(d)})^{-0.5} e^{-\pi (C_q^{(d)})^2 \mu_q^{(d)}} \exp\left(-\frac{1}{2}(A_q^{(d)} b_{q,d})^2\right) (2\pi(\nu_{q,d}^2 + \sigma_d^2))^{-0.5} \right] \\
&\text{where} \quad A_q^{(d)} = (\nu_{q,d}^{-2} + \sigma_d^{-2})^{-1}.
\end{aligned}
\tag{D.4}
$$

APPENDIX E DERIVATIONS FOR VFE

E.1 Maximizaton of variational free energy

For the integral inside parenthesis in (A.29), letting $\alpha = K_{f,u}K_{u,u}\mathbf{f}$ and $M = K_{f,f} - Q_{f,f}$, $p(\mathbf{f}|\mathbf{f}_u) = \mathcal{N}(\mathbf{f}|\alpha, M)$:

$$\begin{aligned}\int p(\mathbf{f}|\mathbf{f}_u)\log p(\mathbf{y}|\mathbf{f})d\mathbf{f} &= \int p(\mathbf{f}|\mathbf{f}_u)\left[-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{f})^T(\mathbf{y}-\mathbf{f}) - \frac{d}{2}\log(2\pi\sigma^2)\right]d\mathbf{f} \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \mathbb{E}_{\mathbf{f}|\mathbf{f}_u}\left[\mathrm{tr}\left(\frac{1}{2\sigma^2}(\mathbf{f}-\mathbf{y})(\mathbf{f}-\mathbf{y})^T\right)\right] \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\mathrm{tr}((\alpha-\mathbf{y})(\alpha-\mathbf{y})^T + A) = \\ &= \log\mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I) - \mathrm{tr}(K_{f,f} - Q_{f,f}).\end{aligned}$$

Substituing back in (A.29):

$$F_V(\mathbf{x}_u,\phi) = \int \phi(\mathbf{f}_u)\log\frac{\mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I)p(\mathbf{f}_u)}{\phi(\mathbf{f}_u)}d\mathbf{f}_u - \mathrm{tr}(K_{f,f} - Q_{f,f}). \tag{E.1}$$

For any $\mathbf{x}_u$ fixed, $F_V(\mathbf{x}_u,\phi)$ is then maximized by maximizing the integral term. But this is just the evidence lower bound between ϕ and the unnormalized distribution $\mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I)p(\mathbf{f}_u)$, relative to $\mathbf{f}_u$. Without constraining ϕ, then we must have:

$$\begin{aligned}\phi(\mathbf{f}) &\propto \mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I)p(\mathbf{f}_u) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y}-\alpha)^T(\mathbf{y}-\alpha) - \frac{1}{2}\mathbf{f}_u^T K_{u,u}\mathbf{f}_u\right) \\ &\propto \exp\left(\frac{1}{\sigma^2}\mathbf{y}^T K_{f,u}K_{u,u}^{-1}\mathbf{f}_u - \frac{1}{2}\mathbf{f}_u^T\left(K_{u,u}^{-1} + \frac{1}{\sigma^2}K_{u,u}^{-1}K_{u,f}K_{f,u}K_{u,u}^{-1}\right)\mathbf{f}_u\right).\end{aligned} \tag{E.2}$$

The quadratic form can be completed relative to $\mathbf{f}_u$ to find:

$$\begin{aligned}A^\dagger &= (K_{u,u}^{-1} + \sigma^{-2}K_{u,u}^{-1}K_{u,f}K_{f,u}K_{u,u}^{-1})^{-1} = K_{u,u}(K_{u,u} + \sigma^{-2}K_{u,f}K_{f,u})^{-1}K_{u,u} \\ \mu^\dagger &= \sigma^{-2}K_{u,u}(K_{u,u} + \sigma^{-2}K_{u,f}K_{f,u})^{-1}K_{u,f}\mathbf{y}.\end{aligned} \tag{E.3}$$

Since ϕ is itself Gaussian, the unconstrained optimum is the same as for the constrained one. Finally, since $\phi(\mathbf{f}) = \frac{1}{Z}\mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I)p(\mathbf{f}_u)$, with $Z = \int \mathcal{N}(\mathbf{y}|\alpha,\sigma^2 I)p(\mathbf{f}_u)d\mathbf{f}_u$,

substituting back into (E.1), we find the objective function for $\mathbf{x}_u$,

$$\begin{aligned} F_V(\mathbf{x}_u) &= \log \int \mathcal{N}(\mathbf{y}|K_{f,u}K_{u,u}\mathbf{f}, \sigma^2 I)\mathcal{N}(0, K_{u,u})d\mathbf{f}_u - \text{tr}(K_{f,f} - Q_{f,f}) \\ &= \log \mathcal{N}(\mathbf{y}|0, Q_{f,f} + \sigma^2 I) - \text{tr}(K_{f,f} - Q_{f,f}). \end{aligned} \tag{E.4}$$

E.2 Equivalence between VFE and DTC prediction

Starting with (A.31), for the covariance, we use the matrix inversion lemma, by letting $\Delta = \sigma^2 I$:

$$\begin{aligned} &K_{*,u}(K_{u,u} + K_{u,f}\Delta^{-1}K_{f,u})^{-1}K_{u,*} = \\ &K_{*,u}(K_{u,u}^{-1} - K_{u,u}^{-1}K_{u,f}(K_{f,u}K_{u,u}^{-1}K_{u,f} + \Delta)^{-1}K_{f,u}K_{u,u}^{-1})K_{u,*} = \\ &Q_{*,*} - Q_{*,f}(Q_{f,f} + \Delta)^{-1}Q_{f,*}. \end{aligned}$$

Substituting in the covariance term for (A.31), we find the same covariance as in (A.15). For the mean term, using again the matrix inversion lemma, we find:

$$\begin{aligned} &K_{*,u}(K_{u,u} + K_{u,f}\Delta^{-1}K_{f,u})^{-1}K_{u,f}\Delta^{-1}\mathbf{y} = \\ &K_{*,u}(K_{u,u}^{-1} - K_{u,u}^{-1}K_{u,f}(K_{f,u}K_{u,u}^{-1}K_{u,f} + \Delta)^{-1}K_{f,u}K_{u,u}^{-1})K_{u,f}\Delta^{-1}\mathbf{y} = \\ &(Q_{*,f} - Q_{*,f}(Q_{f,f} + \Delta)^{-1}Q_{f,f})\Delta^{-1}\mathbf{y} = \\ &Q_{*,f}(\Delta^{-1} - (Q_{f,f} + \Delta)^{-1}Q_{f,f}\Delta^{-1})\mathbf{y} = \\ &Q_{*,f}(\Delta^{-1} - (\Delta Q_{f,f}^{-1})(Q_{f,f} + \Delta))^{-1}\mathbf{y} = \\ &Q_{*,f}(\Delta^{-1} - (\Delta + \Delta Q_{f,f}^{-1}\Delta)^{-1})\mathbf{y} = \\ &Q_{*,f}(Q_{f,f} + \Delta)^{-1}\mathbf{y}, \end{aligned}$$

where in the last equality it was used (B.2). This is the same mean term as in (A.15), thus proving the equality.

APPENDIX F REINFORCE GRADIENT

We have that

$$\begin{aligned}
&\nabla \mathbb{E}_{q(\theta;\lambda)}\left[\log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right)\right] \\
&= \nabla \int \log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) q(\theta;\lambda) d\theta \\
&= \int \nabla_\lambda \left(\log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) q(\theta;\lambda)\right) d\theta \\
&= \int q(\theta;\lambda) \nabla_\lambda \log q(\theta;\lambda) d\theta + \int \log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) \nabla_\lambda q(\theta;\lambda) d\theta
\end{aligned} \quad . \tag{F.1}$$

The first term in the sum equals to

$$\begin{aligned}
\int q(\theta;\lambda) \nabla_\lambda \log q(\theta;\lambda) d\theta &= \int q(\theta;\lambda) \frac{\nabla_\lambda q(\theta;\lambda)}{q(\theta;\lambda)} d\theta \\
&= \nabla \int q(\theta;\lambda) d\theta \\
&= \nabla 1 = 0,
\end{aligned} \tag{F.2}$$

while the second term equals to

$$\begin{aligned}
&\int \log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) \nabla_\lambda q(\theta;\lambda) d\theta \\
&= \int \log\left(\frac{\bar{g}(\theta)}{q(\theta;\lambda)}\right) \nabla_\lambda (\log q(\theta;\lambda)) q(\theta;\lambda) d\theta \\
&= \mathbb{E}_{q(\theta;\lambda)}\left[\log \frac{\bar{g}(\theta)}{q(\theta;\lambda)} \nabla_\lambda \log q(\theta;\lambda)\right]
\end{aligned} \tag{F.3}$$

Joining the two terms, we arrive at (5.16).